Georg Rehm · Stefan Dietze ·
Sonja Schimmler · Frank Krüger (Eds.)

LNAI 14770

Natural Scientific Language Processing and Research Knowledge Graphs

First International Workshop, NSLP 2024
Hersonissos, Crete, Greece, May 27, 2024
Proceedings

Springer

Lecture Notes in Computer Science

Lecture Notes in Artificial Intelligence 14770

Founding Editor

Jörg Siekmann

Series Editors

The series Lecture Notes in Artificial Intelligence (LNAI) was established in 1988 as a topical subseries of LNCS devoted to artificial intelligence.

The series publishes state-of-the-art research results at a high level. As with the LNCS mother series, the mission of the series is to serve the international R & D community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings.

Georg Rehm · Stefan Dietze · Sonja Schimmler ·
Frank Krüger
Editors

Natural Scientific Language Processing and Research Knowledge Graphs

First International Workshop, NSLP 2024
Hersonissos, Crete, Greece, May 27, 2024
Proceedings

Springer

Editors
Georg Rehm
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Berlin, Germany

Stefan Dietze
GESIS Leibniz Institut für Sozialwissenschaften and Heinrich Heine University Düsseldorf
Cologne, Germany

Sonja Schimmler
Technical University of Berlin and Fraunhofer FOKUS
Berlin, Germany

Frank Krüger
Wismar University of Applied Sciences
Wismar, Germany

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Artificial Intelligence
ISBN 978-3-031-65793-1 ISBN 978-3-031-65794-8 (eBook)
https://doi.org/10.1007/978-3-031-65794-8

LNCS Sublibrary: SL7 – Artificial Intelligence

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Scientific research is almost exclusively published in unstructured text formats, which are not readily machine-readable. While technological approaches can help to get this flood of scientific information and new knowledge under control, the development of such technologies is very complex in practice and hinders the creation of infrastructures and systems to track research and assist the scientific community with applications such as dedicated scientific search engines and recommender systems. The 1st Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024) aimed to bring together researchers working on the processing, analysis, transformation and making-use-of scientific language and research knowledge graphs including all relevant sub-topics. NSLP 2024 was held as a full-day workshop co-located with the Extended Semantic Web Conference (ESWC 2024) in Hersonissos, Greece, on 27 May 2024.[1]

In addition to the opportunity to submit papers covering original research that fits the workshop's topics of interest, the event also offered two shared tasks: *Field of Research Classification of Scholarly Publications* (FoRC) and *Software Mention Detection in Scholarly Publications* (SOMD). Participants could sign up for one or both shared tasks and also for one or more of the respective (sub-)tasks. Automated evaluations of submitted systems were carried out through the platform Codalab. This proceedings volume contains several short papers that report on submitted systems and also overview papers that report on the two shared tasks on a general level. We will attempt to organise similar shared tasks in future editions of this workshop.

With NSLP 2024 as the first edition of what is intended to be a workshop series, we were very happy about a total of 26 submissions out of which 21 papers were accepted (81%) after a thorough, double-blind peer-review process with three reviews per submission. The NSLP 2024 workshop consisted of paper and poster presentations (including overviews of the results of the two shared tasks and shared task contributions) as well as two invited keynotes, given by Natalia Manola (OpenAIRE, Greece) and Francesco Osborne (Open University, UK).

We would like to thank the ESWC 2024 organisers and overall workshop chairs for accepting our workshop proposal. We would also like to extend our gratitude to the keynote speakers for their insightful talks. Thanks are also due to the members of the Programme Committee for reviewing the paper submissions under rather tight deadlines. Finally, we would like to thank Raia Abu Ahmad and Ekaterina Borisova for setting up and maintaining the workshop website.

This workshop was organised under the umbrella of the project NFDI for Data Science and Artificial Intelligence (NFDI4DS), which is part of the wider German NFDI

[1] https://nfdi4ds.github.io/nslp2024/

initative (National Research Data Infrastructure). Without the financial support of this project, neither the workshop nor this proceedings volume would have been possible.

May 2024

Georg Rehm
Stefan Dietze
Sonja Schimmler
Frank Krüger

Organization

Organizing Committee and Program Committee Chairs

Georg Rehm	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
Stefan Dietze	GESIS Leibniz Institut für Sozialwissenschaften & Heinrich Heine University Düsseldorf, Germany
Sonja Schimmler	Technical University of Berlin & Fraunhofer FOKUS, Germany
Frank Krüger	Wismar University of Applied Sciences, Germany

Program Committee

Rana Abdullah	Universität Hamburg, Germany
Raia Abu Ahmad	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
Marcel R. Ackermann	Schloss Dagstuhl – Leibniz Center for Informatics (LZI), Germany
Mathieu d'Aquin	Université de Lorraine, France
Ekaterina Borisova	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
Davide Buscaldi	LIPN, CNRS, University Paris 13, France
Catherine Faron	Université Côte d'Azur, France
Daniel Garijo	Universidad Politécnica de Madrid, Spain
Paul Groth	University of Amsterdam, Netherlands
Robert Jäschke	Humboldt-Universität zu Berlin, Germany
Pedro Ortiz Suarez	Common Crawl Foundation, USA
Wolfgang Otto	GESIS Leibniz Institut für Sozialwissenschaften, Germany
Haris Papageorgiou	R. C. Athena, Greece
Simone Ponzetto	University of Mannheim, Germany
Diego Reforgiato Recupero	University of Cagliari, Italy
Philipp Schaer	TH Köln, Germany
Jennifer D'Souza	TIB Leibniz-Informationszentrum Technik und Naturwissenschaften, Germany

Tilahun Taffa	Universität Hamburg, Germany
Atsuhiro Takasu	University of Tokyo, Japan
Ricardo Usbeck	Leuphana University, Germany

Shared Tasks Committee

Georg Rehm	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
Raia Abu Ahmad	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
Ekaterina Borisova	Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Germany
Stefan Dietze	GESIS Leibniz Institut für Sozialwissenschaften & Heinrich Heine University Düsseldorf, Germany
Saurav Karmarkar	GESIS Leibniz Institut für Sozialwissenschaften, Germany
Frank Krüger	Wismar University of Applied Sciences, Germany

Sponsor

The NSLP 2024 workshop and this proceedings volume were financially supported by the consortium *NFDI for Data Science and Artificial Intelligence (NFDI4DS)* as part of the non-profit association National Research Data Infrastructure (NFDI e. V.). The consortium is funded by the Federal Republic of Germany and its states through the German Research Foundation (DFG) project NFDI4DS (no. 460234259).

https://www.nfdi4datascience.de

Contents

Scholarly Information Processing

Scholarly Question Answering Using Large Language Models in the NFDI4DataScience Gateway

Hamed Babaei Giglou[1], Tilahun Abedissa Taffa[2,3](✉), Rana Abdullah[3], Aida Usmanova[2], Ricardo Usbeck[2], Jennifer D'Souza[1], and Sören Auer[1]

[1] TIB Leibniz Information Centre for Science and Technology, Hannover, Germany
{hamed.babaei,jennifer.dsouza,auer}@tib.eu

[2] Artificial Intelligence and Explainability, Leuphana Universität Lüneburg, Lüneburg, Germany
aida.usmanova@stud.leuphana.de, ricardo.usbeck@leuphana.de

[3] Semantic Systems, Universität Hamburg, Hamburg, Germany
{tilahun.taffa,rana.abdullah}@uni-hamburg.de

https://www.leuphana.de/en/institutes/iis/artifical-intelligence-and-explainability.html

Abstract. This paper introduces a scholarly Question Answering (QA) system on top of the NFDI4DataScience Gateway, employing a Retrieval Augmented Generation-based (RAG) approach. The NFDI4DS Gateway, as a foundational framework, offers a unified and intuitive interface for querying various scientific databases using federated search. The RAG-based scholarly QA, powered by a Large Language Model (LLM), facilitates dynamic interaction with search results, enhancing filtering capabilities and fostering a conversational engagement with the Gateway search. The effectiveness of both the Gateway and the scholarly QA system is demonstrated through experimental analysis.

Keywords: Scholarly Question Answering · Federated Search · Retrieval Augmented Generation · Large Language Models · NFDI4DS Gateway

1 Introduction

With recent advances in artificial intelligence (AI), decision-making has gradually shifted from rule-based systems to machine learning and deep learning-based developments [11]. This paradigm shift has changed how we approach information retrieval and Question Answering (QA) systems, including Scholarly QA. Scholarly QA systems answer natural language questions over bibliographic data sources [2,26]. Notably, scholarly resources appear in different bibliographic

H. Babaei Giglou, T. A. Taffa, R. Abdullah and A. Usmanova—Equal contribution.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 3–18, 2024.
https://doi.org/10.1007/978-3-031-65794-8_1

repositories. To narrow down the answer search space - federated search comes into play. A federated search platform enables one to navigate the vast landscape of scholarly resources available across multiple databases and repositories [8]. Furthermore, federated search aggregates information from multiple sources to provide a comprehensive and holistic view of relevant resources. The efficacy of faceted search in scholarly-based filtering has been well-demonstrated [15], paving the way for robust systems employing federated search methods.

Adhering to FAIR principles [29] in managing research data, initiatives like the NFDI4DataScience[1] (NFDI4DS) consortium have emerged as a collaborative endeavor designed to support researchers throughout the entire research data life cycle, ensuring their practices align with the FAIR principles. The NFDI4DS Gateway, as a part of the NFDI4DS vision [22], includes a federated search. The Gateway - a unified and intuitive search interface that enables users to query various scientific databases such as DBLP, Zenodo, and OpenAlex. The overall aim of the NFDI4DS Gateway is to design an entry point that categorizes and summarises multiple search results (such as researchers, publications, machine learning models, and benchmark results) such that practitioners and researchers gain a swift overview of existing contributions [27].

Retrieval Augmented Generation (RAG) harnesses the power of advanced natural language processing (NLP) techniques to improve the quality and relevance of responses to user queries. Integrating Large Language Model (LLM)-based components is at the core of RAG's functionality. LLMs are the backbone of RAG's response generation process, leveraging extensive training on large text to understand and generate human-like responses. RAG-based scholarly QA systems can seamlessly integrate with federated search to improve the process of filtering and selecting scholarly resources in the context of scholarly research [12]. Therefore, on top of the NFDI Gateway, we built a RAG-based scholarly QA system. The RAG retrieval component of the system scans the retrieved resources to identify the top-N most relevant documents based on the user's question. After placing the resources, the scholarly QA uses an LLM (Large Language Model) to extract correct answers to the user's questions directly from the selected documents. By seamlessly integrating the RAG-based scholarly QA with the Gateway, users can efficiently filter through vast amounts of scholarly content, enabling more targeted and productive research efforts.

This approach aims to enhance the user experience, fostering more intuitive and tailored engagement with available information, ultimately contributing to more effective and nuanced research outcomes. Furthermore, a detailed analysis of our experiments is framed as two main research questions (RQs).

- **RQ1:** To what extent does the federated search implemented in the Gateway achieve optimal performance?
- **RQ2:** How does integrating the Scholarly QA on top of the Gateway improve the retrieval of relevant search results?

Our main contributions are twofold:

[1] https://www.nfdi4datascience.de/.

- the NFDI4DS Gateway analysis and completeness of federated search evaluation through information retrieval metrics.
- A scholarly QA system based on RAG on top of the Gateway.

The source code can be found at https://github.com/semantic-systems/nfdi-search-engine-chatbot.

2 Related Works

Data management is a multi-step process that involves obtaining, cleaning, and storing data to allow accurate analysis and produce meaningful results. As an example of this, the Open Research Knowledge Graph [4,24] is an infrastructure for the production, curation, publication, and use of FAIR scientific information with the ultimate goal of providing swift knowledge management within the scientific domain by the digitalization of scholarly articles in the form of the knowledge graph. On the other hand, the federated search [30], as they involve the efficient retrieval of information from multiple data sources, play an essential role in data management as it helps in optimizing the use of data and deriving valuable insights from the data. As shown in [3,11] work, the researchers face a flood of papers that hinders the discovery of necessary knowledge, as a result of this, [11] trained models to identify challenges and directions across the corpus by a dedicated search engine.

Federated search [23] serves as a crucial tool for managing data within scholarly articles, enabling the retrieval of information from diverse sources through a search application constructed on top of one or more data sources [8]. A federated search facilitates information retrieval from multiple scholarly sources, demonstrating remarkable efficacy across various fields, particularly in scientific data management. Shokouhi et al. [23] outlined the challenges inherent in federated search within scholarly domains, delineating three significant hurdles: retrieving relevant documents, identifying suitable collections necessitating knowledge representation and unifying results from multiple sources. Similarly, Kumar et al. [10] dived into how federated search helps libraries and other institutions with a valuable tool to explore various fields and articles. Furthermore, Kirstein et al. [9] introduced *Piveau* as a comprehensive open data management solution grounded in semantic web technologies. Leveraging a spectrum of standards prevalent in the semantic web, such as RDFs and DCAT, this standardization via the semantic web overcomes limitations in search capabilities, ensuring superior quality information retrieval.

The Scholarly QA work in [12] proposes a QA model that extracts question-related full-text scientific articles using an LLM-based retrieval agent and generates answers using RAG techniques. [26] has explored Knowledge Graph QA using an LLM in a few-shot setting for handling bibliographic questions. NLQxform [28] introduces a natural language interface for directly querying the DBLP by automatically translating questions into SPARQL queries. Unlike these Scholarly QAs, we introduce RAG-based Scholarly QA.

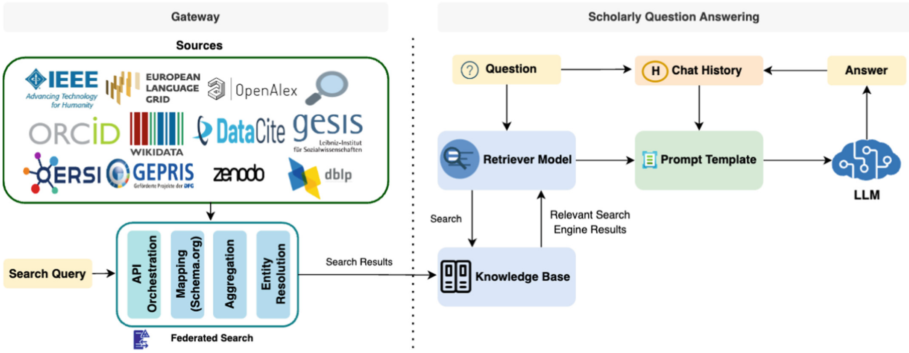

Fig. 1. A functional view of the NFDI4DS Gateway architecture with scholarly QA application.

3 Methodological Framework

The NFID4DS Gateway performs a federated search through various data store APIs. Subsequently, the search results will be indexed into the QA system to allow users to find acquired information via chat. The architectural representation of the Gateway with a scholarly QA system is illustrated in Fig. 1.

3.1 The Gateway – Federated Search

The Gateway conducts federated searches across diverse data stores, generating results that humans can easily interpret. The following key components underpin its functionality:

1. Keyword search across data stores (API Orchestration).
2. Grouping results using a faceted taxonomy (Mapping and Aggregation).
3. Deduplication of results (Entity Resolution).

In the following, we will delve into each of these components in detail, explaining their functions and contributions to the overall functionality of the Gateway.

1) **API Orchestration.** It uses a one-search-box interface to obtain user keywords, and the search results are expressed in a one-result-list-only manner. It subsequently employs federated search using ad-hoc based searches through 11 open-source scholarly repositories, i.e., DBLP[2], OpenAlex[3], CORDIS[4], European Language Grid (ELG)[5], GEPRIS[6], GESIS[7], ORCiD[8], RESO-

[2] https://dblp.org/.
[3] https://openalex.org/.
[4] https://cordis.europa.eu/.
[5] https://live.european-language-grid.eu/.
[6] https://gepris.dfg.de/.
[7] https://www.gesis.org/en/home.
[8] https://orcid.org/.

DATE[9], WIKIDATA[10], IEEE[11], and Zenodo[12]. Among these repositories, DBLP, OpenAlex, IEEE, GESIS, RESODATE, WIKIDATA, and Zenodo provide research resources like publications, datasets, software, etc. GEPRIS provides Deutsche Forschungsgemeinschaft (DFG) funded projects. Likewise, CORDIS is a primary source for projects financed by the European Union (EU) Commission. ELG is a platform that provides multi-lingual, cross-lingual, and mono-lingual language technologies in the EU. Unlike the others, ORCiD delivers a unique, persistent, researcher-owned, and controlled digital identifier that distinguishes researchers uniquely.

2) **Mapping and Aggregation.** The Gateway interacts with various data source APIs, including SPARQL endpoints; the retrieved results often have different structures. For example, while one source refers to an author of a publication as 'author', another refers to them as 'creator', and terminology differences extend to scholarly resources such as datasets, which are referred to as 'corpus' in one source and 'dataset' in another, such as Zenodo. To resolve the naming variations, we have developed a systematic approach based on customized faceted taxonomy from schema.org[13] that harmonizes and aggregates heterogeneous results from API orchestration. This faceted taxonomy acts as a unifying framework that allows us to map the different terminology and structures found in other data sources, thereby coherently facilitating the aggregation and presentation of search results.
The faceted taxonomy based on schema.org is defined to represent different entities found in data sources. These schema.org classes encompass information including organizations, individuals, authors, creative works (articles, datasets, projects, software applications, learning resources, and media objects), and their respective attributes. In particular, the Author and Person classes encapsulate attributes related to individuals who contribute to creative works, while the Organization class encapsulates attributes specific to organizational entities. In addition, the `CreativeWork` super class serves as a foundation for various entities, providing common attributes such as `abstract`, `author`, and `datePublished` inherited by its subclasses. Each class within schema.org contributes to a structured representation of data entities, facilitating organization, interoperability, and standardized data handling within the Gateway.

3) **Entity Resolution.** Following the initial mapping of the publications, researchers, and other resources using the schema.org taxonomy, it becomes necessary to identify and merge duplicate objects within the results. To accomplish this task, we leverage the DEDUPE model [6], which employs machine learning techniques, specifically fuzzy matching, deduplication, and entity resolution, to handle structured data effectively. Later, the DEDUPE

9 https://resodate.org/.
10 https://www.wikidata.org/.
11 https://www.ieee.org/.
12 https://zenodo.org/.
13 https://schema.org/.

model can be fine-tuned on a custom dataset comprising positive and negative samples, thus enabling the model to differentiate between genuine duplicates and distinct entities.

For publication deduplication, the DEDUPE model is trained on a set of attributes, i.e., Digital Object Identifier (DOI), title, author list, abstract, and publication date for publication identification by clustering objects based on similarity scores calculated across attributes. Subsequently, within each cluster, objects that exceed the predefined similarity threshold are merged to form a unified entity, thus resulting in a set of deduplicated records. Later, the resulting records are sorted based on relevancy score using BM25Plus. BM25Plus is a variant of BM25 (Best Match) [20] ranking algorithm, introducing additional term weighting factors to enhance the ranking.

3.2 Scholarly Question Answering

As shown in Fig. 1, our RAG-based [13] scholarly QA has two components: (i) a retriever that returns top-K relevant passage to the user's question and (ii) a generator LLM that generates a human-like response based on a given context from the retriever to a user question.

Retriever. The retrieval model uses a user question as a query to explore relevant information from a knowledge base. The knowledge base comprises a set of documents retrieved per search query through the Gateway. The retriever model operates in three sequential steps:

1. Step 1: The preprocessing knowledge base of search results to obtain a set of documents combined textual data by combining the key-value dictionary per obtained search result.
2. Step 2: The retriever model extracts embeddings for the documents and indexes them within the knowledge base.
3. Step 3: Given a specific question, the retriever model extracts embeddings and computes cosine similarity with the knowledge base, thereby retrieving the top-K appropriate relevant documents to answer the question

We opted for an ensemble retriever model. This ensemble accompanies techniques such as TF-IDF [21], SVM, and KNN retrievers with the Sentence-BERT [19] model serving as the foundational framework. Per the user question, the ensemble retriever queries retriever models to obtain their results; next, it ranks them using each retriever's weights to obtain the final documents most similar to the query. In our retriever collection, the SVM is being trained with the query as a positive class and the rest of the knowledge base documents as negative using sentence-BERT embeddings; next, based on the positive class probability, the documents are ranked and obtain top-k items. By integrating diverse retrieval methodologies, our ensemble model aims to capitalize on the strengths of each component, thereby enhancing overall retrieval performance. Upon experimentation, we manually determined the optimal configuration for our ensemble model.

Based on our observations, we assigned weights of 0.3 to TF-IDF, 0.3 to KNN, and 0.4 to SVM retrievers by try-and-error analysis.

Generator. As shown in Fig. 1, the generator model uses LLM and retriever documents and a prompt template to query LLM to generate a human-like answer to the user questions based on obtained relevant documents from the retriever model. As observed, LLMs showed a great capability for generating human-like responses. However, they might hallucinate and forget the discussion due to the overwhelming information. We provide explicit instructions beside questions and relevant documents, using a predefined prompt template to avoid this. The prompt template enables the scholarly QA to query LLM effectively and answer the user question accurately. The prompt template is described as follows:

> Provide your answers only on the knowledge provided here. Do not use any outside knowledge.
> If you don't know the answer, say that you don't know. Don't try to make up an answer.
> Given the following context, answer the below question:
>
> {context}
>
> Question: {question}
> Helpful Answer:

In the prompt template, {*context*} is the placeholder for retriever model results, and {*question*} is the user question. To account for follow-up questions, we have used conversation buffer memory that keeps track of chat history, consisting of previous questions and answers within five previous conversations. The follow-up questions can reference past chat history, e.g., "What is the open research knowledge graph?" followed by "How to use it?" Such queries challenge direct retriever similarity-based searches, including ensemble retriever models. We provided the chat history for LLM in the prompt template by adding the history questions and answers to the end of retrieval model outputs at {*context*} placeholder. As an LLM, we use GPT-3.5 [16] with the LangChain framework [5] for implementation.

4 Evaluation

4.1 Evaluation Dataset

This section outlines the procedures for constructing the dataset for both the Gateway and scholarly QA evaluations.

Constructing Queries for Assessing the Gateway Performance. The comparison feature of ORKG empowers researchers to construct comprehensive comparisons [25] among scholarly articles spanning diverse domains. A pivotal aspect of this feature is the inclusion of human-generated comparisons. In the evaluation of federated search, we focused on the comparison titles at ORKG, crafted by the researchers themselves. Consider a scenario where a user aims to formulate a comparison for "ontology learning from text" and utilizes the Gateway to gather relevant papers and sources for their study. When a user queries the title on the Gateway, a user can easily use the documents obtained to construct an ORKG comparison for "ontology learning from the text" as shown in https://orkg.org/comparison/R186047. So, comparison titles can be used as a query to study the Gateway's performance in finding relevant documents for researchers.

Through this process, we obtained 1,235 unique comparisons from ORKG as of *February 2nd, 2024*, spanning 161 research fields. Among the obtained research fields, we selected 27 research fields related to AI and data science. Consequently, we identified 316 comparison topics within 27 research fields that fall into the AI and data science category for human annotations to curate titles as a query. Ultimately, we curated a collection of 275 comparison titles for performance analysis of the Gateway and executed queries on the Gateway as of *February 16th, 2024*. The remaining 41 comparison titles we found them inappropriate for querying the Gateway.

Generating Scholarly QA Datasets. We designed a systematic approach to generate well-suited questions tailored to search results. The questions are designed to simulate what questions users ask while using the Gateway. We constructed the AI-QA dataset using GPT-4 [17] and the Comparison-QA dataset using ORKG comparisons. For the AI-QA dataset, we employed k-means [7] clustering methodology on retrieved documents per query, enabling us to efficiently organize the data for generating questions. For search result sets containing more than 50 entries, we applied a clustering number of 10, and for result sets with fewer than 50 entries, a clustering number of 5 was considered appropriate. Search results with less than 5 entries were not included in question generation. Subsequently, we employed GPT-4-Turbo [17] to generate two appropriate questions per cluster using a predefined prompt template. The prompt template is defined as follows:

> The task is to generate questions based on the provided information.
> Given a list of texts, generate only two questions, no more than two.
> Make questions variant.
> The questions should imitate what a user might look for in the given documents.
>
> Return questions as a Python list.
>
> Documents:
> {documents}

This approach proves advantageous in generating questions for scholarly QA evaluation as it relies on documents already recognized for question generation. However, in the evaluation phase, the retriever model gathers search results similar to those of the questions, which the LLM later uses to generate answers. Following the question generation step, we acquired a total of **3,298** questions across 1,651 clusters for scholarly QA evaluations, where we consider each clustering per question as a ground truth.

Since the ORKG comparison is aimed to allow researchers to compare contributions of different articles based on predefined properties such as "research problem" or "model". For Comparision-QA, we used comparison properties as questions using the following standard template:

In the paper "{paper}", what is the {property}?

We considered 275 comparison titles to query the Gateway to obtain federated search results; for the 275 ORKG comparisons comprising 2,395 papers, only 184 were retrieved by the Gateway. So, we used 184 papers and their properties to construct questions, and values for the property per paper in the comparison were considered as answers. In the end, a total of 1,354 questions were constructed.

The overview of the datasets is presented in Table 1.

Table 1. Statistics for the number of search queries (Query), number of comparison papers (Comparison Papers), number of papers from ORKG comparison that are being covered in search results (ORKG Coverage), and comparison specific questions (Comparison-QA).

Query	AI-QA	Comparison Papers	ORKG Coverage	Comparison-QA
275	3,298	2,303	184	1,354

4.2 Evaluation Metrics

Gateway Evaluation Metrics. In evaluating the performance of the Gateway, we employed multiple approaches, primarily focusing on response time, number of retrieved documents, and relevancy scores. The response time analysis serves as a critical metric in assessing the efficiency and responsiveness of the Gateway. Another key aspect of our evaluation involved analyzing the number of documents retrieved by the Gateway in response to user queries. This metric provides valuable information about the comprehensiveness and effectiveness of the search results generated by the system. To further refine our evaluation, we calculated relevancy scores per retrieved document similarity to the search query based on varying thresholds and representations such as sentence-BERT, TF-IDF, and BM25 [1]. With sentence-BERT sentence embeddings, TF-IDF, and BM25 scores, we calculated cosine similarity between documents and queries for all metrics to get relevancy scores.

Scholarly QA Evaluation Metrics. In AI-QA, we utilized question clusters as answers, while in comparison-QA, property values were employed as answers. Subsequently, we assessed performance using n-gram overlap specific metrics like ROUGE [14] (Recall-Oriented Understudy for Gisting Evaluation) and BLEU [18] (Bilingual Evaluation Understudy), focusing specifically on ROUGE-1, ROUGE-L, and BLEU-1 as our evaluation criteria. Because LLMs generate responses based on their comprehension, they might deviate from the ground truth text, making evaluation with metrics like ROUGE and BLEU difficult. Consequently, incorporating similarity scores into the assessment process can offer further insights into their proficiency in capturing subtle language nuances. We used the BERTScore – a sentence-BERT average cosine similarity metric as an evaluation. Furthermore, as the Comparison-QA dataset poses challenges with answers often appearing within the paper context rather than solely in abstracts and titles, we opted for the Exact Match score as another evaluation metric only for this dataset.

4.3 Results

Gateway and Scholarly QA Results. The performance of the Gateway has been assessed by considering factors such as its response time, the number of documents retrieved, and the relevance of those documents. The Gateway performances are reported in Fig. 2 and Fig. 3. The results for scholarly QA evaluation, employing various metrics, are reported in Table 2. We identified 432 questions without answers for AI-QA, while we obtained 26 questions without answers for Comparison-QA. This happened due to the input limitation of GPT-3.5. Hence, we excluded these questions from evaluations.

Table 2. Evaluation results of the scholarly QA using AI-QA and Comparison-QA datasets, showcasing ROUGE, BLEU, BERTScore, and Exact Match scores for the RAG-based scholarly QA development.

Dataset	ROUGE-1	ROUGE-L	BLEU-1	BERTScore	Exact Match
AI-QA	4.21	2.92	38.94	36.81	–
Comparison-QA	6.82	6.10	3.10	26.96	13.93

RQ1: [Gateway] To what extent does the federated search implemented in NFDI4DS achieve optimal performance? We address this question by analyzing the findings presented in Fig. 2 and Fig. 3. Ultimately, for a search platform, it is essential to retrieve relevant results while maintaining a fast response time across various queries. The analysis of response time and retrieved documents status in Fig. 2 for 275 search queries showed that the federated search is capable of obtaining **123** documents on average within an average response time of **4.93** seconds. Notably, slow performance is observed in the

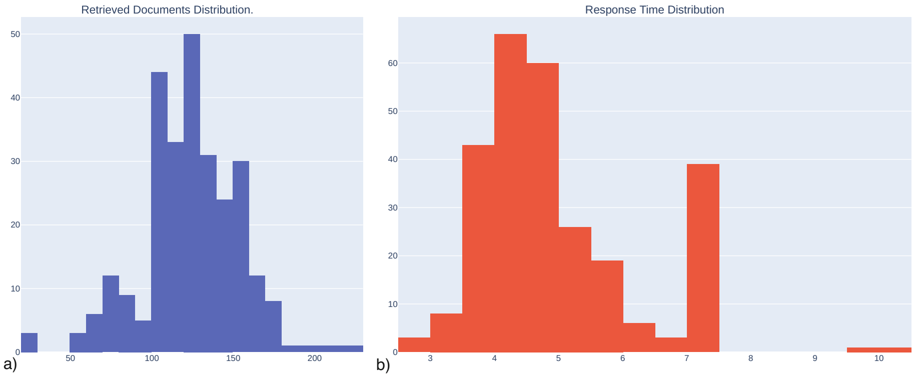

Fig. 2. Gateway retrieved documents distribution is presented in the left figure. The x-axis represents the number of retrieved documents, and the y-axis the number of queries. The right figure represents the response time distribution, with the x-axis as a response time in seconds and the y-axis as the number of queries.

search query of the "Kinect human activity recognition dataset" with approximately 10 s response time and search results of 169 documents. Similarly, for the "Motion Capture system" search query, we obtained 227 documents within 4.3 s. This shows that depending on different search keywords and how complex the query is, it may result in sacrificing response time. In general, according to Fig. 2, the distribution analysis indicated that the number of retrieved documents follows a *normal* distribution, while the distribution of response time is *positively skewed*. This highlights the significant performance of the Gateway in terms of response time and document retrieval.

We calculated cosine similarities with three metrics to analyze the retrieved documents' relevancy. We set relevancy thresholds to see how many queries with their corresponding documents are considered very relevant to each other. The relationship between the relevancy threshold and the number of retrieved documents is depicted in Fig. 3, indicating a decrease as the threshold increases. The TF-IDF metric generates the highest similarity scores between documents and queries, albeit focusing primarily on token frequency rather than semantic understanding. BM25, an improvement upon TF-IDF, proves particularly effective for information retrieval tasks, displaying a different score distribution with numerous low similarity scores. Despite this, BM25 still identifies certain documents as highly relevant (with similarity above 0.3) for specific queries. Conversely, sentence-BERT initially achieves the highest average recall but drops to zero at a threshold of 0.8. Comparatively, BM25 and sentence-BERT yield similar results, implying that capturing nuanced semantics may not be crucial for retrieving relevant articles; instead, identifying standard terms and phrases appears more pivotal. Evaluating the optimal threshold of 0.3, TF-IDF emerges as the optimal ranking model. The overall relevancy analysis across different thresholds indicates that the Gateway effectively retrieves search results based

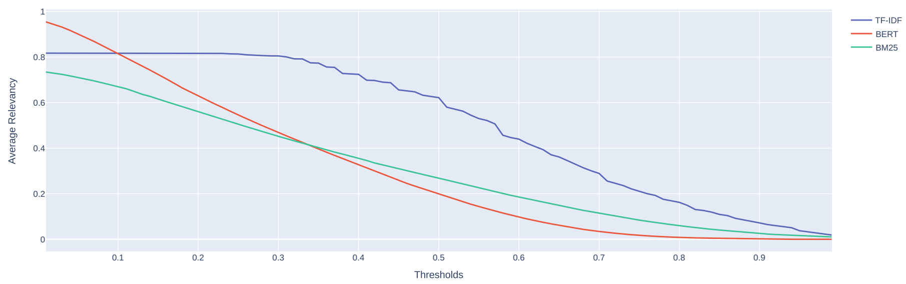

Fig. 3. Gateway retrieved documents relevancy w.r.t search query analysis using TF-IDF, BM25, and sentence-BERT embeddings for similarity measurement and different thresholds in the range of [0.0, 0.99]

on keyword search but struggles with semantic retrieval. However, setting the threshold to 0.3 demonstrates approximately 50% semantic similarity among documents, highlighting the Gateway's proficiency in identifying relevant documents from keyword and semantic perspectives.

RQ2: [Scholarly QA] How does integrating the Scholarly QA on top of the Gateway improve the retrieval of relevant search results? We address this question by analyzing the results presented in Table 2 for both automated constructed Comparison-QA and AI-QA datasets. According to the ROUGE-1 metric, unigrams overlap between the developed QA-generated responses and existing answers. This overlap is more significant for Comparison-QA (6.82%) than for AI-QA (4.21%). Similarly, when considering ROUGE-L, which measures the Longest Common Subsequence, the overlap for Comparison-QA (6.10%) surpasses that of AI-QA (2.92%). However, despite the QA's promising BLEU-1 score of 38.94% on the AI-QA dataset, its performance on the Comparison-QA dataset is lacking. This suggests that the developed QA responses align more closely with the clustered documents, which are the ground truth in our AI-QA dataset.

It is essential to note that both the ROUGE and BLEU metrics have limitations when applied to LLM-based generations. This is because LLM-generated responses may exhibit variations that mimic human-like responses, making it challenging for these metrics to evaluate their quality accurately. Still, they show how much of the generated text is similar to ground truth. Nevertheless, we reported a BERTScore of 36.81% for the AI-QA dataset and 26.96% for the Comparison-QA dataset. These obtained BERTScore results suggest that the quality of the scholarly QA's responses, particularly in terms of semantic similarity to ground truth references, varies significantly between the two datasets. As mentioned earlier, the variation between the two datasets was expected since the Comparison-QA mostly extracted humans from the whole body of the paper rather than only the title and abstract.

We computed the exact match for Comparison-QA, revealing a 13.93% match between the ground truth and the QA-generated text. This highlights the scholarly QA's proficiency in recognizing relevant information, mainly when it appears in the search results. To the best of our knowledge, there is no other baseline system or scholarly QA system available to which we can compare.

5 Limitations and Future Directions

This section discusses the limitations encountered in the implementation of the scholarly QA model and outlines potential future directions for addressing these shortcomings.

Inadequate Availability of Comparison-QA Dataset Answers. The scholarly QA's performance is hindered by the frequent unavailability of answers to the Comparison-QA answers in search results, resulting in suboptimal performance. Addressing this limitation requires an extensive collection of queries from ORKG comparisons. Another limitation arises from the lack of diversity in the questions, as the current methodology employs a single template for forming questions on this dataset.

Suboptimal AI-QA Dataset Generation. The AI-QA dataset, generated from clustered search results, sometimes yields many documents per cluster. Thus, an optimal clustering method is necessary to manage the data effectively. Additionally, soliciting human feedback on the generated questions is crucial for refining and enhancing the dataset's quality. In future works, it is helpful to have a small human-generated dataset to justify the evaluation's validity further.

Exploring Diverse LLMs. Future research should focus on exploring a more comprehensive range of LLMs within scholarly QA to study their diversity and identify more optimal models for scholarly documents. This endeavor necessitates dataset curation tailored explicitly to the Gateway results.

6 Conclusion

In this work, we present an interactive scholarly QA system based on the RAG approach on top of the NFDI4DataScience Gateway search results, facilitating user interaction with a wealth of data. Subsequently, we automatically evaluated both the Gateway and scholarly QA using an automatically constructed dataset. The analysis indicates that as early prototypes, both the Gateway and QA show satisfactory performance. However, there is a need for future work to stabilize both systems and harness data science expertise.

Acknowledgments. This work was jointly supported by the German Research Foundation (DFG) project NFDI4DS under Grant no.: 460234259 and the German BMBF project SCINEXT under grant no. 01lS22070.

References

1. Amati, G.: BM25. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems, pp. 257–260. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-39940-9_921
2. Auer, S., et al.: The SciQA Scientific Question Answering Benchmark for Scholarly Knowledge. Sci. Rep. **13**(1), 7240 (2023). https://www.nature.com/articles/s41598-023-33607-z
3. Auer, S., Kovtun, V., Prinz, M., Kasprzik, A., Stocker, M., Vidal, M.: Towards a knowledge graph for science. In: Akerkar, R., et al. (eds.) Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, WIMS 2018, Novi Sad, Serbia, 25–27 June 2018, pp. 1:1–1:6. ACM (2018). https://doi.org/10.1145/3227609.3227689
4. Auer, S., et al.: Improving access to scientific literature with knowledge graphs. Bibliothek Forschung ud Praxis **44**(3), 516–529 (2020). https://doi.org/10.1515/bfp-2020-2042
5. Chase, H.: LangChain (2022). https://github.com/langchain-ai/langchain
6. Gregg, F., Eder, D.: dedupe (2022). https://github.com/dedupeio/dedupe
7. Jin, X., Han, J.: K-means clustering. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 563–564. Springer, Cham (2010). https://doi.org/10.1007/978-0-387-30164-8_425
8. Kairouz, P., et al.: Advances and open problems in federated learning. Found. Trends Mach. Learn. **14**(1–2), 1–210 (2021). https://doi.org/10.1561/2200000083
9. Kirstein, F., Stefanidis, K., Dittwald, B., Dutkowski, S., Urbanek, S., Hauswirth, M.: Piveau: a large-scale open data management platform based on semantic web technologies. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 648–664. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_38
10. Kumar, S., Sanaman, G., Ra, N.: Federated search: New option for libraries in the digital era (2007)
11. Lahav, D., et al.: A search engine for discovery of scientific challenges and directions. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, 22 February–1 March 2022, pp. 11982–11990. AAAI Press (2022). https://doi.org/10.1609/aaai.v36i11.21456
12. Lála, J., O'Donoghue, O., Shtedritski, A., Cox, S., Rodriques, S.G., White, A.D.: PaperQA: retrieval-augmented generative agent for scientific research. CoRR abs/2312.07559 (2023). https://doi.org/10.48550/ARXIV.2312.07559
13. Lewis, P.S.H., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, 6–12 December 2020, Virtual (2020). https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html

14. Lin, C.Y.: ROUGE: a package for automatic evaluation of summaries. In: Text Summarization Branches Out, pp. 74–81. Association for Computational Linguistics, Barcelona (2004). https://aclanthology.org/W04-1013
15. Mahdi, M., Ahmad, A., Ismail, R., Kadhim, H., Mohammed, M.: Solution for information overload using faceted search - a review. IEEE Access **8**, 1 (2020). https://doi.org/10.1109/ACCESS.2020.3005536
16. OpenAI: ChatGPT (2023). https://openai.com/chat-gpt/. Accessed 5 May 2023
17. OpenAI: GPT-4 technical report. CoRR abs/2303.08774 (2023). https://doi.org/10.48550/ARXIV.2303.08774
18. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: a method for automatic evaluation of machine translation. In: Isabelle, P., Charniak, E., Lin, D. (eds.) Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pp. 311–318. Association for Computational Linguistics, Philadelphia (2002). https://doi.org/10.3115/1073083.1073135
19. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019, pp. 3980–3990. Association for Computational Linguistics (2019). https://doi.org/10.18653/V1/D19-1410
20. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: BM25 and beyond. Found. Trends Inf. Retr. **3**(4), 333–389 (2009). https://doi.org/10.1561/1500000019
21. Sammut, C., Webb, G.I.: TF-IDF. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 986–987. Springer, Cham (2010). https://doi.org/10.1007/978-0-387-30164-8_832
22. Schimmler, S., et al.: NFDI4DS infrastructure and services. In: Klein, M., Krupka, D., Winter, C., Wohlgemuth, V. (eds.) 53. Jahrestagung der Gesellschaft für Informatik, INFORMATIK 2023, Designing Future - Zukünfte gestalten, Berlin, Germany September 26-29, 2023. LNI, vol. P-337, pp. 919–924. Gesellschaft für Informatik, Bonn (2023). https://doi.org/10.18420/INF2023_103
23. Shokouhi, M., Si, L.: Federated search. Found. Trends Inf. Retr. **5**(1), 1–102 (2011). https://doi.org/10.1561/1500000010
24. Stocker, M., et al.: Fair scientific information with the open research knowledge graph. FAIR Connect **1**, 19–21 (2023). https://doi.org/10.3233/FC-221513
25. Stocker, M., et al.: Fair scientific information with the open research knowledge graph. FAIR Connect **1**(1), 19–21 (2023). https://doi.org/10.3233/FC-221513
26. Taffa, T.A., Usbeck, R.: Leveraging LLMs in scholarly knowledge graph question answering. In: Banerjee, D., Usbeck, R., Mihindukulasooriya, N., Singh, G., Mutharaju, R., Kapanipathi, P. (eds.) Joint Proceedings of Scholarly QALD 2023 and SemREC 2023 Co-located with 22nd International Semantic Web Conference ISWC 2023, Athens, Greece, 6–10 November 2023. CEUR Workshop Proceedings, vol. 3592. CEUR-WS.org (2023). https://ceur-ws.org/Vol-3592/paper5.pdf
27. Usbeck, R., et al.: NFDI4DS gateway and portal. In: Sure-Vetter, Y., Goble, C.A. (eds.) 1st Conference on Research Data Infrastructure - Connecting Communities, CoRDI 2023, Karlsruhe, Germany, 12–14 September 2023. TIB Open Publishing (2023). https://doi.org/10.52825/cordi.v1i.391
28. Wang, R., Zhang, Z., Rossetto, L., Ruosch, F., Bernstein, A.: NLQxform: a language model-based question to SPARQL transformer. CoRR abs/2311.07588 (2023). https://doi.org/10.48550/ARXIV.2311.07588

29. Wilkinson, M., et al.: The fair guiding principles for scientific data management and stewardship. Sci. Data **3**, 160018 (2016). https://doi.org/10.1038/sdata.2016.18. Erratum in: Sci. Data **6**(1), 6 (2019)
30. Yang, L., Tan, B., Zheng, V.W., Chen, K., Yang, Q.: Federated recommendation systems. In: Yang, Q., Fan, L., Yu, H. (eds.) Federated Learning. LNCS (LNAI), vol. 12500, pp. 225–239. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63076-8_16

Cite-worthiness Detection on Social Media: A Preliminary Study

Salim Hafid[1(✉)], Wassim Ammar[1], Sandra Bringay[1,2], and Konstantin Todorov[1]

[1] LIRMM, CNRS, University of Montpellier, Montpellier, France
{salim.hafid,wassim.ammar,sandra.bringay,konstantin.todorov}@lirmm.fr
[2] University Paul Valéry, Montpellier, France

Abstract. Detecting cite-worthiness in text is seen as the problem of flagging a missing reference to a scientific result (an article or a dataset) that should come to support a claim formulated in the text. Previous work has taken interest in this problem in the context of scientific literature, motivated by the need to allow for reference recommendation for researchers and flag missing citations in scientific work. In this preliminary study, we extend this idea towards the context of social media. As scientific claims are often made to support various arguments in societal debates on the Web, it is crucial to flag non-referenced or unsupported claims that relate to science, as this promises to contribute to improving the quality of the debates online. We experiment with baseline models, initially tested on scientific literature, by applying them on the SciTweets dataset which gathers science-related claims from X. We show that models trained on scientific papers struggle to detect cite-worthy text from X, we discuss implications of such results and argue for the necessity to train models on social media corpora for satisfactory flagging of missing references on social media. We make our data publicly available to encourage further research on cite-worthiness detection on social media.

Keywords: Cite-worthiness · Science-related discourse · Social Media · NLP

1 Introduction

Social media, especially X (ex-Twitter), has become a vital platform for scientific discourse among scholars, but also among non-academic users. Scientists rely on X as a convenient platform for sharing findings and connecting with peers [30], while non-scientific users often call upon scientific results or formulate science-related claims in order to give more weight to their arguments in societal debates on various, often controversial topics. For example, discussions surrounding the recent COVID-19 global pandemics were often fueled by science-related arguments—verified or not—relating to vaccines efficiency or protection measures. While a lot of attention has been given to analysing science-related

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 19–30, 2024.
https://doi.org/10.1007/978-3-031-65794-8_2

claims from scientific literature [23], only recently the natural language processing (NLP) community started taking interest in scientific discourse on social media and on the Web at large [22]. These recent efforts have been largely motivated by the observation that scientific discourse is arguably different on social media as compared to academic literature, where social media users leaning on science in their discourse would often lack rigour, oversimplify or mis-contextualize scientific findings [21].

A specific problem in that context is that of cite-worthiness detection, seen as the task of *"identifying citing sentences, i.e., sentences which contain a reference to an external source"* in text [1]. In particular, this task can be useful for flagging a missing reference to a scientific result (an article or a dataset) that should come to support a claim formulated in the text, hence giving credit to the original author, giving credibility to the claim presented or providing additional insights. Previous work has taken interest in this problem in the specific context of scientific literature, motivated by the need to allow for reference recommendation for researchers and to flag missing citations in scientific work [1]. In our work, we extend this idea towards the context of social media, leveraging the results and models reported in [1]. While scientific claims are often made to support various arguments in societal debates on the social Web, the lack of citation standards, as compared to academic writing, leads to the presence of largely unsupported science-related claims and mis-contextualized scientific findings, which in turn leads to a poor quality of the debates online, lacking transparency, credibility and accuracy, with potentially harmful effects on democratic discourse [15–17].

In [1], several well-known pre-trained language models, such as SciBERT and Longformers are fine-tuned for the specific task of cite-worthiness detection in scientific literature and evaluated against a simple logistic regression baseline, by relying on data tailored for the task.[1] In our preliminary study, we follow the protocol provided by [1], by applying and fine-tuning the same models and baselines, but in contrast using data coming from X exclusively. Namely, we rely on the SciTweets dataset [3],[2] which gathers human annotated science-related claims from X, based on the definition of scientific web claims and the annotation protocol given in [3]. We further preprocess and filter tweets from SciTweets to map them to the cite-worthiness definition from [1]. We observe consistent decline across all metrics when evaluating models on X data. This hints that the inherent difference between academic and social media scientific discourse [13,14,18] translates to a degraded performance of baseline models on the downstream task of cite-worthiness detection, calling for specific models that are capable of taking into consideration the specificity of scientific discourse on the Web.

In this work, we contribute:

[1] https://github.com/copenlu/cite-worth.
[2] https://github.com/AI-4-Sci/SciTweets.

1. SCiteTweets, the first publicly available dataset for cite-worthiness detection on social media, consisting in 415 tweets constructed by preprocessing and filtering tweets from the SciTweets dataset.[3]
2. The first empirical evaluation of cite-worthiness detection on social media, where we observe that performance of models trained on scientific publications consistently declines when evaluated on data from X.

2 Related Work

The notion of cite-worthiness relates to the notion of check-worthiness, which has been extensively researched by fact-checking related studies over the years.[4] A sentence is defined as "check-worthy" if it is worth fact-checking (e.g., contains a verifiable factual claim, is potentially harmful, and is of general interest) [28,29], whereas a sentence is "cite-worthy" if it contains a reference to an external source [1]. While check-worthiness detection can help professional fact-checkers detect which claims to focus on, cite-worthiness detection can be used to flag scientific results which are presented without references.

Determining whether a (scientific) text lacks and hence requires a citation, has been one of the challenges in the NLP community. The larger group of approaches has tackled this problem in the context of scientific publishing, using corpora constructed from academic articles in specific fields. For example, [24] use Support Vector Machines on a dataset created from the ACL Anthology Reference corpus [25], while more advanced approaches [6] measure the performance of a Convolutional Recurrent Neural Network on the ACL Arc dataset[5] as well as arXivCS [26] and Scholary Dataset.[6] The limitations of these works are mainly related to domain-specificity, class imbalances, and little to no presence of data quality analysis. These issues were addressed in [1], where the authors build and share a curated multi-domain dataset specifically dedicated to the task of cite-worthiness detection, that is used to evaluate a number of language models against a logistic regression baseline.

On the social media side, existing work [7] observed that the nature of X has led to a more lenient way of citing, especially in the scientific field where discourse is expected to be more formal. The larger amount of work analysing X data and scientific discourse is generally about the lack of trust in the shared content [8], more precisely focusing on fact-checking. For example, in [9] the authors create a manually annotated dataset to identify claims as check-worthy, while in [10] the authors leverage Large Language Models to build datasets for identifying misinformation.

[3] The data is made publicly available at https://github.com/SalimHFX/SCiteTweets/.

[4] See the CheckThat! Lab editions hosted by the CLEF conferencehttps://checkthat.gitlab.io/clef2024/task1/.

[5] https://paperswithcode.com/dataset/acl-arc-1.

[6] https://www.db.soc.i.kyoto-u.ac.jp/~sugiyama/Dataset2.html.

Studying scientific citation in social media is a relatively novel task. In [11], the authors suggest that tweets can predict the citation of papers in the biomedical field, concluding that X citations may be an alternative to traditional ones on the impact of research findings. Supporting that work, [12] assembles a dataset relating tweets and citations of arXiv papers. Finally, [3] presents a definition of scientific web claims and provide a curated dataset of tweets annotated according to that definition. This dataset, although limited in size, provides hints about citation tendencies in scientific discourse on X.

In an attempt to provide preliminary insights into this under-researched problem, we build on the work of [1] by reproducing their experiments on X-provenance data using the SciTweets dataset from [3] in order to highlight the shortcomings of state-of-the-art pre-trained models when taken out of the academic literature context, which in turn hints to the inherent difference of discourse on social media as compared to scientific papers.

3 Data

To evaluate cite-worthiness performance on social media, we use the following two distinct datasets (examples from each dataset are shown in Table 1):

- **CiteWorth** [1]**:** To our best knowledge, CiteWorth is the largest dataset dedicated to cite-worthiness detection from scientific-publication text. It is extracted from the S2ORC dataset [5] which consists of 81.1M english-language scientific publications. It is then filtered, where sentences are given "cite-worthy" labels indicating that they originally contained a citation at the end of the sentence. The final dataset consists of 1.1M sentences, where over 375k sentences are labeled as cite-worthy.
- **SciTweets** [3]**:** SciTweets is a dataset dedicated to online scientific discourse, where authors developed a hierarchical definition of science-relatedness and curated ground-truth data from X. Tweets are categorized into different categories of science-relatedness depending on whether they contain scientific knowledge, a reference to scientific knowledge, or are related to scientific research in general. The final dataset consists of 1,261 human-annotated tweets. We use the SciTweets dataset to construct **SCiteTweets**, our dataset for cite-worthiness detection on X, by mapping SciTweets labels to cite-worthiness labels. We explain this procedure in detail in Sect. 4.1.

Table 1. Samples from the existing labels in both datasets used in our experiments

	Size	Labels	Examples
CiteWorth	1,181,793	Cite-worthy	The success rate of PNA in the literature varies from 79–100%
		Non Cite-worthy	We compared visual electrophysiology recording of patients with the normal range as defined in our laboratory
SciTweets	1,261	Scientific knowledge	also cancer is virtually incurable bc all cancers are different :)
		Reference to scientific knowledge	Modeling precision treatment of breast cancer looks great ! http://t.co/4XzfGlwAWn
		Related to scientific knowledge	Lupus Research Institute Awards $1-Million Grants to Discover What Causes Lupus http://t.co/aXopNmLyI7
		Non science-related	These birds won't stop cherping!

4 Experiments

4.1 Setting

To evaluate the performance of existing cite-worthiness detection models on a social media corpus, we run multiple experiments to achieve the following goals: (1) reproducing the results found by authors of the CiteWorth dataset [1], (2) applying those models on the SciTweets dataset [3] to evaluate the performance of existing cite-worthiness detection models on a social media corpus, where we experiment with training models on the CiteWorth dataset and on the SciTweets datasets. To reproduce results from the CiteWorth dataset [1], we pick the following three models which all have been previously used by the authors: a logistic regression model, which represents the simplest explainable baseline, a SciBERT model [2] which had the best precision score in the authors' experiments, and a Longformer model [4] which achieved the best F1 score in the authors' experiments. While in their experiments authors used two distinct versions of Longformer, Longformer-Ctx where they use sequence modeling to embed entire paragraphs, and Longformer-Solo, where they embed single sentences, in this paper we opted to use Longformer-Solo (embedding single sentences only), as it best fits the tweets' inherently short format.

Prior to conducting the experiments, we needed to further preprocess the SciTweets dataset in order to ensure a correct mapping between its labels and the cite-worthiness labels from CiteWorth. While CiteWorth contains sentence texts and labels pointing to whether the text is cite-worthy or not, SciTweets' texts are multi-labeled. The first step was to select a label from SciTweets which can be qualified as equivalent to the cite-worthiness label from CiteWorth. The structure of the SciTweets multi-labeled dataset is as follows: a tweet is either science-related or not, if it is science-related, then the tweet is further categorized as belonging to one or more of the following subcategories: *"cat. 1: containing a scientific claim"*, *"cat. 2: containing a reference to scientific knowledge"*, or *"cat. 3: related to scientific research in general"* [3]. The first two categories are good candidates, as they can both contain cite-worthy text. However category 2 is the most suited since it references an external source of scientific nature, much like how authors constructed the CiteWorth dataset, where they focused on sentences that have an indication of a citation which is in essence an external scientific reference. Furthermore, we selected the remaining science-related tweets (categories 1 and 3) as our negative class. By doing so, we ensure that both our positive and negative classes contain science-related text, and that the classes only differ in cite-worthiness, thus matching the CiteWorth setup. The implications of this choice will be discussed further in Sect. 5.

Moreover, we also preprocessed the tweets to match the CiteWorth setup, where we removed user-handles and URLs from cite-worthy tweets. We also removed "citation markers" at the end of sentences, as defined by authors of CiteWorth, where a citation marker is *"any text that trivially indicates a citation, such as the phrase "is shown in""*. Authors argue that removing such citation markers prevents models from learning and using these signals for prediction.

To do so, we removed excess punctuation and hanging words (*"by"*, *"via"*). This step was possible due to how the Category 2 of SciTweets [3] was constructed, where the URLs direct to actual scientific articles. We call the resulting dataset SCiteTweets, which contains tweets extracted from SciTweets that were preprocessed and filtered as described above to match the cite-worthiness definition from [1]. We show statistics of SCiteTweets in Table 2, and examples of cite-worthy and non cite-worthy sentences from both datasets in Table 3.

Table 2. Data used for the experiments

Labels	CiteWorth	SCiteTweets
Cite-worthy	375,388	207
Non Cite-worthy	806,405	208

Table 3. Examples of cite-worthy and non cite-worthy sentences from scientific papers (CiteWorth) and from tweets (SCiteTweets)

	Cite-Worthy	Non Cite-Worthy
CiteWorth [1]	The known forms of terrestrial life involve carbon-based chemistry in liquid water	We compared visual electrophysiology recording of patients with the normal range as defined in our laboratory
SCiteTweets	Hopes raised for cancer treatment after experiments halted tumour growth in mice	proper preparation prevents poor performance

We run *three distinct experiments*, **(1)** training and evaluating on the CiteWorth dataset. This experiment is a direct reproduction of results from CiteWorth authors [1]; **(2)** training on CiteWorth and evaluating on SCiteTweets. This experiment enables us to evaluate whether models trained on a large amount of cite-worthy sentences extracted from scientific publications translates to a good performance on cite-worthy sentences from social media; **(3)** training and evaluating on SCiteTweets. This experiment enables us to evaluate whether models trained on a small amount of social media data translates to a good performance on cite-worthy sentences from social media. For each experiment, we use three distinct base-models (Logistic Regression, SciBERT, and Longformer), thus amounting to nine experiments in total. We then evaluate using Precision (P), Recall (R), and F1-score (F1) for each experiment. For all models we reproduce the experimental setting of the CiteWorth paper [1], for transformer-based models we train models on 3 epochs and follow authors' settings for all hyperparameter values such as batch size, learning rate and dropout probability. For the Logistic Regression model we use a C value of 0.11 following authors. Since the amount of social media data we have is limited (See Table 2), we run a 10-fold cross-validation of SCiteTweets data for experiments (2) and (3). For experiment (2), the training set is the same in all folds (model is trained on CiteWorth) and only the evaluation set changes in each fold. For experiment (3), both training and evaluation sets change in each fold. We follow the same train-test split size as CiteWorth in each fold. The same seed is used for cross-validating experiments (2) and (3), thus ensuring that models are evaluated on the same test sets between the two experiments.

Table 4. Experimental results. For each model, three experiments were run, corresponding to experiments (1), (2) and (3) as described in Sect. 4.1

Models	Experiments		Metrics		
	Trained	Evaluated	P	R	F1
Logistic Regression	CiteWorth	CiteWorth	46.65	64.85	54.26
	CiteWorth	SCiteTweets	49.38	47.43	**48.83**
	SCiteTweets	SCiteTweets	56.11	57.83	**56.95**
SciBERT	CiteWorth	CiteWorth	65.60	52.08	58.06
	CiteWorth	SCiteTweets	53.29	23.90	**32.99**
	SCiteTweets	SCiteTweets	76.91	70.24	**73.42**
Longformer	CiteWorth	CiteWorth	56.85	68.03	61.94
	CiteWorth	SCiteTweets	54.34	23.89	**33.18**
	SCiteTweets	SCiteTweets	34.26	19.91	**25.18**

4.2 Results

We show the results of all experiments in Table 4. For experiments **(2)** and **(3)**, the presented scores are averages across 10 folds. The results of experiment **(1)** (reproducing CiteWorth results) were satisfactory, as they closely mirrored the findings outlined in the CiteWorth paper [1]. The results of experiment **(2)** (training on CiteWorth and evaluating on SCiteTweets) show a consistent decline in F1-points across all three models (LR, SciBERT, Longformer) compared to experiment **(1)**. For the baseline LR model, the decrease is of roughly 5 F1 points. For the SciBERT model, the decrease is more pronounced, with the Recall and F1 score halving compared to experiment **(1)**, recording a decrease of over 25 F1 points. And for the Longformer model, the decrease is even more noticeable, where the model loses close to 30 F1 points when evaluated on tweets compared to its performance on scientific articles.

Finally, the results of experiment **(3)** (training and evaluating on SCiteTweets) showed that most models performed best on tweets when trained on tweets. More specifically, models perform better on tweets when trained even on a small amount of tweets (experiment **(3)**), than when trained on a large amount of scientific papers (experiment **(2)**). Moreover, Longformer, the best performing model from experiment **(1)**, i.e., the best performing model on scientific papers, is the worst performing model on tweets, having even worse scores than experiment **2**. Finally, the SciBert model outperformed both LR and Longformer on the tweets dataset.

4.3 Discussion

First, we attribute the performance of the Longformer model on experiment **(3)** to the small data size of the tweets data. We hypothesize that further experiments on a larger scale social media dataset would result in the Longformer model performing best on tweets when trained on tweets, as observed for the SciBERT and LR models (data size limitations are discussed in Sect. 5). Secondly, the consistent decline in F1-points across all three models (LR, SciBERT, Longformer) when training on CiteWorth and evaluating on SCiteTweets (compared to training and evaluating on CiteWorth) may be explained by differences in the linguistic structure of scientific text on the Web, where science, as discussed on the Web, differs in language from traditional scientific text from scientific papers. Existing literature has shown that scientific text on the Web uses a specialized language [13,14], while communication studies have shown that scientific knowledge online is often sensationalized, lacks perspective [18], and has a tendency to favor conflict [19]. To verify this in our data, we show word clouds of cite-worthy text from both tweets and scientific papers in Fig. 1. Cite-worthy sentences from scientific papers show a high usage of terms such as *"may"*, *"however"*, which might indicate a more careful contextualized phrasing of scientific results and of the scope in which they are valid. In contrast, cite-worthy sentences from tweets do not show usage of such terms, which might indicate a more straightforward and possibly decontextualized phrasing of scientific findings on social media. We leave for future research a more thorough analysis of linguistic differences between scientific papers text and social media text with regards to cite-worthiness.

The conclusions from the experiments in this preliminary study are that transformer-based models fine-tuned on sentences from scientific papers do not perform satisfactory on tweets for the task of cite-worthiness detection, making it difficult to correctly identify cite-worthy and check-worthy tweets, a step which has been stated by professional fact-checkers in a survey [20] as one of the main challenges and the most useful tasks to automate. In future work, we want to investigate the usefulness of training transformer-based models on larger social media corpora, with the goal of enhancing citation detection performance on social media.

5 Limitations

One limitation of our study is the size of our tweets dataset (SCiteTweets, extracted from SciTweets [3]). While the experimental results do underline a

Fig. 1. Word clouds for cite-worthy sentences from tweets (left) and from scientific papers (right)

clear trend (i.e., that models trained on scientific papers underperform on scientific text from X), our results have to be cemented by further experiments on larger scale datasets. However, to our best knowledge, the SciTweets dataset we used is the only currently existing dataset whose labels can be mapped to a cite-worthiness detection task. Another limitation is the mapping from SciTweets labels to CiteWorth labels, where we used tweets which contain a *reference to scientific knowledge* in order to match the cite-worthiness definition from the CiteWorth authors [1]. With this definition, we hope to flag tweets where users missed including references but nonetheless used language showing that there is such a reference, e.g., the following tweet with no actual reference URL: *"I read a recent study which shows that vaccines are not efficient"*. This use-case is already useful and necessary, as recent literature [27] showed that these so-called informal references are prominent on X and are shared and engaged with on social media twice as much as the actual research articles they implicitly refer to. However, ultimately, we also want to be able to flag tweets where the reference is missing and where users never meant to include it, e.g., the following tweet: *"Vaccines are not efficient"*. We consider the use-case discussed in this paper as a necessary first step towards flagging missing references on social media, and we leave the second use-case to future work.

6 Conclusion

This paper addresses the problem of detecting cite-worthiness in text, seen as the task of flagging a missing reference to a scientific article that should come to support a claim formulated in the text. While previous work has mainly taken interest in this problem from a scientific literature perspective, in our study, we extend this idea to the social media context. The paper lays ground for a discussion as of how flagging missing scientific references in claims made on social media can help improve the quality of societal debates and increase trust in social media platforms. Our preliminary results show that state of the art models applied on scientific literature corpora perform less well when let to deal with claims coming from X/Twitter. This observation opens the way for research into the development of (a) larger annotated datasets for cite-worthiness detection on claims from X and (b) the development of language models tailored for scientific discourse that could be fine-tuned for that specific task.

Acknowledgements. This work is supported by the AI4Sci grant, co-funded by MESRI (France, grant UM-211745), BMBF (Germany, grant 01IS21086), and the French National Research Agency (ANR).

References

1. Wright, D., Augenstein, I.: CiteWorth: cite-worthiness detection for improved scientific document understanding. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 1796–1807 (2021)
2. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3615–3620 (2019)
3. Hafid, S., Schellhammer, S., Bringay, S., Todorov, K., Dietze, S.: SciTweets-a dataset and annotation framework for detecting scientific online discourse. In: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, pp. 3988–3992 (2022)
4. Beltagy, I., Peters, M.E., Cohan, A.: LongFormer: the long-document transformer. arXiv preprint arXiv:2004.05150 (2020)
5. Lo, K., Wang, L.L., Neumann, M., Kinney, R., Weld, D.S.: S2ORC: the semantic scholar open research corpus. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 4969–4983 (2020)
6. Färber, M., Thiemann, A., Jatowt, A.: To cite, or not to cite? Detecting citation contexts in text. In: Pasi, G., Piwowarski, B., Azzopardi, L., Hanbury, A. (eds.) ECIR 2018. LNCS, vol. 10772, pp. 598–603. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76941-7_50
7. Della Giusta, M., Jaworska, S., Vukadinović Greetham, D.: Expert communication on Twitter: comparing economists' and scientists' social networks, topics and communicative styles. Public Underst. Sci. **30**(1), 75–90 (2021)
8. Moturu, S.T., Liu, H.: Quantifying the trustworthiness of social media content. Distrib. Parallel Databases **29**, 239–260 (2011)
9. Sundriyal, M., Akhtar, M.S., Chakraborty, T.: Leveraging social discourse to measure check-worthiness of claims for fact-checking. arXiv preprint arXiv:2309.09274 (2023)
10. Satapara, S., Mehta, P., Ganguly, D., Modha, S.: Fighting fire with fire: adversarial prompting to generate a misinformation detection dataset. arXiv preprint arXiv:2401.04481 (2024)
11. Eysenbach, G.: Can tweets predict citations? Metrics of social impact based on Twitter and correlation with traditional metrics of scientific impact. J. Med. Internet Res. **13**(4), e2012 (2011)
12. Jain, N., Singh, M.: TweetPap: a dataset to study the social media discourse of scientific papers. In: 2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 328–329. IEEE (2021)
13. August, T., Card, D., Hsieh, G., Smith, N.A., Reinecke, K.: Explain like I am a scientist: the linguistic barriers of entry to r/science. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems, pp. 1–12 (2020)

14. Chandrasekharan, E., et al.: The Internet's hidden rules: an empirical study of Reddit norm violations at micro, meso, and macro scales. In: Proceedings of the ACM on Human-Computer Interaction, vol. 2, no. CSCW, pp. 1–25 (2018)
15. Allcott, H., Gentzkow, M.: Social media and fake news in the 2016 election. J. Econ. Perspect. **31**(2), 211–236 (2017)
16. Vosoughi, S., Roy, D., Aral, S.: The spread of true and false news online. Science **359**(6380), 1146–1151 (2018)
17. Garimella, K., Morales, G.D.F., Gionis, A., Mathioudakis, M.: Quantifying controversy on social media. ACM Trans. Soc. Comput. **1**(1), 1–27 (2018)
18. De Semir, V.: Scientific journalism: problems and perspectives. Int. Microbiol. **3**(2), 125–128 (2000)
19. Dunwoody, S.: Science journalism: prospects in the digital age. In: Routledge Handbook of Public Communication of Science and Technology, pp. 14–32. Routledge (2021)
20. Arnold, P.: The challenges of online fact checking. Technical report, Full Fact (2020)
21. Didegah, F., Mejlgaard, N., Sørensen, M.P.: Investigating the quality of interactions and public engagement around scientific papers on Twitter. J. Informet. **12**, 960–971 (2018)
22. Liu, Y., Whitfield, C., Zhang, T., Hauser, A., Reynolds, T., Anwar, M.: Monitoring COVID-19 pandemic through the lens of social media using natural language processing and machine learning. Health Inf. Sci. Syst. **9** (2021)
23. Raza, H., Faizan, M., Hamza, A., Mushtaq, A., Akhtar, N.: Scientific text sentiment analysis using machine learning techniques. Int. J. Adv. Comput. Sci. Appl. (2019)
24. Sugiyama, K., Kumar, T., Kan, M., Tripathi, R.C.: Identifying citing sentences in research papers using supervised learning. In: 2010 International Conference on Information Retrieval & Knowledge Management (CAMP), pp. 67–72 (2010)
25. Bird, S., et al.: The ACL anthology reference corpus: a reference dataset for bibliographic research in computational linguistics. In: International Conference on Language Resources and Evaluation (2008)
26. Färber, M., Thiemann, A., Jatowt, A.: A high-quality gold standard for citation-based tasks. In: International Conference on Language Resources and Evaluation (2018)
27. Alperin, J.P., Fleerackers, A., Riedlinger, M., Haustein, S.: Second-order citations in altmetrics: a case study analyzing the audiences of COVID-19 research in the news and on social media. Quant. Sci. Stud. 1–28 (2024)
28. Nakov, P., et al.: Overview of the CLEF-2022 CheckThat! Lab task 1 on identifying relevant claims in tweets. In: 2022 Conference and Labs of the Evaluation Forum, CLEF 2022, pp. 368–392. CEUR Workshop Proceedings. CEUR-WS.org (2022)
29. Alam, F., et al.: Overview of the CLEF-2023 CheckThat! Lab task 1 on check-worthiness in multimodal and multigenre content. In: Working Notes of CLEF (2023)
30. Fang, Z., Costas, R., Tian, W., Wang, X., Wouters, P.: An extensive analysis of the presence of altmetric data for Web of science publications across subject fields and research topics. Scientometrics **124**(3), 2519–2549 (2020)

Towards a Novel Classification of Table Types in Scholarly Publications

Jilin He[1], Ekaterina Borisova[2(✉)], and Georg Rehm[2]

[1] Technische Universität Berlin (TU), Berlin, Germany
jilin.he@campus.tu-berlin.de

[2] Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), Berlin, Germany
{ekaterina.borisova,georg.rehm}@dfki.de

Abstract. Tables are one of the prevalent means of organising and representing structured data. They contain a wealth of valuable information that is challenging to extract automatically, yet can be leveraged for downstream tasks such as question answering and knowledge base construction. Table Type Classification (TTC) is one of the tasks which contributes to better semantic understanding and extraction of knowledge in tabular data. While multiple classification schemas exist, almost all of them are focused on web tables. Therefore, these classifications might overlook certain types which are common in other areas such as scientific research. This paper addresses this gap by introducing ten novel TTC taxonomies tailored towards tables used in scholarly publications. We also evaluate the applicability of taxonomies derived from web tables to scientific tables. Additionally, we propose a new dataset containing 13,000 annotated table images, called TD4CLTabs. Our results indicate that both existing and newly proposed taxonomies are suitable and effective for classifying scientific tables.

Keywords: Table type classification · taxonomy construction · table understanding

1 Introduction

Tables are used to summarise and present information in a structured manner across various areas such as business, finance, science, education, and healthcare [40]. With a growing interest in the field of Table Understanding (TU), several studies have focused on the automatic extraction of knowledge from tables [3,16,36,45] and applying it to various tasks, e. g., question answering [5,7,9,20,22,29,33,43,48,50], knowledge base construction [25,27], table-to-text generation [28], tabular data augmentation [12,44,45], content extension and completion [21,27], fact-checking [1,6], and natural language inference [17].

Table Type Classification (TTC) is the TU sub-task aimed to categorise tables according to a predefined schema based on their layout structure, content or purpose of use [45]. Classifying tables into specific types helps to uncover

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 31–48, 2024.
https://doi.org/10.1007/978-3-031-65794-8_3

the semantics of the data they contain, facilitating tasks such as detecting and filtering layout tables (which do not contain any meaningful data), recognising table structures, and information extraction [14,15,23,25]. Even though various TTC schemas exist [4,8,11,25–27,41], most were designed focusing on tabular structures that exist in web pages, commonly referred to as *web tables* [26]. As a consequence, these classifications might overlook certain table features and types, especially domain specific ones. In particular, they might not be fully applicable to tables found in scholarly papers. We refer to such tables as *scientific tables*, defining them as tabular structures found in (digital) scholarly publications and labelled as a table by the authors. To the best of our knowledge, there is only one study by Kruit et al. [25] that proposed a table type taxonomy derived from scientific tables. No taxonomies based on structural or layout features exist for the field of scientific publications. The present paper addresses this gap by developing ten novel taxonomies based on scientific tables. To this end, we collect a corpus of tables extracted from Computational Linguistics (CL) articles. We develop various taxonomies based on two well-established classification schemas and by considering table features identified in previous studies and our own corpus analysis. We train and evaluate classifiers on the dataset of scientific tables that we annotated according to the two pre-existing schemas and our newly proposed taxonomies.

Our contributions can be summarised as follows:

– We construct and release the TD4CLTabs dataset with 13,000 annotated images of scientific tables extracted from CL articles.
– We propose and evaluate ten novel TTC taxonomies defined based on scientific tables.
– We assess the applicability of taxonomies derived from web tables to scientific tables.
– We offer a list of table features which are potentially important for TTC. The list includes attributes considered by previous taxonomies, alongside those overlooked by these schemas but identified in the literature and in our TD4CLTabs dataset.

This article is structured as follows: Sect. 2 discusses related work. Section 3 describes our approach to the dataset and taxonomies construction. Sections 4 and 5 present the evaluation results and main findings, respectively. Section 6 outlines limitations. Concluding remarks are provided in Sect. 7.

2 Related Work

Tables are ubiquitous data structures, often stored in relational databases (e. g., MySQL, PostgreSQL), spreadsheets (e. g., Microsoft Excel, Google Sheets), web pages (e. g., Wikipedia), and scientific articles. Tables vary greatly in terms of their layout structures and content, posing challenges for automatic TU [2,46]. In order to effectively process and extract knowledge from tables, several TTC schemas have been proposed.

The existing schemas vary in their complexity, ranging from simple binary classifications to multi-layer taxonomies. Additionally, most TTC schemas have been designed based on tables found in web pages. For instance, in the pioneering work by Wang and Hu [42], web tables were classified into two categories: *genuine*, i. e., leaf tables (not containing other tables, lists, images, etc.) and *non-genuine*. Later Cafarella et al. [4] distinguished between *extremely small* tables, *HTML forms*, *calendars*, *non-relational* (contain low-quality data), and *relational* (contain high-quality data) tables. Subsequent studies proposed more fine-grained classifications by organising table types into hierarchical taxonomies. Crestan et al. [11] introduced the categories of *relational knowledge* tables, which contain relational data, and *layout* tables, which do not contain any meaningful data at all. The former class included sub-types defined based on the positioning of table headers: *vertical listing*, *horizontal listing*, *matrix*, *attribute/value*, *enumeration*, and *calendar*. The layout category contained *formatting* and *navigational* tables. Lautert et al. [26] refined this taxonomy by revisiting the relational knowledge tables class and incorporating types derived from cell features. On the first layer, relational knowledge tables were categorised as *horizontal*, *vertical*, and *matrix*. These were subsequently divided into *concise* (contain merged cells), *nested* (contain a table in a cell), *splitted* (contain repeated labels in headers), *simple* and *composed multivalued* (contain multiple values in a single cell) categories. Chen and Cafarella [8] devised an alternative TTC taxonomy focusing on the use-case of web spreadsheets. In contrast to previous studies, this taxonomy incorporates major classes such as *data frame* spreadsheets and *non-data frame* (flat) spreadsheets, along with their respective sub-categories. More recent studies have shifted back to single-level classification schemas. Eberius et al. [14] distinguished between three main table types, namely *matrix*, *horizontal listing*, and *vertical listing* (see Fig. 6 in Appendix A). Similarly, Lehmberg et al. [27] also classified tables into three major categories: *relational*, *entity*, and *matrix*.

In contrast to web tables, there is currently only one TTC taxonomy defined based on scientific tables extracted from Computer Science papers. It was proposed by Kruit et al. [25] for the development of Tab2Know, i.e., a novel end-to-end system for building a knowledge base from scientific tables. This taxonomy consists of four root classes (*observation*, *example*, *input*, *other*) with their respective sub-classes and primarily focuses on the narrative role tables play in scholarly articles rather than their structural characteristics.

As emphasised by Zhang and Balog [45], the established approaches to TTC were designed for different use-cases. Therefore, it is not surprising that existing schemas might overlook certain table features. For instance, Shigarov et al. [38, 39] highlighted that current classifications fail to address header and cell-related characteristics such as header hierarchies, the presence of non-textual content and diagonally split cells. Additionally, the schemas do not consider the concepts of complicated tables (i. e., containing spanning cells) and void cells introduced by Chi et al. [10] and Rolan et al. [35], respectively (see Fig. 7 in Appendix B).

In earlier studies, TTC relied on traditional machine learning algorithms such as decision trees, support vector machines, and logistic regression [4, 11, 14, 25, 26, 42]. Recent research has shifted towards the adoption of deep learning techniques such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and attention mechanisms for automatic feature extraction from tables [18, 31]. Previous approaches primarily utilised plain-text and HTML representations of tables. However, not all tables are readily accessible in a machine-readable format. For instance, scientific tables are commonly embedded in unstructured PDF documents. Such tables have to be extracted and transformed into a format suitable for training and testing models. One of the widely used approaches involves obtaining the image-like representations of tables from a PDF file [24, 25, 49] which can either be directly used as model input or first converted into structured formats like CSV or JSON.

3 Methodology

3.1 Data

To assess the applicability of web tables-based taxonomies to the area of science and to construct novel TTC taxonomies, we created a corpus of table images from scholarly articles in the ACL Anthology.[1] We fetched a total of 3,219 papers from the year 2022, chosen as the latest collection of publications in the readily available ACL Anthology corpus.[2] As ACL papers are available only in PDF, Tab2Know was used to obtain table images. Out of the 3,219 PDF files, Tab2Know successfully processed 2,687, resulting in a total of 15,292 table images. Since Tab2Know is designed to locate and extract tables without their respective captions and titles, these are not present in our corpus.

3.2 Taxonomies Construction

We applied two established schemas based on web tables to the corpus of scientific tables, i. e., the classifications proposed by Eberius et al. [14] and Crestan et al. [11]. We picked these two taxonomies based on their usage in recent applications and tasks. We did not consider the taxonomy proposed by Kruit et al. [25] since it classifies tables based on their narrative role in scientific articles rather than their layout structure.

In order to determine whether any adjustments are needed in the two taxonomies, such as excluding under-represented classes, we examined their presence and distribution in a sample of 1200 table images from our corpus. The results are presented in Fig. 1. Eberius et al.'s schema, featuring the classes *listing* and *matrix*, was directly adopted to the TTC task due to their high frequency in the corpus. The taxonomy by Crestan et al. was adjusted by keeping *horizontal listing*, *vertical listing*, *matrix*, and *enumeration*, while disregarding other classes (e. g., calendar, form, layout tables, etc.) since these could not be observed in

[1] https://aclanthology.org.
[2] https://github.com/shauryr/ACL-anthology-corpus.

the sample data. Additionally, all tables of the attribute/value class were classified as either vertical listing or horizontal listing since they represent specific instances of these classes [11]. Together with the class *other tables*, which was introduced for tables that do not fit any of the pre-defined classes, we refer to the final two taxonomies as Baseline_I and Baseline_II, respectively. The graphical illustration of the baseline taxonomies is provided in Fig. 2(a).

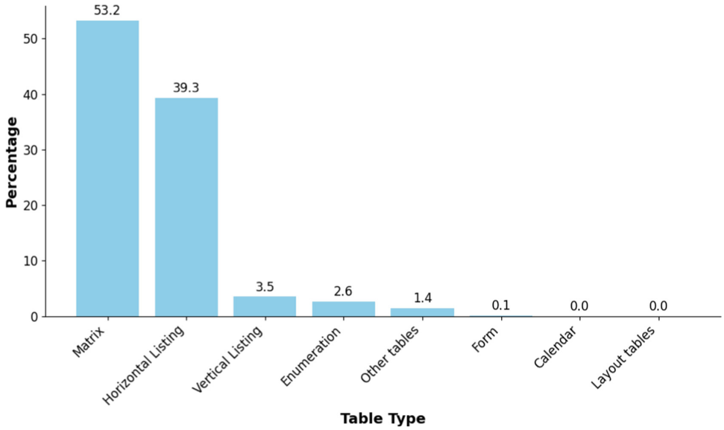

Fig. 1. The distribution of table types defined by Crestan et al. [11] and Eberius et al. [14] in a sample of 1200 table images extracted from the ACL Anthology Corpus.

In addition, ten novel taxonomies were defined by incorporating the table types from the baseline taxonomies as well as header and cell features. As a fist step, we determined which classes should be preserved from Baseline_I and Baseline_II by analysing the results of their preliminary frequency of occurrence (Fig. 1). Hence, only the matrix and horizontal listing classes were considered while designing the taxonomies. Vertical listing and enumeration were disregarded due to their low frequencies in the dataset. Then, we compiled a list of table layout features which are neglected by the existing taxonomies but distinguished by previous studies (see Sect. 2). We further extended the list with additional features observed during the examination of the 1200 sample tables. The collected features fall into header and other table attributes and are outlined in Table 1.

Initially, we constructed the TTC taxonomies by combining the selected table types and additional header features. We refer to these as Header-Feature Table Taxonomies (HFTTs) and present them in Figs. 2 (b) and (c). Thus, taking into account the absence or presence of a header hierarchy, we extended Baseline_I with the classes *flat listing*, *flat matrix*, *hierarchical listing*, and *hierarchical matrix* classes, and called it HFTT_Novel_I. Then, we incorporated the positioning of hierarchical headers (HHs) within the classes matrix and horizontal listing into HFTT_Novel_I. For the former, HH might exclusively appear

Table 1. Header and other features potentially significant for Table Type Classification. Attributes identified based on a sample of 1200 tables extracted from ACL papers are highlighted in italics.

Header Features
– Positioning of headers [11,26]
– Hierarchy of headers [39]
– *Varied positioning of hierarchical headers in Matrix*
– Presence of diagonally split cells in Matrix [38]
Other Features
– Presence of missing and void cells [35]
– Presence of non-textual content [38]
– *Presence of hierarchical rows*
– Presence of spanning cells [10,26]
– *Presence of other complex cells*
– Table splitting [26]

in a column header (CH), row header (RH), or in both. We refer to these three additional classes as *type-1*, *type-2*, *type-3 hierarchical matrix*. In the case of horizontal listing, HH may be positioned on the left, right or middle of a table, potentially with repetitions. We name the resulting taxonomy HFTT_Novel_II. As can be seen from Fig. 2(b), for HFTT_Novel_III, we further distinguished between matrix with diagonally split cells at the top-left cell (*pseudo matrix*) and without those (*regular matrix*). Note that pseudo matrices often bear a resemblance to listing. For the final HFTT_Novel_IV, we excluded HH and the three respective HH positioning types related to matrix and pseudo matrix. Eventually, the ten different taxonomies developed vary in terms of their number of classes, from 3 to 17. Baseline_I contains the fewest number of categories, while FFTT_Novel_V includes the highest number.

As outlined in Table 1, HFTT can be extended with other table features related to cell types and table splitting. Thus, each feature introduces a new category within each table type across HFTTs. When focusing solely on header features, the resulting table types are mutually exclusive. For instance, if a table is categorized as matrix, it cannot simultaneously belong to the listing class. Similarly, once it falls into the type-1 hierarchical matrix, it cannot be classified as type-2, type-3 or pseudo matrix. However, when considering both header and other table features, the resulting table types become inclusive. Thus, matrix can exhibit features such as spanning cells and being split at the same time, leading to a new category called *split complex matrix*. We refer to the refined HFTTs, containing header features, cell-related attributes, and table splitting, as Full-Feature Table Taxonomies (FFTTs). Figure 3 shows two examples.

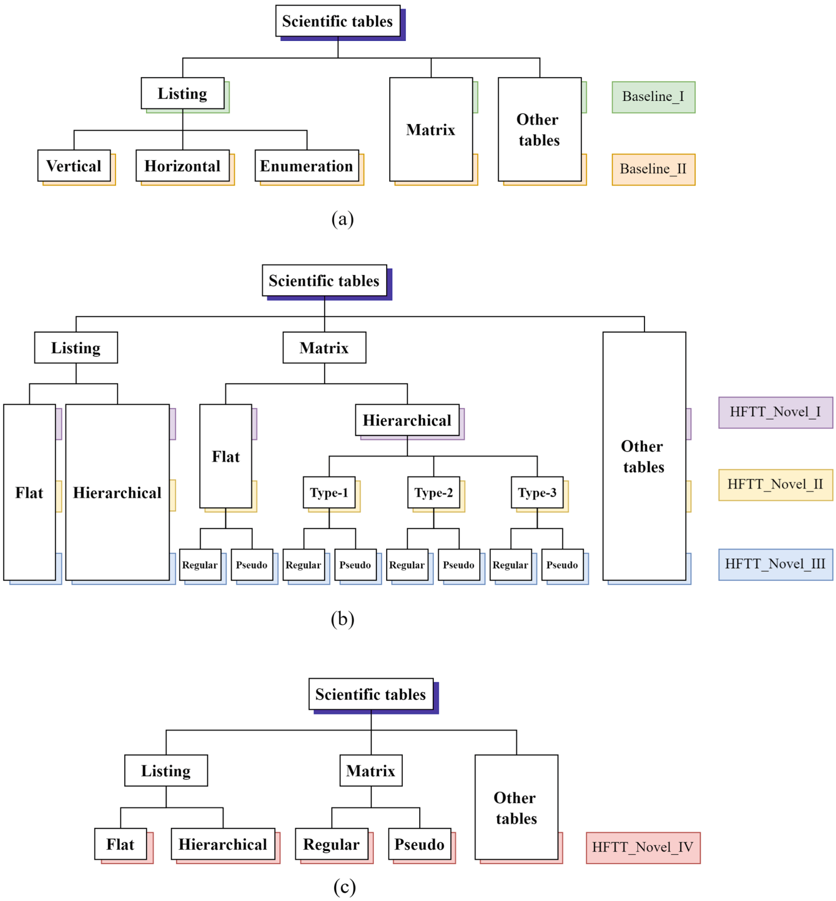

Fig. 2. The table type taxonomies proposed in this study: Figure (a) depicts two baseline taxonomies, while (b) and (c) illustrate four newly defined taxonomies. The colours highlight each taxonomy and its respective classes.

Method	BC5CDR (Big Dict)	BC5CDR (Small Dict)
Fully Supervised	85.00 (77.68/93.83)	
Binary PU	72.38 (61.89/87.17)	79.14 (81.85/76.60)
Method	CoNLL2003 (KB)	CoNLL2003 (Dict)
Fully Supervised	96.09 (92.77/99.65)	
Binary PU	88.88 (81.04/98.39)	88.08 (79.09/99.38)

(a) Split Complex Matrix

Dataset	Source(s)	Target	Context	Evidence	#Instances	Task
English Datasets						
Rumour Has It (Qazvinian et al., 2011)		Topic	Tweet		10K	Rumours
PHEME (Zubiaga et al., 2016b)		Claim	Tweet		4.5K	Rumours
Emergent (Ferreira and Vlachos, 2016)		Headline	Article*		2.6K	Rumours
FNC-1 (Pomerleau and Rao, 2017)		Headline	Article		75K	Fake news
RumourEval '17 (Derczynski et al., 2017)		Implicit[1]	Tweet		7.1K	Rumours
FEVER (Thorne et al., 2018)	W	Claim	Facts		185K	Fact-checking
Snopes (Hanselowski et al., 2019)	Snopes	Claim	Snippets		19.5K	Fact-checking
RumourEval '19 (Gorrell et al., 2019)		Implicit[1]	Post		8.5K	Rumours
COVIDLies (Hossain et al., 2020)		Claim	Tweet		6.8K	Misconceptions
TabFact (Chen et al., 2020)	W	Statement	WikiTable		118K	Fact-checking
Non-English Datasets						
Arabic FC (Baly et al., 2018b)		Claim	Document		3K	Fact-checking
DAST (Danish) (Lillie et al., 2019)		Submission	Comment		3K	Rumour
Croatian (Bošnjak and Karan, 2019)		Title	Comment		0.9K	Claim verifiability
ANS (Arabic) (Khouja, 2020)		Claim	Title		3.8K	Claim verification
Ara(bic)Stance (Alhindi et al., 2021)		Claim	Title		4K	Claim verification

(b) Listing with the presence of hierarchical rows and non-textual contents

Fig. 3. Examples of scientific tables belonging to the Full-Feature Table Taxonomies.

3.3 Annotation

To label the corpus of 15,292 table images according to the defined taxonomies, we run an annotation project. LabelStudio[3] was used as the annotation tool and since there was only one annotator involved, a Master student of Data Science, no inter-annotator agreement (IAA) score was calculated. To ensure that the final corpus contains well-structured images, displaying only the complete and clear layout of tables, we filtered out inappropriate samples while annotating. To this end, we introduced the class *non-table* and used the following rules during the annotation:

– If a table is partially extracted, as if incorrectly cropped, it is not considered to be a complete table and should be annotated as non-table.
– If a table is fully extracted but labelled as Figure in a paper, it should be annotated as non-table.
– If a table is fully extracted but there is other information in the image, such as segments of text, it should be annotated as non-table.
– If a table is fully extracted but an image contains multiple scattered tables, it is considered as incorrect input and should be annotated as non-table.

As a result, 280 table images belong to the non-table category and were excluded from the corpus. We also checked the labelled data with respect to annotation errors. Consequently, 54 images were removed from the corpus.

The final dataset comprises 13,301 annotated scientific table images along with their respective metadata (image name, image label, image path, and dataset split). We refer to the final corpus as TD4CLTabs (Type Detection for Computational Linguistics Tables) dataset.[4] As a post-processing step, we encoded the categorical features with numerical values. Then we divided the dataset into a training set containing 10,347 table images and a test set comprising 2,954 samples.

3.4 Models

Considering recent advances of deep learning in computer vision (CV), alongside the proven successful application of table images for TU tasks such as table detection and table structure recognition [30,32,34,37,49], we approach TTC as an image classification task. In particular, TTC based on HFTTs was tackeled as a multi-class problem, while classification based on FFTTs was addressed as a multi-label task.

Two models, ResNet50 [19] and Vision Transformer (Vit) [13], were trained. [5] ResNet50 is a deep CNN model widely utilised in CV tasks, exhibiting efficient

[3] https://labelstud.io.
[4] https://zenodo.org/records/10972922.
[5] The code is available on Software Heritage: https://archive.softwareheritage.org/browse/directory/1f492fb7db23db3a57484edd196af4fdf7139061/?origin_url=https://github.com/JilinHe/TD4CLTabs&revision=b549ac21bb59386734457eb6a36b8d358b0a68ee&snapshot=6b93b959741a8fbfff4f5ebeaf71e8177b81ff6f.

performance in image classification problems. ViT presents a newer approach to CV, utilising the Transformer architecture's unique ability to capture global image information, outperforming traditional CNN models. We combined pre-encoded labels from all hierarchy levels into one flat list and fed them as input into the models along with table images.

ResNet50 was implemented using the Fastai framework.[6] For the Vit model, we utilised the Hugging Face implementation.[7] To enhance the robustness and reliability of the image classification models, cross-validation was applied with k set to 4. For both models, the batch size was set to 16. The resize dimensions of (500, 900) and (224, 224) were chosen for ResNet50 and Vit, respectively. FocalLoss was employed as the loss function for ResNet50, while the default CrossEntropy was used for Vit. The training process for ResNet50 extended to 30 epochs with early stopping enabled and a patience of 5 epochs. Vit was trained for 15 epochs with the option to save the best model. Both models utilised pretrained weights, with ResNet50 set to True and Vit using the 'google/vit-base-patch16-224-in21k' pretrained configuration.

3.5 Evaluation Metrics

To evaluate the performance of the two models on the multi-class classification task, error rate, precision (weighted), recall (weighted), and F1 score (weighted) were used. In the case of multi-label classification, hamming loss, macro and micro F1 scores were utilised.

4 Results

4.1 Dataset Analysis

The table images in our dataset have a wide range of resolutions, spanning from a minimum of 100×100 pixels to a maximum of either 1200×200 or 1000×1400 pixels. In terms of dimensions, tables average 7.60 rows and 6.68 columns.

The distribution of tables per class within each HFTT is presented in Fig. 4. As can be seen, with the increase in the number of classes, the degree of data imbalance also rises. The analysis shows that matrix tables are approximately 15% more common than listings in the dataset. Interestingly, other tables comprise less than 5%. Among the matrix tables, those with HHs constitute approximately half of all (49%). Furthermore, the majority of such tables (about 64%) fall under type-1 hierarchical matrix, i.e., have HHs located in a CH. Matrix tables with diagonally split cells are quite frequent (about 71%). The least common across the matrix sub-categories are type-2 hierarchical and type-3 hierarchical. In terms of the listing class, horizontal tables are more frequent (about 84% of the total) than vertical and enumeration types. In contrast to hierarchical matrix tables, the number of hierarchical listings in the dataset is considerably lower (approx. 8% of all listings).

6 https://www.fast.ai.
7 https://huggingface.co.

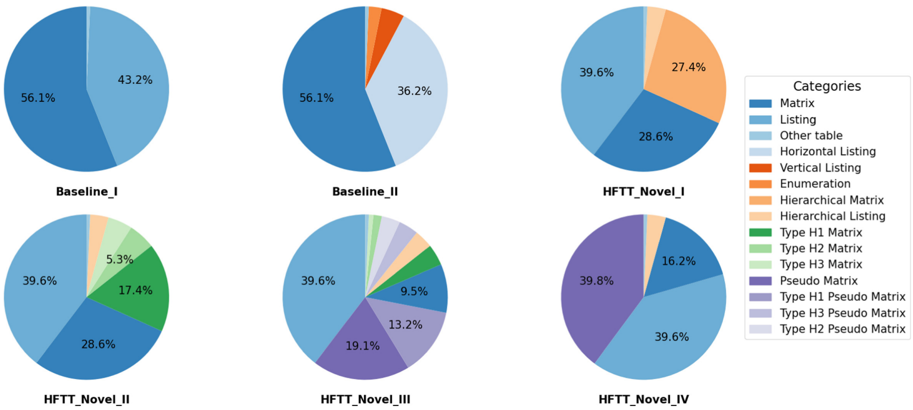

Fig. 4. The distribution of table types in the baseline and Header-Feature Table Taxonomies within the TD4CLTabs dataset. Note that only proportions exceeding 5% are explicitly labelled with numerical values.

Figure 5 illustrates the distribution of table splitting and cell-related features incorporated into FFTTs within the TD4CLTabs dataset. The results indicate the infrequent occurrence of those across the given corpus of scientific tables. The highest value of about 13% was achieved for the missing and void cells type, followed by the presence of hierarchical rows (approximately 10%). A limited number of tables contain cells with non-textual content (about 3%) and other complex cells (about 2%).

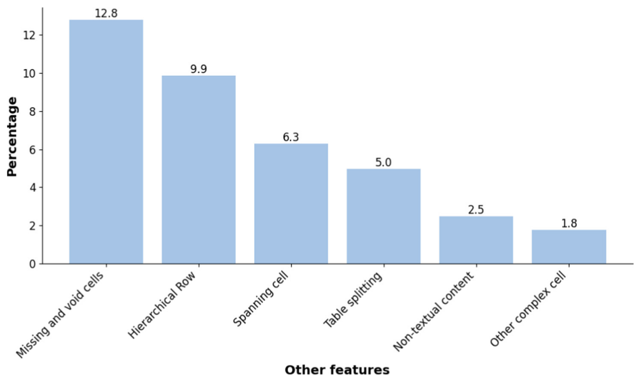

Fig. 5. The distribution of cell types and table splitting across the TD4CLTabs dataset

4.2 Table Type Classification

Table 2 presents the TTC results across HFTTs. The Vit model outperforms ResNet50 in all but one case, namely HFTT_Novel_II. We can also see a general trend of decreasing performance among the models as the number of classes in the taxonomy increases. The class imbalance indicated in Sect. 4.1 might have also influenced the predictions. The best F1 value (0.82) was obtained for Vit based on Baseline_I. This is not surprising since it is a 1-level schema with the least number of classes and the most balanced data. The second highest F1 scores (0.78) were achieved by Baseline_II and HFTT_Novel_IV, both of which contain two additional categories when compared to Baseline_I. Even though HFTT_Novel_III contains four more categories than HFTT_Novel_II, the models based on these taxonomies result in very similar results (approx. 1% difference). The study also shows that HFTT_Novel_IV achieved the highest scores among the novel taxonomies.

Table 2. Multi-class classification results based on baseline and Header-Feature Table Taxonomies

Taxonomy	ResNet50				Vit			
	Error Rate	Precision	Recall	F1	Error Rate	Precision	Recall	F1
Baseline_I	0.22	0.78	0.78	**0.77**	0.17	0.82	0.83	**0.82**
Baseline_II	0.23	0.76	0.77	**0.76**	0.21	0.77	0.79	**0.78**
HFTT_Novel_I	0.26	0.74	0.74	**0.73**	0.23	0.74	0.77	**0.75**
HFTT_Novel_II	0.27	0.73	0.73	**0.72**	0.26	0.72	0.74	**0.71**
HFTT_Novel_III	0.28	0.73	0.72	**0.71**	0.27	0.72	0.73	**0.72**
HFTT_Novel_IV	0.25	0.76	0.75	**0.75**	0.21	0.78	0.79	**0.78**

The results for multi-label classification based on FFTTs are provided in Table 3. In terms of micro F1, the Vit model demonstrates overall better performance compared to ResNet50 across all taxonomies, except FFTT_Novel_IV and FFTT_Novel_V. However, all models exhibit low macro F1 scores, indicating the dataset imbalance. The hamming loss values are also consistently low across the models (0.05–0.07), suggesting an overall good performance of the classifiers. Similar to the classification based on HFTTs, we note a trend where models tend to perform worse on FFTTs with a larger number of classes. Furthermore, the highest score (0.75) for FFTTs is about 7% and 2% lower compared to those obtained for the baselines and HFTTs, respectively.

To address the problem of class imbalance, we applied the random oversampling technique [47] on novel HFTTs.[8] This involved duplicating instances of the minority classes to align with the majority classes. As shown in Table 4, oversampling consistently improved F1 scores by 1–5% across the models. The Vit model based on HFTT_Novel_IV is the only instance where a slight

[8] Note that we have not addressed the data imbalance for FFTTs.

Table 3. Multi-label classification results based on Full-Feature Table Taxonomies. The threshold is set to 0.5. If the probability of the prediction is greater than 0.5, it as a positive prediction. Otherwise, it is a negative prediction.

Taxonomy	ResNet50			Vit		
	$F1_{Micro}$	$F1_{Macro}$	Hamming Loss	$F1_{Micro}$	$F1_{Macro}$	Hamming Loss
FFTT_Novel_I	**0.73**	0.54	0.07	**0.75**	0.38	0.07
FFTT_Novel_II	**0.69**	0.49	0.06	**0.72**	0.32	0.06
FFTT_Novel_III	**0.70**	0.55	0.07	**0.71**	0.37	0.07
FFTT_Novel_IV	**0.69**	0.58	0.06	**0.68**	0.36	0.06
FFTT_Novel_V	**0.66**	0.53	0.05	**0.61**	0.25	0.05
FFTT_Novel_VI	**0.70**	0.54	0.07	**0.72**	0.47	0.06

decrease in score (by about 2%) is observed. All other evaluation scores also increased in the majority of HFTT classifiers. Furthermore, comparable results to ResNet50 with Baseline_I were achieved on ResNet50 with HFTT_Novel_I and HFTT_Novel_IV. However, despite the overall improvement in model performance, the prediction accuracy for novel taxonomies still remains lower (by approximately 5%) than that of Baseline_I based on Vit.

Table 4. Multi-class classification results based Header-Feature Table Taxonomies after applying oversampling

Taxonomy	ResNet50				Vit			
	Error Rate	Precision	Recall	F1	Error Rate	Precision	Recall	F1
HFTT_Novel_I	0.22	0.78	0.78	**0.77**	0.24	0.77	0.76	**0.76**
HFTT_Novel_II	0.25	0.75	0.75	**0.74**	0.24	0.76	0.76	**0.76**
HFTT_Novel_III	0.24	0.77	0.76	**0.75**	0.24	0.77	0.77	**0.75**
HFTT_Novel_IV	0.22	0.78	0.78	**0.77**	0.24	0.78	0.76	**0.76**

5 Discussion

The study indicates that matrix and listing tables are the most commonly used across CL papers. In particular, matrix with hierarchical headers, frequently found in CHs, matrix with diagonally split cells, and horizontal listings are prevalent. Hence, these types are worth considering when classifying scientific tables. In contrast, the findings suggest that incorporating table splitting and cell features may not be advantageous, as they seem to be relatively uncommon in scientific tables.

The study further showcased the applicability of the TTC schema by Eberius et al. to scientific tables. In this sense, Crestan's et al. taxonomy also proved to be adaptable after smaller adjustments. The models based on these baseline schemas demonstrate greater efficiency on TTC than those trained on the newly proposed taxonomies. Hence, although the two established classification schemas were designed for web tables, they are still suitable for scientific tables.

While the experimental results do not demonstrate a clear advantage of the novel domain-specific taxonomies, they do show the promising outcomes. Among the newly developed taxonomies, HFTT_Novel_I and HFTT_Novel_VI have proven to be the most successful. This could potentially be attributed to the smaller number of categories within those, indicating a lower level of complexity, compared to other schemas. These taxonomies also achieved efficiency comparable to the results obtained for ResNet50 with the baseline schemas.

6 Limitations

While this study sheds light on devising TTC taxonomies for scientific tables, it is not without limitations. First, the annotations may be subjective and contain errors due to the involvement of only one annotator. Having at least one additional annotator and curator, and subsequently validating the results by calculating the IAA score, would be beneficial. Second, the novel taxonomies were constructed and tested based on scientific tables from CL papers. Thus, the applicability of those to other domains remains an open research question, which we leave for future work. Third, the study considered only two existing web table based taxonomies, limiting the analysis to types within them and potentially neglecting other categories relevant to scientific tables. Finally, the hierarchy of the taxonomies' labels was not taken into account in this study. Additionally, to tackle class imbalance, we considered only oversampling and applied it only to taxonomies with header features. Future endeavours could incorporate the label hierarchy in the model training process and focus on annotating more samples for the minority classes or on utilising other automatic methods for solving class imbalance (e. g., resampling).

7 Conclusion

In this paper, we developed and evaluated the effectiveness of ten novel TTC taxonomies tailored for tables found in scholarly publications. Additionally, we examined the applicability of well-established schemas designed for and based on web tables to the use-case of scientific tables. The findings reveal that existing taxonomies are indeed suitable for classifying scientific tables. However, while established taxonomies demonstrate their efficiency, comparable performance can also be achieved with two novel domain-specific taxonomies. Finally, our study indicates that header features are essential for classifying scientific tables, whereas cell features and table splitting have not shown to provide significant advantages. The proposed taxonomies can be beneficial for downstream tasks such as information retrieval from scholarly papers by helping to reduce the search space, data integration allowing mapping of scientific tables with similar structures across different datasets, and scientific table structure recognition.

Acknowledgments. The work presented in this paper was partially supported by the consortium NFDI for Data Science and Artificial Intelligence (NFDI4DS, no. 460234259) (https://www.nfdi4datascience.de) as part of the non-profit association National Research Data Infrastructure (NFDI e. V.). The NFDI is funded by the Federal Republic of Germany and its states.

A Examples of Matrix, Horizontal Listing, and Vertical Listing Tables

	Lake	Area
1	Windermere	5.69 sq mi (14.7 km^2)
2	Kielder Reservoir	3.86 sq mi (10.0 km^2)
3	Ullswater	3.44 sq mi (8.9 km^2)
4	Bassenthwaite Lake	2.06 sq mi (5.3 km^2)
5	Derwent Water	2.06 sq mi (5.3 km^2)

(a) Horizontal listing

Government[3]	
• Type	Mayor–Council
• Body	New York City Council
• Mayor	Bill de Blasio (D)
Area[2]	
• Total	468.9 sq mi (1,214 km^2)
• Land	304.8 sq mi (789 km^2)
• Water	164.1 sq mi (425 km^2)
• Metro	13,318 sq mi (34,490 km^2)
Elevation[4]	33 ft (10 m)

(b) Vertical listing

	Right-handed	Left-handed	Total
Males	43	9	52
Females	44	4	48
Totals	87	13	100

(c) Matrix

Fig. 6. Examples of web tables falling under the categories within the table type classification schema by Eberius et al. [14]. The samples are taken from WDC Web Table Corpus 2015.

B Illustrations of Table Features

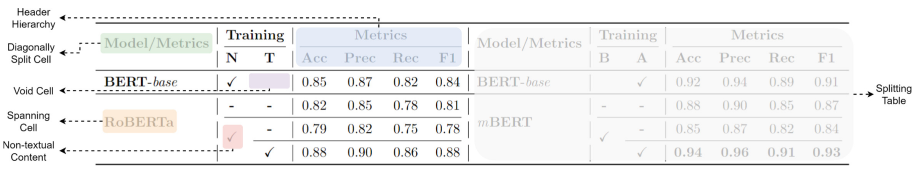

Fig. 7. Illustration of a splitting table with spanning cells, diagonally split cells, void cells, hierarchical headers, and cells with non-textual content.

References

1. Aly, R., et al.: The fact extraction and VERification over unstructured and structured information (FEVEROUS) shared task. In: Aly, R., et al. (eds.) Proceedings of the Fourth Workshop on Fact Extraction and VERification (FEVER), pp. 1–13. Association for Computational Linguistics, Dominican Republic (2021). https://doi.org/10.18653/v1/2021.fever-1.1
2. Bonfitto, S., Casiraghi, E., Mesiti, M.: Table understanding approaches for extracting knowledge from heterogeneous tables. WIREs Data Min. Knowl. Discov. **11**(4), e1407 (2021). https://doi.org/10.1002/widm.1407
3. Borisov, V., Leemann, T., Sessler, K., Haug, J., Pawelczyk, M., Kasneci, G.: Deep neural networks and tabular data: a survey. IEEE Trans. Neural Netw. Learn. Syst. 1–21 (2022). https://doi.org/10.1109/tnnls.2022.3229161
4. Cafarella, M.J., Halevy, A.Y., Zhang, Y., Wang, D.Z., Wu, E.: Uncovering the relational web. In: WebDB, pp. 1–6. Citeseer (2008)
5. Chen, W., Chang, M.W., Schlinger, E., Wang, W., Cohen, W.W.: Open question answering over tables and text. arXiv (2021)
6. Chen, W., et al.: TabFact: a large-scale dataset for table-based fact verification. In: International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia (2020)
7. Chen, W., Zha, H., Chen, Z., Xiong, W., Wang, H., Wang, W.Y.: HybridQA: a dataset of multi-hop question answering over tabular and textual data. In: Cohn, T., He, Y., Liu, Y. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2020, pp. 1026–1036. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.findings-emnlp.91
8. Chen, Z., Cafarella, M.: Automatic web spreadsheet data extraction. In: Proceedings of the 3rd International Workshop on Semantic Search Over the Web, SSW 2013. Association for Computing Machinery, New York (2013). https://doi.org/10.1145/2509908.2509909
9. Cheng, Z., et al.: HiTab: a hierarchical table dataset for question answering and natural language generation. In: Muresan, S., Nakov, P., Villavicencio, A. (eds.) Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1094–1110. Association for Computational Linguistics, Dublin (2022). https://doi.org/10.18653/v1/2022.acl-long.78
10. Chi, Z., Huang, H., Xu, H.D., Yu, H., Yin, W., Mao, X.L.: Complicated table structure recognition. arXiv preprint arXiv:1908.04729 (2019)
11. Crestan, E., Pantel, P.: Web-scale table census and classification. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 545–554 (2011)
12. Del Bimbo, D., Gemelli, A., Marinai, S.: Data augmentation on graphs for table type classification. In: Krzyzak, A., Suen, C.Y., Torsello, A., Nobile, N. (eds.) S+SSPR 2022. LNCS, vol. 13813, pp. 242–252. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-23028-8_25
13. Dosovitskiy, A., et al.: An image is worth 16×16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
14. Eberius, J., Braunschweig, K., Hentsch, M., Thiele, M., Ahmadov, A., Lehner, W.: Building the Dresden web table corpus: A classification approach. In: 2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC), pp. 41–50. IEEE (2015)

15. Ghasemi-Gol, M., Szekely, P.: TabVec: table vectors for classification of web tables. arXiv preprint arXiv:1802.06290 (2018)
16. Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A.: Revisiting deep learning models for tabular data. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 18932–18943. Curran Associates, Inc. (2021)
17. Gupta, V., Mehta, M., Nokhiz, P., Srikumar, V.: INFOTABS: inference on tables as semi-structured data. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 2309–2324. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.acl-main.210
18. Habibi, M., Starlinger, J., Leser, U.: DeepTable: a permutation invariant neural network for table orientation classification. Data Min. Knowl. Disc. **34**(6), 1963–1983 (2020)
19. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR abs/1512.03385 (2015). http://arxiv.org/abs/1512.03385
20. Herzig, J., Müller, T., Krichene, S., Eisenschlos, J.M.: Open domain question answering over tables via dense retrieval. arXiv (2021)
21. Hu, K., et al.: VizNet: towards a large-scale visualization learning and benchmarking repository. arXiv (2019)
22. Iyyer, M., Yih, W.T., Chang, M.W.: Search-based neural structured learning for sequential question answering. In: Barzilay, R., Kan, M.Y. (eds.) Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1821–1831. Association for Computational Linguistics, Vancouver (2017). https://doi.org/10.18653/v1/P17-1167
23. Kardas, M., et al.: AxCell: automatic extraction of results from machine learning papers. In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 8580–8594. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.emnlp-main.692, https://aclanthology.org/2020.emnlp-main.692
24. Karishma, Z., Rohatgi, S., Puranik, K.S., Wu, J., Giles, C.L.: ACL-Fig: a dataset for scientific figure classification. arXiv (2023)
25. Kruit, B., He, H., Urbani, J.: Tab2Know: building a knowledge base from tables in scientific papers. In: Pan, J.Z., Tamma, V., d'Amato, C., Janowicz, K., Fu, B., Polleres, A., Seneviratne, O., Kagal, L. (eds.) ISWC 2020. LNCS, vol. 12506, pp. 349–365. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62419-4_20
26. Lautert, L.R., Scheidt, M.M., Dorneles, C.F.: Web table taxonomy and formalization. ACM SIGMOD Rec. **42**(3), 28–33 (2013)
27. Lehmberg, O., Ritze, D., Meusel, R., Bizer, C.: A large public corpus of web tables containing time and context metadata. In: Proceedings of the 25th International Conference Companion on World Wide Web, WWW 2016 Companion, pp. 75-76. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE (2016). https://doi.org/10.1145/2872518.2889386
28. Moosavi, N.S., Rücklé, A., Roth, D., Gurevych, I.: Learning to reason for text generation from scientific tables. arXiv preprint arXiv:2104.08296 (2021)
29. Nan, L., et al.: FeTaQA: free-form table question answering. Trans. Assoc. Comput. Linguist. **10**, 35–49 (2022). https://doi.org/10.1162/tacl_a_00446
30. Nassar, A., Livathinos, N., Lysak, M., Staar, P.: TableFormer: table structure understanding with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4614–4623 (2022)

31. Nishida, K., Sadamitsu, K., Higashinaka, R., Matsuo, Y.: Understanding the semantic structures of tables with a hybrid deep neural network architecture. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31 (2017)
32. Paliwal, S., Vishwanath, D., Rahul, R., Sharma, M., Vig, L.: TableNet: deep learning model for end-to-end table detection and tabular data extraction from scanned document images. arXiv (2020)
33. Pasupat, P., Liang, P.: Compositional semantic parsing on semi-structured tables. In: Zong, C., Strube, M. (eds.) Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 1470–1480. Association for Computational Linguistics, Beijing (2015). https://doi.org/10.3115/v1/P15-1142
34. Prasad, D., Gadpal, A., Kapadni, K., Visave, M., Sultanpure, K.: CascadeTabNet: an approach for end to end table detection and structure recognition from image-based documents. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 572–573 (2020)
35. Roldán, J.C., Jiménez, P., Corchuelo, R.: On extracting data from tables that are encoded using HTML. Knowl.-Based Syst. **190**, 105157 (2020)
36. Sahakyan, M., Aung, Z., Rahwan, T.: Explainable artificial intelligence for tabular data: a survey. IEEE Access **9**, 135392–135422 (2021). https://doi.org/10.1109/ACCESS.2021.3116481
37. Schreiber, S., Agne, S., Wolf, I., Dengel, A., Ahmed, S.: DeepDeSRT: deep learning for detection and structure recognition of tables in document images. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), vol. 01, pp. 1162–1167 (2017). https://doi.org/10.1109/ICDAR.2017.192
38. Shigarov, A.: Table understanding: problem overview. WIREs Data Min. Knowl. Discov. **13**(1), e1482 (2023). https://doi.org/10.1002/widm.1482
39. Shigarov, A.O., Mikhailov, A.A.: Rule-based spreadsheet data transformation from arbitrary to relational tables. Inf. Syst. **71**, 123–136 (2017). https://doi.org/10.1016/j.is.2017.08.004
40. Shwartz-Ziv, R., Armon, A.: Tabular data: deep learning is not all you need. Inf. Fusion **81**, 84–90 (2022). https://doi.org/10.1016/j.inffus.2021.11.011
41. Wang, Y., Hu, J.: Detecting tables in HTML documents. In: Lopresti, D., Hu, J., Kashi, R. (eds.) DAS 2002. LNCS, vol. 2423, pp. 249–260. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45869-7_29
42. Wang, Y., Hu, J.: A machine learning based approach for table detection on the web. In: Proceedings of the 11th International Conference on World Wide Web, pp. 242–250 (2002)
43. Zayats, V., Toutanova, K., Ostendorf, M.: Representations for question answering from documents with tables and text. arXiv preprint arXiv:2101.10573 (2021)
44. Zhang, L., Zhang, S., Balog, K.: Table2vec: neural word and entity embeddings for table population and retrieval. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1029–1032 (2019)
45. Zhang, S., Balog, K.: Web table extraction, retrieval, and augmentation: a survey. ACM Trans. Intell. Syst. Technol. (TIST) **11**(2), 1–35 (2020)
46. Zheng, X., Burdick, D., Popa, L., Zhong, X., Wang, N.X.R.: Global table extractor (GTE): a framework for joint table identification and cell structure recognition using visual context. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 697–706 (2021)

47. Zheng, Z., Cai, Y., Li, Y.: Oversampling method for imbalanced classification. Comput. Inform. **34**(5), 1017–1037 (2015)
48. Zhong, V., Xiong, C., Socher, R.: Seq2SQL: generating structured queries from natural language using reinforcement learning. arXiv (2017)
49. Zhong, X., ShafieiBavani, E., Yepes, A.J.: Image-based table recognition: data, model, and evaluation. arXiv (2020)
50. Zhu, F., et al.: TAT-QA: a question answering benchmark on a hybrid of tabular and textual content in finance. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 3277–3287. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.acl-long.254

OCR Cleaning of Scientific Texts with LLMs

Gábor Madarász[1], Noémi Ligeti-Nagy[1], András Holl[2(✉)], and Tamás Váradi[1]

[1] HUN-REN Hungarian Research Centre for Linguistics, Budapest, Hungary
{gabor.madarasz,noemi.ligeti-Nagy,tamas.varadi}@nytud.hun-ren.hu
[2] Library and Information Centre, Hungarian Academy of Sciences, Budapest, Hungary
andras.holl@nytud.hun-ren.hu, holl.andras@konyvtar.mta.hu

Abstract. Correcting Optical Character Recognition (OCR) errors is a major challenge in preprocessing datasets consisting of legacy PDF files. In this study, we develop Large Language Models specially finetuned to correct OCR errors. We experimented with the mT5 model (both the mT5-small and mT5-large configurations), a Text-to-Text Transfer Transformer-based machine translation model, for the post-correction of texts with OCR errors. We compiled a parallel corpus consisting of text corrupted with OCR errors as well as corresponding clean data. Our findings suggest that the mT5 model can be successfully applied to OCR error correction with improving accuracy. The results affirm the mT5 model as an effective tool for OCR post-correction, with prospects for achieving greater efficiency in future research.

Keywords: OCR errors · Large Language Models · mT5 model · natural scientific language processing

1 Introduction

This paper reports on a collaborative project between the HUN-REN Hungarian Research Centre for Linguistics (HUN-REN NYTK) and the Library and Information Centre of the Hungarian Academy of Sciences (MTA KIK)designed to make the content of the REAL Repository more easily accessible to researchers and more easy to curate and enhance for MTA KIK. Prior to embarking on the data-mining of the texts in the Repository, the files have to be converted to machine readable raw text format. The paper will focus on techniques to clean the texts of OCR errors, which is a major challenge in this preprocessing phase. Our strategy is to compile parallel corpora consisting of sentences with OCR errors and their correct counterparts, which are used as training date to finetune a large language model so as to enable it to correct badly OCR'ed texts. The Structure of the paper is as follows: Sect. 2 describes the context and the motivation for the work, Sect. 3 reviews related work in OCR cleaning, Sect. 4

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 49–58, 2024.
https://doi.org/10.1007/978-3-031-65794-8_4

elaborates the various datasets used for the training corpus, Sect. 5 contains a brief description of the training method, Sect. 6 enumerates and discusses the results and finally, the paper ends with some Conclusions.

2 Motivation

The Library of the Hungarian Academy of Sciences was established in 1826, and since then it has been serving the members of the Academy and the whole Hungarian research community. Besides its main collection, the library has a special collection of manuscripts and rare books, and an Oriental Collection as well. The digital collections – in the form of an open access repository – were created in 2008. This repository – named REAL – has diverse holdings, mirroring the printed collection of the library. Its content is partly based on an extensive digitisation project and it contains born-digital materials too (e.g. modern journals within the scope of our library). The third source of material is the OA mandate of the Academy – researchers supported by the Academy are mandated to reposit their output in REAL. The diversity of input channels results in a mixed document content – scanned and born digital, publishers' PDFs and accepted manuscripts (with an assortment of handwritten documents and images to top it up).

The original goal of the repository was to supply digital documents for the researchers. We store PDF documents (most of which have a text layer) and the inclusion criterion was that they are suitable for the human user. Each document is checked by a librarian, so some basic document and metadata quality can be guaranteed. On the other hand, we are aware of the problems of OCR (or occasionally, the lack of it), the errors and gaps in the meta-data.

The question of language information for the documents is such a problem. Human users can obviously perceive whether a document is written in a language that is accessible for them, but we cannot filter search results for language. The lack of document language information was an early setback for our project.

The REAL Repository contains more than 250 thousand documents, about a half of which, amounting to one billion words, is suitable for the project.

The Library's most fundamental goal with this project is to enhance metadata (e.g. provide detailed language information). We would also like to improve the quality of the text layer, correcting errors in the OCR, and provide clean text layers for search and text mining.

Furthermore, we would like to be able to recognise named entities in the text. One specific task we would like to accomplish is finding references to other publications. Similarly, references to grants, large research facilities and software are also of interest to MTA KIK. (The library operates the national bibliographic database, a CRIS-like system).

In summary, we would like to improve the data and metadata quality, text-mine information for scientometric (and other) purposes, and improve the efficiency of search.

3 Related Work

There is a growing interest in utilizing neural technologies for post-OCR text correction. One of the few studies specifically addressing the correction of Hungarian texts using neural technologies is by Laki et al. [3], who explored four distinct correction experiments: machine translation with the Marian neural machine translation (NMT) system, fine-tuning a Hungarian BART model for machine translation, Context-based Character Correction (CCC) combined with machine translation using the Marian NMT system, and CCC detection with fine-tuning of Hungarian BART for machine translation.

Another notable work in this area includes research on Sanskrit texts by Maheshwari et al. [4], who reported a significant improvement in Character Error Rate (CER) using mT5 (+14.1%) and ByT5 (+23.4%) models. Piotrowski [5] focused on the application of pre-trained language models for OCR post-correction, achieving a 4.3% word error rate improvement by fine-tuning mT5 and plT5 models.

Alternative approaches to OCR correction have also been explored. Rigaud et al. [7] introduced the ICDAR2019 winning OCR correction method CCC, which combines a convolutional network for detection with a correction mechanism utilizing a BERT model and a bidirectional LSTM (Long Short Term Memory) model with an attention mechanism. Schaefer and Neudecker [8] proposed a two-step approach that includes OCR error detection with a bidirectional LSTM and subsequent error correction with a sequence-to-sequence translation model. Furthermore, Gupta et al. [2] implemented an unsupervised multi-view post-OCR error correction technique employing GPT, GPT2, and GPT2XL autoregressive models, benchmarked against a 3-gram model trained on Wikipedia. Lastly, Amrhein [1] addressed OCR error correction using a character-based NMT approach, showcasing the versatility of neural methods in enhancing OCR accuracy across various languages and scripts.

4 The Training Data

When creating the training data, we ensured that the model should be able to identify when to leave the text unchanged by including both error-free and OCR erroneous sentences, with a distribution of 33.6% error-free to 66.4% erroneous data. The dataset comprises 1,355,963 sentence pairs, encompassing a total of 51,658,231 words, with an average sentence length of approximately 19 words. The average Character Error Rate (CER) across the entire training dataset is 12.354%, and the Word Error Rate (WER) is 11.739%, when measured against the reference data (error-free sentences).

The training data was compiled from several sources, detailed below.

4.1 The "JIM Corpus"

The construction of a parallel training corpus for OCR correction involved selecting a substantial volume of text available in both electronic (error-free) and OCR-processed (erroneous) versions. This selection was manually or semi-automatically annotated, and then corrected by annotators, leading to the creation of the “JIM” corpus. The process used the complete works of **J**óka**i** and **M**ikszáth, two famous Hungarian writers, chosen for their availability in electronically published formats by the publisher, facilitating a comparison between non-OCR and OCR-processed texts.

The initial challenge was the consolidation of all works by Jókai and Mikszáth into individual files, as each author’s works were originally contained in a single file. By following the order of works listed on the https://szaktars.hu website and using a script based on the titles, the works were successfully separated into individual files. This meticulous organization was essential for matching the texts with their corresponding OCR-processed versions, which included additional elements like title pages and indexes not present in the digital editions.

Following the separation of works into individual files, the next step was the construction of a parallel corpus. This involved identifying the OCR-processed counterparts of each work and mapping them at file level, a task complicated by the digital edition containing only the text body, whereas the OCR versions included the complete books. Furthermore, inconsistencies in the availability and order of texts between the OCR versions and digital editions necessitated manual file matching. The subsequent segmentation of these works into smaller units for parallel processing was achieved through sentence-level segmentation and a novel rolling window segmentation method, addressing various challenges such as text normalization and word separation issues.

The parallel corpus underwent semi-automatic annotation to identify and categorize OCR errors, coherence issues, and punctuation differences arising from variations between editions. This process involved listing and prioritizing differences between the OCR and silver texts, ensuring that only OCR-related errors were considered during model evaluation.

Finally, the parallel corpus also underwent further manual correction by four annotators to address discrepancies caused by different editions, using both the error-containing OCR output and the error-free digital text for guidance. Corrections were made with reference to the original PDFs to align the digital text with the version from which the OCR was generated, without strictly adhering to the PDF layout or typographical errors present in the original. Adjustments included adding missing sentences from the OCR to the digital text, ignoring word breaks caused by hyphenation in the OCR that matched the PDF, and not incorporating hyphenation or page numbers from the PDF into the corrected text. The principle behind these corrections was to focus on discrepancies between the OCR text and the corrected version, aiming for textual integrity rather than slavish adherence to the original PDF formatting, especially regarding spacing around punctuation and treatment of hyphenation and page numbers.

The final version of the JIM corpus contained 646 478 sentences (OCR-ed and digital each).

4.2 The Datamaker Pipeline

A parallel corpus generated from the REAL repository materials consists of parallel sentences extracted using the pdftotext utility (version 0.86.1) from original texts produced during scanning and OCR-ed texts using Tesseract 5.0. A fully automatic pipeline processes the texts, arranging the raw texts into a training data format suitable for T5-based models.

T5 (Text-to-Text Transfer Transformer [6]) is an encoder-decoder model that converts all NLP problems into a text-to-text format. It is trained using teacher forcing, which means that for training, we always need an input sequence and a corresponding target sequence. The input sequence is fed to the model using `input_ids`. The target sequence is shifted to the right by being prepended with a start-sequence token and is fed to the decoder using `decoder_input_ids`. In teacher-forcing style, the target sequence is then appended with the EOS (end-of-sequence) token and corresponds to the labels. However, it's important to note that the PAD token is not used as the start-sequence token. Instead, a separate token (typically designated as a special token like <s> or similar) is used to signify the start of a sequence. The PAD token is used to fill out sequences for batching purposes so that all sequences in a batch have the same length.

Phase 1: Rule-Based Preprocessing

- Remove sentence separation using the Hungarian tokenizer Quntoken[1].
- Remove newline characters.
- Tokenize sentences using huSpaCy and apply some filtering criteria:
  1. Filter sentences based on the number of tokens ($8 < \text{token\_count} \leq 500$). This step is based on the observation that sentences shorten than 8 tokens usually contain only little information; on the other hand, the maximum number of tokens is specified as 500 because of the `max_token` value of the model (512).
  2. Filter Languages other than Hungarian (only keep sentences detected as Hungarian).
  3. Exclude sentences containing only digits.
  4. Filter sentences with numbers + special character to letter ratio exceeding 0.2.
  5. Exclude sentences with words longer than 30 characters.
  6. Replace commas within quotes in each sentence – One of the most common OCR errors in Hungarian is that the quotation mark characters („) are recognised by the OCR software as double commas, so we replace these by a rule-based approach where necessary.
  7. Remove spaces before punctuation.

Phase 2: Sentence Pairing Based on Similarity
We match the original and Tesseract sentences based on similarity calculated using the NYTK/sentence-transformers-experimental-hubert-hungarian

[1] https://github.com/nytud/quntoken).

Sentence Transformer model, the huSpacy hu_core_news_lg model, and the Python difflib SequenceMatcher algorithm. Sentences are classified as error-free if all three similarities equal 1.0. During pairing, only sentences with a specified threshold similarity value are included in the database, avoiding the inclusion of sentence pairs with similar meanings but different syntax. This method increased the database by 451,820 sentence pairs.

4.3 Synthetic Data

In the process of creating the Gold Standard Corpus, Laki et al. [3] conducted a comprehensive error analysis, identifying 8,593 distinct OCR error types with the assistance of human annotators. This analysis provided insight into the frequency of various OCR errors. Using these findings, we developed a tool capable of generating synthetic corpora of practically unlimited size. This tool simulates OCR errors by replacing random characters with corresponding OCR erroneous pairs and by inserting or deleting characters, while throughout keeping to the observed frequency of OCR errors in the error-free texts of scanned newspapers. As a result, our training database was augmented with an additional 257,665 lines, significantly improving the diversity and representativeness of our training data.

Figure 1 shows the proportion of the above data sources in the training dataset.

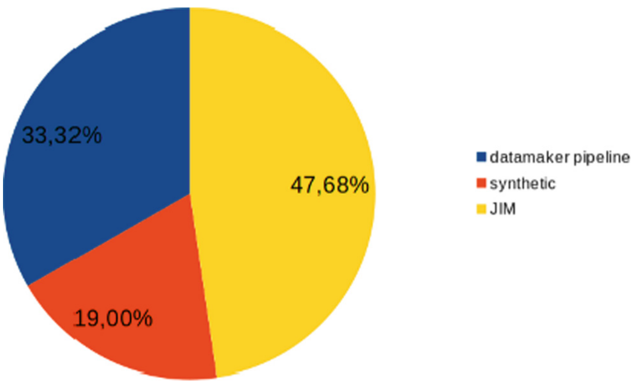

Fig. 1. Proportion of the data in the training dataset

4.4 The Gold Standard Corpus

In parallel with the training and testing of the models we also started the development of a gold standard corpus, which involved a thorough annotation process to ensure that the textual data closely mirrored the original PDFs from which it was derived. This process was rooted in texts extracted from random sections of files from the books of two major publishing companies, covering a wide range of domains. The final corpus contains 100,000 lines, with each line representing a paragraph from the original text, potentially comprising multiple sentences or occasionally being an empty line for structural purposes.

In the first phase of annotation, annotators were tasked with comparing the content of a given PDF to its text (.txt) version created from the PDF, adjusting the text to match the PDF exactly. This step involved two annotators reviewing and annotating each file independently to ensure thoroughness. Subsequently, their outputs were merged to form a single, finalized version of the text. This rigorous process was guided by key principles designed to retain the original formatting and errors present in the PDFs, excluding page numbers and ensuring correct text structuring, such as maintaining paragraph integrity, differentiating between document sections with double line breaks, and accurately representing dialogue, content lists, images, tables, and footnotes as per specific guidelines.

The annotation principles emphasized the importance of character-level fidelity to the PDF content, even preserving typographical errors. Modifications excluded page numbers and end-of-line hyphenations unless they contributed to the meaning or structure. Text structuring guidelines were strictly followed, including spacing around titles and paragraphs, separation of documents within a volume, and the handling of dialogue units, content lists, images, and tables with appropriate placeholders. Special characters were replaced with their Unicode equivalents, and footnotes were tagged accurately, ensuring that they reflected their placement in the PDF. This detailed approach resulted in a corpus that, while preserving the essence and layout of the original documents, facilitated easier handling and processing for research purposes.

5 The Training Method

The training data was randomly partitioned into two sets: 90% for training and 10% for testing. We fine-tuned the `google/mt5-large` model using the HuggingFace transformers library on a single NVIDIA A100 SXM4 80 GB GPU, executing the training for a total of 38,137 steps, which corresponds to approximately one epoch. During training, we employed a Linear Warmup strategy for the learning rate. The model was configured to handle a maximum token sequence length of 128 for both input and output, with a batch size set to 32. The fine tuning took 27 h 8 min.

6 Results and Discussion

This section presents the evaluation of our OCR correction model. We assess the model's performance using several metrics: Word Error Rate (WER), ROUGE-L score, and the identification of perfect matches in OCR erroneous sentences. Additionally, we analyze the model's capability to differentiate between erroneous and non-erroneous sentences. The evaluation was carried out by comparing the errors identified in the original text with the errors identified by the model

Fig. 2. Model training performance metrics over iterations. The three plots represent the changes in evaluation loss, training loss, and learning rate against the number of steps taken during the training phase of the model.

in relation to the "target" (error-free) sentences. The test database contains a wide range of texts, from academic works to literature and newspaper articles (Fig. 2).

6.1 Metric Definitions

Before delving into the results, we define the metrics used for evaluation:

- WER (Word Error Rate): Measures the proportion of incorrect words to the total words in the reference text, lower values indicate better performance.
- ROUGE-L: Reflects the overlap of n-grams between the system output and reference texts, with higher scores indicating better quality.
- OCR Erroneous Sentences: Sentences identified by the model as containing OCR errors.
- Perfect Matches: Instances where the corrected text exactly matches the reference text.

6.2 The SOTA

Laki et al.'s [3] mT5 scored 0.923515 ROUGE-L on the test set (the same test set we used for the new model). The overall WER after correction was 0.224. (from 0.2327 = 0.9% improvement) Out of the 4799 sentences with OCR errors in the test set, only 198 have a perfect match between the corrected and the target sentence (4.13%). Their model incorrectly identified 60 out of 1981 non-erroneous sentences as erroneous, resulting in a false-positive rate of 2.98%.

6.3 Performance Improvement

Our model demonstrates significant improvements in text correction accuracy, as evidenced by the metrics:

- The overall WER improved from 0.2327 to 0.1814, marking a 5.1% enhancement in the OCR erroneous sentences.
- For the entire test data, the improvement in WER is 0.148, amounting to a 6.5% improvement.

- The mean ROUGE-L score increased from 0.90 to 0.94 for OCR erroneous sentences, relative to the reference sentences.
- Out of 4799 OCR erroneous sentence pairs, 1095 were perfect matches after correction, achieving a 22.82% success rate.
- The model incorrectly identified 59 out of 1981 non-erroneous sentences as erroneous, resulting in a false-positive rate of 2.97%.

The observed improvements in WER and ROUGE-L scores highlight the effectiveness of our model in correcting OCR-generated text errors. The significant percentage of perfect matches further demonstrates the model's accuracy in identifying and correcting errors. However, the false-positive rate indicates a need for refinement in distinguishing between erroneous and non-erroneous sentences, suggesting an area for future work.

7 Conclusion

In our research aimed at correcting OCR errors, we efficiently employed the mT5 model, leveraging its Text2Text machine translation capabilities. We explored both mT5-small and mT5-large variants during the model's fine-tuning process. The outcomes suggest that the mT5 model is notably efficient in rectifying texts with OCR errors. We anticipate that improvement in the training dataset and the use of larger model variants could further improve correction accuracy. Additionally, we generated synthetic data to emulate OCR errors, thereby enriching our training dataset. The experimental results affirm the mT5 model's effectiveness in OCR error correction, highlighting the potential for achieving superior performance with ongoing advancements. Our review of relevant literature and international studies suggests that integrating character-based and sequence-to-sequence correction techniques could yield higher accuracy and reduce the likelihood of erroneous corrections. Moreover, the strategic application of Large Language Models (LLMs) in the detection, correction, and verification phases presents a promising direction for future research. The insertion of our recently developed gold standard corpus into the training data could also improve our results.

Acknowledgments. The present research was conducted with the support of the Hungarian Academy of Sciences in the framework of the Natonal Program 'Science in support of the Hungarian Language'.

References

1. Amrhein, C.: Post-correcting OCR errors using neural machine translation. Ph.D. thesis, Universität Zürich (2017). https://api.semanticscholar.org/CorpusID:231696696
2. Gupta, H., Del Corro, L., Broscheit, S., Hoffart, J., Brenner, E.: Unsupervised multi-view post-OCR error correction with language models. In: Moens, M.F., Huang, X., Specia, L., Yih, S.W.T. (eds.) Proceedings of the 2021 Conference on Empirical

Methods in Natural Language Processing, pp. 8647–8652. Association for Computational Linguistics, Online and Punta Cana (2021). https://doi.org/10.18653/v1/2021.emnlp-main.680, https://aclanthology.org/2021.emnlp-main.680
3. Laki, L.J., et al.: OCR hibák javítása neurális technológiák segítségével [Correction of OCR errors using neural technologies]. In: XVIII. Magyar Számítógépes Nyelvészeti Konferencia, pp. 417–430. Szegedi Tudományegyetem, Informatikai Intézet, Szeged (2022). Original text in Hungarian
4. Maheshwari, A., Singh, N., Krishna, A., Ramakrishnan, G.: A benchmark and dataset for post-OCR text correction in Sanskrit. In: Findings of the Association for Computational Linguistics: EMNLP 2022, pp. 6287–6294. Association for Computational Linguistics (2022). https://doi.org/10.18653/v1/2022.findings-emnlp.527, https://aclanthology.org/2022.findings-emnlp.527
5. Piotrowski, M.: Post-correction of OCR results using pre-trained language model (2021). http://poleval.pl/files/2021/09.pdf. Presentation slides
6. Raffel, C., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
7. Rigaud, C., Doucet, A., Coustaty, M., Moreux, J.P.: ICDAR 2019 competition on post-OCR text correction. In: 2019 International Conference on Document Analysis and Recognition (ICDAR), pp. 1588–1593 (2019). https://doi.org/10.1109/ICDAR.2019.00255
8. Schaefer, R., Neudecke, C.: A two-step approach for automatic OCR post-correction. In: Proceedings of the Workshop on Computational Humanities Research (LaTeCH-CLfL 2020), pp. 52–57. Association for Computational Linguistics (2020)

Identifying and Leveraging Research Software

RTaC: A Generalized Framework for Tooling

Nisarg Bhavsar(✉), Abhinav Thakur, Amrit Lal Singh, and Ashish Patwa

Indian Institute of Technology Kharagpur, Kharagpur 721302, India
nisargbhavsar25@kgpian.iitkgp.ac.in
https://www.iitkgp.ac.in

Abstract. In the rapidly evolving domain of Large Language Models (LLMs), integrating tool usage remains a formidable challenge, particularly when it comes to the dynamic selection and sequencing of tools in response to complex queries. Addressing this, we introduce Reimagining Tooling as Coding (RTaC), a groundbreaking framework that transforms tool usage into a coding paradigm. Inspired by recent advancements [18], RTaC conceptualizes tools as Python functions within a dual-agent system [2], significantly enhancing LLMs' tool usage efficiency. Our comprehensive experiments reveal that RTaC enables coding-based LLMs, such as DeepSeek and CodeLlama, to achieve and surpass GPT-4 benchmarks in cost-effectiveness and latency without compromising on handling intricate tool sequencing with conditional and iterative logic. This research not only sets a new benchmark for tooling efficiency in LLMs but also opens new avenues for the application of LLMs in complex problem-solving scenarios, heralding a significant leap forward in the functionality and versatility of LLMs across diverse domains.

Keywords: Large Language Models (LLMs) · Dual Agent System · Python Functions for Tool Integration · Automated Tool Sequencing · Advanced LLM Applications · RTaC Framework

1 Introduction

In the evolving landscape of LLMs, their use as reasoning and tooling agents has garnered substantial attention. LLMs demonstrate the capacity to interpret and respond to queries by calling tools [11,12], a testament to their advanced language comprehension. This capability to integrate tool usage represents a significant stride in enhancing the scope and accuracy of LLMs in various applications. Current state-of-the-art approaches to the tool-usage problem, which utilize GPT-4 (OpenAI) and Claude-2 (Anthropic), demonstrate impressive results but are closed-source and computationally expensive. Researchers have attempted to solve this problem by fine-tuning smaller language models [11,12,15]. However, these models are ineffective at generalizing to new tools when provided in a

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 61–79, 2024.
https://doi.org/10.1007/978-3-031-65794-8_5

zero-shot manner, referred to as 'dynamic tooling' from here onwards. The discrepancy between the generalized tool-use capabilities of large models and the more restricted capabilities of compact models presents the motivation behind our work - Can we exploit the nature of this task to train small open-source LLMs to generalize their tool-use abilities while keeping the latency minimal? (Fig. 1).

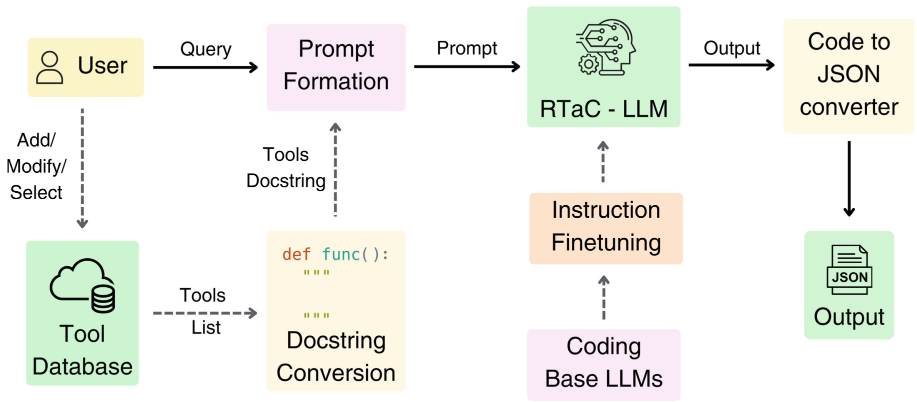

Fig. 1. Overview of "Reimagining Tooling as Coding" (RTaC)

Addressing these challenges, we propose Reimagining Tooling as Coding (RTaC), which reconceptualizes tooling as a code-generation task to exploit the powerful code-comprehension capabilities of LLMs. RTaC provides tools to be used, in docstring format, to instruct fine-tuned coding-base LLMs. It then extracts the output in Python-inspired code format and deterministically converts it to JSON. RTaC promotes docstring reading capability in the LLMs, supporting tool modification, addition, and deletion. We use RTaC to achieve GPT-4 benchmark performance while employing smaller models, such as DeepSeek 1.3B and CodeLlama 7B LLMs, despite a drastic (300x) reduction in parameter count, as shown in Sect. 5. We simultaneously achieve significant (5x) cost reduction per query while matching GPT-4's latency. Moreover, RTaC supports processing complex conditional and iterative logic, surpassing GPT-4's capabilities.

2 Related Works

2.1 Dataset and Tooling Benchmarks

Various domain-specific tooling datasets have been proposed like API-Bank [7], ToolEyes [20], RoT-Bench [21], EasyTool [22] and MetaTool [5]. These are domain-specific and assess LLMs' tool usage and tool-identifying abilities.

- **API-Bank:** Developed from interviews with over 500 users, this benchmark includes a training set created through a multi-agent approach and a diverse

set of manually annotated dialogues to assess LLMs' API usage across various domains and complexities.
- **ToolEyes:** Features a comprehensive evaluation system with 600+ tools across 7 scenarios, assessing LLMs on five critical dimensions to expose capability gaps and generate research insights.
- **RoT-Bench:** Evaluates LLMs' ability to accurately select tools, identify parameters, and fill content in environments ranging from noise-free to highly variable real-world conditions.
- **EasyTool:** Addresses issues from inconsistent documentation by creating standardized tool instructions to improve LLMs' tool usage proficiency.
- **MetaTool:** Assesses LLMs' tool selection awareness and suitability across various tasks and scenarios, highlighting biases and current limitations.

These benchmarks are domain-specific and unrelated to tool usage as a function-calling approach. We thereby went on to build our dataset for the task using a dual agent system and test our approach on it.

2.2 Tooling LLMs

The application of LLMs for tooling is a profound task, and various research, as mentioned in TALM [10], which uses tools in context to solve different tasks; LATM [2] showed that LLMs can be used to create and reuse different tools created by them in order to act as intelligent Agents. Various tooling LLMs like Tool LLaMA, ToolAlpaca [15], and Gorilla [11] are available and suitable for use as domain-specific agents. However, they are captivated by the out-of-domain tool usage capabilities, identifying the correct set of tools and assigning appropriate arguments to them. Tool Llama is a fine-tuned version of Llama-70B on the ToolBench Dataset, as mentioned in the paper [12]. The model works well on general domain tools but fails in context-dependent scenarios that use tools. ToolAlpaca is a generalized tool LLM that adapts off-domain tools for usage. It is still hard for the LLM to reason on complex tool ordering scenarios.

Recent approaches have aimed to augment LLMs with the ability to utilize tools and resources. TALM [10] introduces a framework for integrating tools with LLMs like T5 via a text-to-text API, enabling generalization to out-of-distribution inputs solvable with access to tools. It employs a policy-gradient reinforcement learning algorithm to fine-tune the LLM for tool usage. The Hugging-GPT system [14] leverages the Hugging Face API to solve AI tasks using LLMs. Toolformer [13] is an LLM pre-trained on an annotated dataset, exhibiting prowess in solving complex problems by leveraging external APIs. However, it is constrained by a fixed set of available tools and the inability to chain tool usage. Toolformer also implements novel self-supervised augmentation during training. These approaches demonstrate the potential of enhancing LLMs with tool utilization capabilities while highlighting challenges such as generalization, tool chaining, and scalability to new tools.

Gorilla [11] stands out as a pivotal work that uses the LLAMA-7B model to accurately extract APIs from repositories like TensorHub, HuggingFace, and

TorchHub. It significantly outperforms GPT-4 in API functionality accuracy and reduces hallucination errors, emphasizing enhancing LLMs' practical utility over conversational skills. Evaluated on an extensive dataset of 11,000 API pairs, Gorilla demonstrates commendable retrieval capabilities. However, Its reliance on machine learning datasets and the necessity for fine-tuning raise questions about generalizability and adaptability to custom APIs. Complementing Gorilla, ToolBench [12] is a large-scale benchmark containing over 16,000 high-quality APIs across 3,451 tools, facilitating robust evaluation of LLMs' API usage skills. It employs a 3-stage-construction process, comprehensive API metadata, and diverse instruction generation. GPT-3.5 searches for valid action sequences using accurate API responses through multi-round conversations, leveraging a Depth-First Search Decision Tree (DFSDT) to expand the search space. With its scale, diversity, realism, and expanded search methodology, ToolBench enables a thorough assessment of LLMs' capabilities in utilizing APIs to accomplish tasks.

Ultimately, we look at Tool Alpaca, a framework to improve compact language models' generalized tool usage skills. It first constructs a diverse corpus spanning 50 categories and 426 tools with 3938 usage instances generated via multi-agent simulation. This corpus is then used to fine-tune compact Vicuna models, creating ToolAlpaca-7B and 13B. Experiments on unseen simulated and real-world tools demonstrate that ToolAlpaca models achieve strong generalization comparable to large models like GPT-3.5. Tool diversity is shown to be critical, with performance improving as the variety of tools in the corpus increases. Overall, ToolAlpaca provides an automated approach using simulation and diversity to instill generalized tool usage abilities in compact models, enabling them to adapt to new tools.

All these approaches work well, but they depend on the core tools on which they are trained and fine-tuned. Out-of-domain tool usage is a difficult task for all of these models.

2.3 Prompting Methods

Prompting is also a significant method to improve the LLM context adherence capability and can also be used in the region of agentic LLMs. Various prompting methods, such as Chain-of-Thought [17], Tree-of-Thoughts [19], Graph-of-Thought [1], Skeleton-of-Thought [9], and Knowledge Graph [3] addition, can increase a LLM's overall context understanding capability. One more technique to look at is multi-tool COT Prompting,

Recent research has proposed several innovative frameworks to improve the multi-step reasoning capabilities of LLMs by combining chain-of-thought (CoT) prompting techniques with external tool integration. The "Tree of Thoughts" (ToT) [19] approach frames problem-solving as a search through a tree structure, where each node represents a coherent "thought" or intermediate reasoning step. ToT allows LLMs to explore multiple reasoning paths, generate and evaluate candidate thoughts, and direct the exploration using classical search algorithms like breadth-first and depth-first search. The "Graph of Thoughts" (GoT) [1] framework represents information as an interconnected graph, with vertices denoting individual pieces of information and connections signifying dependencies between them. This flexible structure enables GoT to integrate outputs from various reasoning paths, analyze complex thought networks, and incorporate feedback loops for iterative improvement of LLM outputs.

Furthermore, the MultiTool-CoT [6] framework facilitates LLMs like GPT-3.5 to leverage multiple tools, such as calculators and knowledge retrievers, by inserting triggers for tool invocation at appropriate steps within the CoT reasoning process. Experiments on numerical and knowledge reasoning datasets demonstrate that MultiTool-CoT significantly outperforms baselines, achieving state-of-the-art accuracy by addressing different error types with different tools, with gains from combining tools exceeding individual tool gains.

These approaches have significantly improved over standard prompting techniques across various tasks, including mathematical reasoning, creative writing, and knowledge-based problems. However, challenges persist, such as token limitations for CoT prompting and potential errors in LLM-generated reasoning processes, highlighting the need for further research in this area to unlock the potential of LLMs in complex problem-solving scenarios fully.

3 Method

3.1 RTaC (Reimagining Tooling as Coding)

RTaC is a novel framework that proposes the conversion of tools into Python functions with proper arguments and tool descriptions in the form of a Pythonic tool docstring, which can be appended to the context of the prompt in order to select and sequence the correct tools, with proper arguments. The paper [18] inspires the technique, which discusses how we can empower the capabilities of an LLM using code. The framework involves the creation of the dataset using a dual agent system that generates query output pairs. These pairs are used for instruction fine-tuning various Coding base LLMs, which act as tooling agents. The application of coding-based LLMs helps adhere to various complex conditional and iterative logics in tooling. Our experiments prove that open-source coding base LLMs are better regarding latency and cost per query than benchmark GPT-4. Coding base LLMs concerning normal LLMs perform better due to their fewer hallucinations and higher context adherence ability.

Dataset Generation. The papers above incorporate data generation as their primary approach for adapting base LLMs for tool usage. Gorilla introduces a comprehensive dataset called APIBench by utilizing Self-Instruct [16], which proposes an automated pipeline to generate large-scale instruction datasets from a small set of seed tasks. First, human experts provide sample instructions and API documentation as context. A language model generates new instructions that plausibly use the APIs, creating instruction-API pairs. A vital benefit of this approach is that it does not require manual effort to label training data. Gorilla uses GPT-4 for this data generation.

The RTaC framework employs distinct datasets to rigorously evaluate its performance. Specifically, the datasets are structured as follows: the Static ToolSet comprises 800 query-output pairs, the Dynamic ToolSet includes 700 pairs, and the Conditional/Iterative ToolSet consists of 300 pairs. Additionally, 200 unanswerable philosophical queries are incorporated to test the model's robustness in handling queries beyond its configured capabilities (Fig. 2).

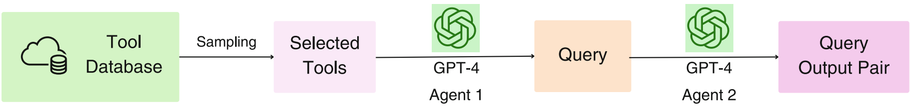

Fig. 2. Dual agent dataset generation

Although APIBench is built over massive APIs, it does not have multi-tool scenarios. ToolLLM proposes an innovative data generation strategy supporting multi-tool interplay-the paper samples API combinations by iterating through tools and sampling intra-category and intra-collection combinations. GPT-3.5 is leveraged to generate instructions involving the sampled APIs, and its behavior is regulated by prompting with documentation, task descriptions, and examples. Generated APIs are validated against the original sample to filter out hallucinations.

Static ToolSet. The Generation of the static toolset was done using a set of initial pre-defined tools as shown in Appendix A.3. As shown in the figure, the query output pairs are generated using an agent fed with different tools. As shown below, query templates are generated, which are then randomly filled by GPT-4, and the solution APIs for those are also generated using the GPT-4 Model. Then, the dataset underwent a thorough human evaluation process, which we used for instruction fine-tuning.

Dynamic ToolSet. The Dynamic toolset, as the name Dynamic suggests, is not pre-defined, and they are generated in the correct format using a different agent. The Dynamic toolset is generated using a procedure when the tools are generated using one agent. Then, we again used GPT-4 as the agent to generate query templates, fill those query templates, and get the relevant set of tool APIs for those queries. These queries then undergo a rigorous human evaluation for syntactic and logical correctness.

Conditional and Iterative ToolSet. Utilizing various tools with conditional and iterative logic requires the creation of particular query output pairs where we use conditional logic like if-else and iterative logic like loops. These kinds of queries are complex to handle by normal LLMs, and they require fine-tuning and an innate logic formation ability present in a coding-based LLM. The query output pair was again generated through GPT-4, and the query templates were filled out using the agent by random values, and then the response was generated. Again, these queries went through a robust human evaluation.

A dataset containing 1800 Query Output pairs was generated with static and dynamic toolsets, and 200 unanswerable philosophical queries were generated; a total of 2000 Query Output pairs were named Stage 1 Dataset. Additionally, 100 query output pairs were generated with conditional and iterative logic, and this was combined with Stage 1 datasets, which were randomly selected to be 500 pool. This dataset was named Stage 2 Dataset. This Whole Dataset was used to do instruction fine-tuning on various LLMs.

Design Framework. Reimagining tooling as a form of coding in the context of LLMs forms the cornerstone of our pipeline design. This approach stems from the observation that tool utilization in LLMs essentially involves executing function calls, assigning values to arguments, and efficiently linking these outputs, mirroring the core elements of coding. This conceptual overlap extends beyond mere theory, as evidenced by the proficiency of Copilot and CodeGen. Grounded in this insight, we adopt a training strategy that treats the tooling challenge within the framework of a coding paradigm. Accordingly, we prioritize fine-tuning LLMs with a foundational background in coding (such as DeepSeek-1.3B Code-Instruct) instead of those exclusively trained on natural language processing tasks.

In our paradigm, tool descriptions are conveyed to LLMs in a docstring format during training, as shown in Fig. 3 (left), emulating standard coding practices. The expected output format is structured as variable assignments from API calls (e.g., var_x = api_call (arguments)), as shown in Fig. 3 (right). This format offers advantages over direct training on JSON outputs by reducing the number of output tokens required and circumventing the need for additional training to correct JSON errors, an issue prevalent in other methods [11,12].

```
def add_work_items_to_sprint(work_ids, sprint_id):
    '''
    Adds the given work items to the sprint

    Parameters:
        work_ids (list): A list of work item IDs
to be added to the sprint
        sprint_id (str): The ID of the sprint to
which the work items should be added
    '''
```

```
# Prioritise my P0 issues and add them to the current
sprint

var_1 = who_am_i()
var_2 = works_list(issue.priority="P0", owned_by=var_1)
var_3 = prioritize_objects(objects=var_2)
var_4 = get_sprint_id()
var_5 = add_work_items_to_sprint(work_ids=var_3,
sprint_id=var_4)
```

Fig. 3. Sample of tool docstring (left) and output in code format (right)

Conditional and iterative logic is handled by allowing the LLM to generate outputs in the format of var_x = api_calls () and incorporate if-else statements and for-loop constructs. In the parsed JSON object, we introduce a specialized magic tools - conditional_magic with the capability for 'JSON in JSON' style argument values, as shown in Fig. 4. Such a format is crucial for managing multiple chain tools dependent on specific conditions or requiring iterative processes.

```python
# if current sprint is "SPR-123", ring
bell. after that, archive all p0 issues
created by user "USR-123".


var_1 = get_sprint_id()

if var_1 == "SPR-123":
    temp_1 = ring_bell()

var_2 = works_list(issue.priority=["p0"],
created_by=["USR-123"], type=["issue"])

for loop_var in var_2:
    temp_2 = archive_item(work_id=loop_var)
```

```
{
  "tool_name": "get_sprint_id",
  "arguments": []
},
{
  "tool_name": "conditional_magic",
  "condition": "$$PREV[0] == \"SPR-123\"",
  "true": [
    {
      "tool_name": "ring_bell",
      "arguments": []
    }
  ],
  "false": []
},
...
{
  "tool_name": "iterational_magic",
  "looping_var": "$$PREV[2]",
  "loop": [
    {
      "tool_name": "archive_item",
      "arguments": [
        {
          "argument_name": "work_id",
          "argument_value": "$$LOOP_VAR"
        }
    ...
```

Fig. 4. Sample code output and conversion using 'JSON in JSON' methodology

Fine Tuning Methods. We propose fine-tuning our LLM using the data generated by the earlier methodology to achieve both input format comprehension and output format adherence. This fine-tuning utilizes Stage 1 and 2 datasets to instill in the LLM docstring comprehension and adherence to our Python-inspired output format.

Training Pipeline: We follow an instruction fine-tuning-based training approach wherein we first prepare our dataset in a structure where an "Allowed tools": token is introduced, followed by the docstrings for the tools to be used as shown in Appendix A.2. We train the model on the LORA [4] framework using the PEFT library under a 4-bit quantization setting through the Bits and Bytes framework. Training is done in two stages. During the first stage, the model is trained for five epochs on queries from the Stage 1 Dataset and the docstrings for the tools in the Problem Statement. The model is further trained for five epochs using the Stage 2 Dataset, where it sees the docstrings for the tools in the Problem Statement and the five new tools described above. This short instruction fine-tuning instills docstring reading capabilities in the LLM and adherence to our Python-inspired code output format.

Inference: RTaC allows the user to add their own set of dynamic tools. These tools are appended to the static tools in the prompt under the "Allowed Tools": token and then passed to the LLM. Similarly, updating docstrings are passed under the "Allowed Tools": token in modifying and deleting already added tools.

3.2 Evaluation Metrics

BLEU score focuses solely on n-gram precision and may not accurately reflect semantic similarity; hence, it is not an optimal metric for evaluating the

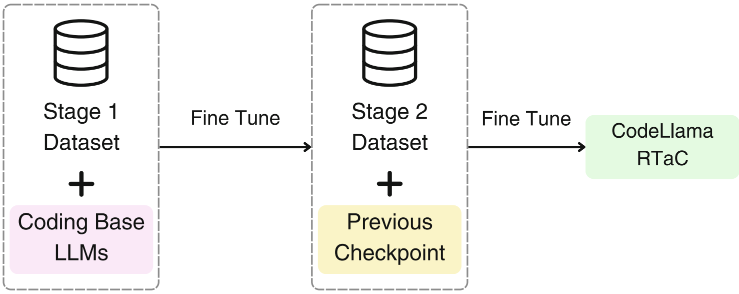

Fig. 5. RTaC Training Pipeline

performance of our approach. Other metrics, such as the JSON Similarity Score and the F1-Score, were used to evaluate the final approach (Fig. 5).

JSON Similarity Score: The JSON Similarity Score is a metric used to quantify the degree of similarity between two JSON objects. It measures how closely the structure and content of two JSON objects align, aiding in data comparison, deduplication, and schema matching. Analyzing key-value pairs, arrays, and nested structures provides insights into the level of resemblance between datasets. This score is precious in data integration processes, ensuring consistency and accuracy across disparate sources. Its ability to assess JSON efficiently facilitates seamless data exchange and interoperability in diverse applications. We use this to assess the correctness of our generated tool structure.

F1-Score: In our evaluation framework, the F1-Score is calculated based on the precision (the proportion of relevant instances among the retrieved instances) and recall (the proportion of relevant instances that were retrieved). The true positives are defined as the set of tools correctly identified and utilized by the LLM, while false positives are those incorrectly predicted tools, and false negatives are correct tools that the model failed to identify. This metric is particularly suited for assessing the accuracy of tool selection and usage in complex LLM operations.

4 Experiments

We have conducted various experiments on open-source coding base, normal, and Closed Source LLMs. The experiments ranged from using retrievers to various prompting methods and fine-tuning approaches on different stage datasets. The Experiments were done on Google colab, where fine-tuning was done on A-100 GPUs, and inference was done on T4 and A100 GPUs. Within the Regime of open source LLMs, we experimented with Llama-7b, Zephyr-7b, Codellama-7b, Codellama-13B, Deepseek-6.7b, Deepseek-1.3B, Toollama, ToolAlpaca. We also explored closed-source LLMs like GPT-4 and GPT-3.5 for closed-source benchmarking. We also compare the approach with a retriever to reduce the tool selection set.

4.1 Retrievers

Our static API toolset consists of only 9 APIs. Tool retrievers can help decrease the tool descriptions that need to be passed to the LLM, thereby decreasing context length and latency. We compare various retrievers like BM25, FAISS, Ensemble, and DPR. Further, tool retrievers become essential for stability when our pipeline is applied to bigger toolsets. We experiment with multiple tool retrievers, considering each API-argument_name pair as a new tool.

Table 1. Retriever Results

Method	Recall		F1-Score		Time (ms)	
	Top-10	Top-15	Top-10	Top-15	Top-10	Top-15
BM25	0.73	0.83	0.41	0.35	47.8	71.8
FAISS	0.83	0.90	0.46	0.37	758	983
Ensemble (BM25 + FAISS)	0.92	0.95	0.41	0.34	954	1110
Dense Passage Retriever	0.70	0.78	0.40	0.32	–	–

As shown in Table 1, retrievers do not work very well in identifying the correct set of tools, and they also fail to identify the appropriate set of arguments for the tools.

4.2 Closed Source LLMs

Prompting Methods. We experimented with various prompting methods, which include zero-shot prompting, few-shot prompting, chain-of-thought prompting, graph-of-thought prompting, tree-of-thought prompting, and knowledge graph-infused prompting and RTaC prompting, which is our flagship method, in which tools are presented in the form of functions and converted into docstring to be added in the final prompt for solving the query.

Our Experiments on prompting with open-source LLMs like llama2-7b and zephyr-7b-beta showed that both the models hallucinate and perform poorly on our metrics and human evaluation. The JSON similarity score between the actual output and the predicted output was less than 0.1, and the F1-score was less than 0.5 in both cases of zero-shot prompting and few-shot prompting for both the models, which showed that it is not the correct way to evaluate open source models with closed source models.

4.3 Open Source LLMs

We have performed various experiments on tooling LLMs like Toollama, Toolpaca, Codellama, Codegen, and Deepseek. Our Experiments range from few-shot

Table 2. Results - Closed-Source LLMs

Model	JSON Similarity	Precision	Recall	F1-Score
GPT-3.5	67.23	82.34	87.32	84.76
GPT-4 Turbo	74.88	88.23	85.45	86.82
GPT-4 Turbo + SoT	75.23	84.52	89.43	86.91
GPT-4 Turbo + GoT	82.32	87.69	90.16	88.91
GPT-4 Turbo + CoT	79.11	88.12	85.84	86.97
GPT-4 Turbo + ToT	80.69	86.52	88.62	87.56
GPT-4 Turbo + KG	80.62	84.32	90.97	87.52
GPT-4 Turbo + RTaC	87.79	92.12	95.81	93.93

prompting to fine-tuning. We present three pipelines for our Problem Statement that utilize open-source LLMs, incrementally building upon our hypothesis of "Reimagining Tooling as Coding". This experimentation helps us arrive at our proposed pipeline, RTaC and also serves as an ablation study (Table 2).

Few-Shot prompting of Coding-Base LLMs. Using this pipeline, we investigate our hypothesis around the efficiency of Coding-Base LLMs over normal LLMs. The LLMs are provided with a prompt similar to that referred to in Sect. 4.1 with few-shot examples involving both static tool usage (5 examples) and conditional, iterative tool (3 examples) (Table 3)

Table 3. Results - Few-Shot prompting of Coding-Base LLMs

Dataset	Model	JSON Similarity	Precision	Recall	F1-Score
Static	CodeLlama 7B	68.78	81.25	77.85	79.51
	CodeLlama 13B	73.47	82.80	85.45	86.82
	DeepSeek 1.3B	60.28	73.33	63.00	67.77
	DeepSeek 6.7B	62.02	90.72	61.48	73.29
Dynamic	CodeLlama 7B	67.94	78.24	74.97	76.57
	CodeLlama 13B	71.12	79.84	85.41	82.53
	DeepSeek 1.3B	60.82	65.29	66.53	65.90
	DeepSeek 6.7B	63.96	92.10	57.53	70.82
Bonus	CodeLlama 7B	56.28	76.43	74.17	75.28
	CodeLlama 13B	59.22	78.13	83.91	80.87
	DeepSeek 1.3B	47.32	63.49	62.18	62.83
	DeepSeek 6.7B	54.87	87.56	59.25	70.68

Discussion: Few-shot prompting on pre-trained LLMs like Llama-2 7B and Zephyr 7B (results provided in Table 4) is substantially surpassed by CodeLlama and DeepSeek, proving our choice for Coding-Base LLMs. However, this pipeline is plagued by a high context length needed to explain the output formats and chatbot behavior, which further leads to higher latencies. Output inspection reveals that while models correctly tend to solve the query, the format is non-convertible, which is necessary for evaluation.

Table 4. Results - Few-Shot prompting of Llama-2 7B and Zephyr 7B

Dataset	Model	JSON Similarity	Precision	Recall	F1-Score
Static	Llama-2 7B Chat	44.57	40.80	41.95	41.37
	Zephyr 7B Chat	50.74	61.58	79.74	69.49
Dynamic	Llama-2 7B Chat	37.94	35.18	38.17	36.61
	Zephyr 7B Chat	41.12	55.39	47.67	51.24

Tool Memorization with "Add Tool" Token. We build upon ToolLlama style fine-tuning of LLMs, which fine-tunes LLMs using query-output pairs to include support for dynamic tooling. To achieve we use the "Added Tools": token. Dynamic tools provided at runtime are appended in docstring format after this token, while the query follows the "Query": token in the input prompt. To instill an understanding of our added tokens, we first generate 50 dynamic tools and queries that interface with them using the Self-Instruct [16] methodology. As described above, three hundred such queries-output-toolset tuples, the 100 bonus query-output pairs, and the Stage 1 Dataset are used for instruction fine-tuning over ten epochs.

Table 5. Results - Tool Memorization with "Add Tool" token

Dataset	Model	JSON Similarity	Precision	Recall	F1-Score
Static	CodeLlama 7B	85.89	92.36	94.31	93.32
	ToolApaca	68.45	81.51	73.85	77.49
	ToolLlama	69.57	86.22	78.13	81.98
	DeepSeek 1.3B	84.68	91.94	94.63	93.27
Dynamic	CodeLlama 7B	75.51	85.45	87.51	86.47
	ToolApaca	63.47	75.93	72.27	74.05
	ToolLlama	66.62	78.24	75.11	76.64
	DeepSeek 1.3B	75.83	82.12	81.85	81.98
Bonus	CodeLlama 7B	83.91	87.67	91.12	89.36
	ToolApaca	67.22	80.33	72.91	76.44
	ToolLlama	68.34	84.96	76.19	80.34
	DeepSeek 1.3B	81.35	89.11	90.71	89.90

Discussion: As shown in Table 5, Experimentations with this pipeline reveal training instability. While more extended training makes the model excel in the static setting, dynamic tool comprehension and usage take a hit. On the other hand, enabling dynamic tooling with controlled training length leads to parameter and data type hallucinations for the memorized static tools. Memorization further limits this pipeline's ability to modify and delete tools. All this motivates moving to a pipeline with the least tool memorization. Instruction fine-tuning shows promising adherence to a code output format that is convertible to JSON.

RTaC (Our Proposed Pipeline). Here, we build upon the previous pipeline by replacing the "Added Tools": token with the "Allowed Tools": token and appending docstrings for all tools, both static and those added dynamically at runtime, after the token. This pipeline avoids tool memorization and instead promotes docstring comprehension.

In the RTaC framework, the "Allowed Tools": token is used to specify the complete set of tools (static, dynamic, and conditional/iterative) that the LLM can utilize dynamically across different queries. This inclusive approach enables the model to adapt its tool usage flexibly depending on the query context, enhancing its applicability and efficiency. Conversely, the "Added Tools": token implies a more constrained approach where only dynamic and conditional/iterative tools are specified at runtime, and the static tools are expected to be memorized by the LLM. This distinction between "Allowed" and "Added" fundamentally impacts the model's performance, with the "Allowed" approach providing a broader, more versatile toolset that does not require the LLM to memorize tools, thereby reducing cognitive load and potentially increasing accuracy.

Table 6. Results - RTaC

Dataset	Model	JSON Similarity	Precision	Recall	F1-Score
Static	DeepSeek 1.3B	87.73	94.38	93.28	93.82
	CodeGen 2B	87.23	81.33	78.43	78.35
	DeepSeek 6.7B	87.79	93.01	95.05	94.01
	CodeLlama 7B	89.91	94.19	94.59	94.38
Dynamic	DeepSeek 1.3B	81.47	90.67	88.88	89.76
	CodeGen 2B	67.43	65.58	65.01	65.29
	DeepSeek 6.7B	82.17	92.03	92.16	92.09
	CodeLlama 7B	85.57	91.11	93.37	92.22
Modified	DeepSeek 1.3B	82.98	91.49	90.14	90.80
	CodeGen 2B	63.84	69.79	60.91	65.04
	DeepSeek 6.7B	87.61	91.18	91.13	91.15
	CodeLlama 7B	86.34	92.77	93.23	92.99
Bonus	DeepSeek 1.3B	83.96	91.47	92.01	91.73
	CodeGen 2B	55.37	69.79	60.91	65.04
	DeepSeek 6.7B	83.17	91.66	93.12	92.38
	CodeLlama 7B	86.92	92.22	94.91	93.54

Discussion: Table 6, shows commendable results on static and dynamic tooling scenarios for RTaC. Further, models under this pipeline gracefully handle the modification and deletion of static tools. This showcases the models' ability to comprehend the given docstrings. While the context length increases over the previous pipeline due to adding static tool docstring in each prompt, even small models such as DeepSeek 1.3B perform well under this pipeline, leading to minimal latencies. It must also be noted that CodeGen 2B is not a code instruct model, which explains its poor performance.

5 Results

Our experiments show that RTaC is comparable to GPT-4, not only in terms of accuracy but also in terms of cost/query and latency. This shows that tooling can be compared to function calls, which coding-based LLMs can efficiently handle (Table 7).

Table 7. Final Result and Comparison with GPT-4

Metric	GPT-4	RTaC		
		GPT-4	CodeLlama 7B	DeepSeek 1.3B
F-1 Score	86.82	**93.93**	93.22	93.28
JSON Similarity	74.88	**87.79**	87.42	85.73
Cost/Query ($)	0.0341	0.0312	0.0086	**0.0060**
Latency (s)	7.32	6.88	7.56	**5.25**
# of Parameters	170B	1760B	7B	**1.3B**

6 Conclusion

This paper introduces the concept of Reimagining Tooling as Coding (RTaC), a novel approach that reframes tool usage as a coding-related task. Our method involves fine-tuning LLMs on tool descriptions presented in docstring format. The desired output is formatted as variable assignments derived from API calls during training. To achieve this, we employ a unique dual-agent dataset generation method encompassing tool usage in various scenarios, including static, dynamic iterative, and conditional settings. By leveraging Coding-Base LLMs, which are inherently adept at comprehending coding elements, we perform instruction fine-tuning using this specially curated RTaC dataset. This innovative strategy empowers smaller, open-source Coding-Base LLMs, such as DeepSeek 1.3B and CodeLlama 7B, to achieve performance comparable to leading models like GPT-4. This is achieved with significantly reduced computational requirements and faster response times. RTaC presents a groundbreaking approach to tackling tool-usage challenges with LLMs, paving the way for significant advancements in LLM applications.

7 Future Scope

While RTaC demonstrates promising accuracy, there are two critical areas for improvement: mitigating hallucinations and achieving scalability. This section highlights potential avenues for future research that could significantly enhance the framework's performance.

Query Reformulation Modules: Prior research [8] has established a strong correlation between query quality and model accuracy. This finding aligns with our observations, where queries formulated to reflect the execution sequence achieve near-perfect accuracy precisely. This underscores the need for further exploration into query-optimizing modules. These modules would be designed to reformulate user queries into a format that the model can process efficiently and accurately.

Tool Retrievers: While docstring comprehension leads to high accuracy on a limited set of tools, scalability motivates the addition of tool retrievers to the pipeline. This will empower RTaC to be scaled to massive API sets and outperform current state-of-the-art methods like Gorilla and ToolLLaMA, which rely on tool memorization.

Improvements in Evaluation: The current evaluation metrics in this research domain, such as JSON Similarity and F1-score, fail to evaluate critical aspects such as correctness and optimality reliably. We find that string and AST-based evaluation is not fit for the task of tooling. The same query can often be answered via multiple sequences of tools, yet they are not scored as such during evaluation. To overcome this, we have been working on a bash-based toolset that can act as a deterministic evaluation benchmark for tooling LLMs. Our methodology involves creating an API set that can be mapped to bash operations on directories and files. Models' outputs can now be deterministically evaluated by state-matching after API call execution (Fig. 6).

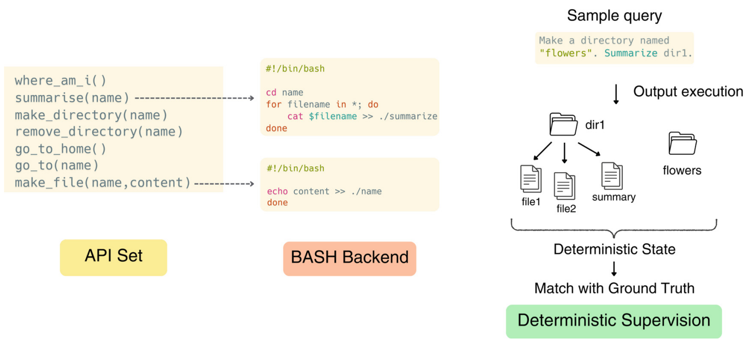

Fig. 6. Design for a bash-based deterministic evaluation benchmark

A Appendix

A.1 JSON Converter

This method is integral to the pipeline, functioning like a compiler. It is designed to transform model-generated code into a specific JSON format. The script categorizes the model's output into two primary types. The first is the General Case, which adheres to a standard variable assignment format using tool names and arguments. The second type is the conditional and iterative case, encompassing additional code structures like conditional statements and for-loops. These structures are used for temporary variable assignments, further expanding the script's capability to handle diverse output formats. In processing these outputs, the script employs several functions. The process_tool function is used for the general case, while iterative and conditional cases are managed by specialized bonus handlers that also leverage process_tool but with modified parameters. The make_tool function checks for the validity of tool and argument names, ignoring invalid entries. The update_arg_val function then processes valid arguments. This function is responsible for determining if argument values are lists, handling them recursively if so, and assessing the validity of each value, including scenarios where values are function calls or reference outputs from previous calls, ensuring comprehensive and accurate JSON conversion (Fig. 7).

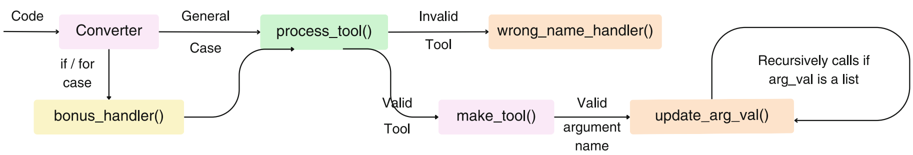

Fig. 7. JSON Conversion Pipeline

A.2 Prompt for Section 4.3

Here are the prompts for Sect. 4.3, testing closed and open-source LLMs for tool usage.

```
The only code you know to write is of type "var_i = function_call (
function_argument)", where i is the ith variable in use. You never output
anything else other than this format. You follow the sequence of completing
the query religiously. You have a given set of functions and you must use
them to answer the query. You are not allowed to use any other functions.

Here are the allowed functions-
{docstring of the functions}

Here are some sample queries and their respective responses:
{sample_python}

Answer very strictly in the same format shown above. Make sure to mention
type argument wherever relevant when calling works_list. Any missing type
arguments is not acceptable. Don't make unnecessary calls to any functions.
When given names, make sure to call search_object_by_name() to get work_ids.
Ensure logical continuity at each step. Ensure that the query is answered
fully. You are not allowed to nest function calls. You are not allowed to
output "python" or any other statement apart from the given format.
```

```
Do not use any other format for output than the one given above. Do not put
any comment in your answer. Anything else other than the format specified is
not acceptable.
Do not define any new helper functions or any other python functions apart
from the ones provided. Do not output any text apart from the final output
code. If you are unable to answer a query, you can output "
Unanswerable_query_error".

Answer the query: {user query}
```

Listing 1.1. Few Shot Prompting

```
Added Tools: {list of all the dynamic tools}

Query: {user query}
```

Listing 1.2. Tool Memorization Prompting

```
Allowed Tools: {list of all the tools}

Query: {user query}
```

Listing 1.3. RTaC Prompting

A.3 Default Tools Used to Generate New Tools

(See Table 8).

Table 8. Default Tools

Tool Description	Functionality
works list	Returns a list of work items matching the request
summarize objects	Summarizes a list of objects. The logic of how to summarize a particular object type is an internal implementation detail
prioritize objects	Returns a list of objects sorted by priority. The logic of what constitutes priority for a given object is an internal implementation detail
add work items to sprint	Adds the given work items to the sprint
get sprint id	Returns the ID of the current sprint
get similar work items	Returns a list of work items that are similar to the given work item
search object by name	Given a search string, returns the id of a matching object in the system of record. If multiple matches are found, it returns the one where the confidence is highest
create actionable tasks from text	Given a text, extracts actionable insights, and creates tasks for them, which are kind of a work item
who am i	Returns the string ID of the current user

References

1. Besta, M., et al.: Graph of thoughts: solving elaborate problems with large language models. arXiv preprint arXiv:2308.09687 (2023)
2. Cai, T., Wang, X., Ma, T., Chen, X., Zhou, D.: Large language models as tool makers. arXiv preprint arXiv:2305.17126 (2023)
3. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. (CSUR) **54**(4), 1–37 (2021)
4. Hu, E.J., et al.: Lora: low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
5. Huang, Y., et al.: MetaTool benchmark for large language models: deciding whether to use tools and which to use. arXiv preprint arXiv:2310.03128 (2023)
6. Inaba, T., Kiyomaru, H., Cheng, F., Kurohashi, S.: MultiTool-CoT: GPT-3 can use multiple external tools with chain of thought prompting. arXiv preprint arXiv:2305.16896 (2023)
7. Li, M., et al.: API-bank: a benchmark for tool-augmented LLMs. arXiv preprint arXiv:2304.08244 (2023)
8. Ma, X., Gong, Y., He, P., Zhao, H., Duan, N.: Query rewriting for retrieval-augmented large language models. arXiv preprint arXiv:2305.14283 (2023)
9. Ning, X., Lin, Z., Zhou, Z., Yang, H., Wang, Y.: Skeleton-of-thought: large language models can do parallel decoding. arXiv preprint arXiv:2307.15337 (2023)
10. Parisi, A., Zhao, Y., Fiedel, N.: TALM: tool augmented language models. arXiv preprint arXiv:2205.12255 (2022)
11. Patil, S.G., Zhang, T., Wang, X., Gonzalez, J.E.: Gorilla: large language model connected with massive APIs. arXiv preprint arXiv:2305.15334 (2023)
12. Qin, Y., et al.: ToolLLM: facilitating large language models to master 16000+ real-world APIs. arXiv preprint arXiv:2307.16789 (2023)
13. Schick, T., et al.: ToolFormer: language models can teach themselves to use tools. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
14. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: HuggingGPT: solving AI tasks with ChatGPT and its friends in hugging face. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
15. Tang, Q., Deng, Z., Lin, H., Han, X., Liang, Q., Sun, L.: ToolAlpaca: generalized tool learning for language models with 3000 simulated cases. arXiv preprint arXiv:2306.05301 (2023)
16. Wang, Y., et al.: Self-instruct: aligning language model with self generated instructions. arXiv preprint arXiv:2212.10560 (2022)
17. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: Advances in Neural Information Processing Systems, vol. 35, pp. 24824–24837 (2022)
18. Yang, K., et al.: If LLM is the wizard, then code is the wand: a survey on how code empowers large language models to serve as intelligent agents. arXiv preprint arXiv:2401.00812 (2024)
19. Yao, S., et al.: Tree of thoughts: deliberate problem solving with large language models. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
20. Ye, J., et al.: ToolEyes: fine-grained evaluation for tool learning capabilities of large language models in real-world scenarios. arXiv preprint arXiv:2401.00741 (2024)
21. Ye, J., et al.: RoTBench: a multi-level benchmark for evaluating the robustness of large language models in tool learning. arXiv preprint arXiv:2401.08326 (2024)

22. Yuan, S., et al.: EASYTOOL: enhancing LLM-based agents with concise tool instruction. arXiv preprint arXiv:2401.06201 (2024)
23. Zhang, K., Chen, H., Li, L., Wang, W.: Syntax error-free and generalizable tool use for LLMs via finite-state decoding. arXiv preprint arXiv:2310.07075 (2023)

Scientific Software Citation Intent Classification Using Large Language Models

Ana-Maria Istrate[1(✉)], Joshua Fisher[2], Xinyu Yang[3], Kara Moraw[4,5], Kai Li[6], Donghui Li[1], and Martin Klein[7]

[1] Chan Zuckerberg Initiative, Redwood City, CA 94063, USA
aistrate@chanzuckerberg.com
[2] Elsevier, Research Collaborations Unit, Amsterdam, The Netherlands
[3] Department of Information Science, Cornell University, Ithaca, USA
[4] Software Sustainability Institute, Edinburgh, UK
[5] EPCC, University of Edinburgh, Edinburgh, UK
[6] School of Information Sciences, University of Tennessee, Knoxville, USA
[7] Los Alamos National Laboratory, Los Alamos, NM, USA

Abstract. Software has emerged as a crucial tool in the current research ecosystem, frequently referenced in academic papers for its application in studies or the introduction of new software systems. Despite its prevalence, there remains a significant gap in understanding how software is cited within the scientific literature. In this study, we offer a conceptual framework for studying software citation intent and explore the use of large language models, such as BERT-based models, GPT-3.5, and GPT-4 for this task. We compile a representative software-mention dataset by merging two existing gold standard software mentions datasets and annotating them to a common citation intent scheme. This new dataset makes it possible to analyze software citation intent at the sentence level. We observe that in a fine-tuning setting, large language models can generally achieve an accuracy of over 80% on software citation intent classification on unseen, challenging data. Our research paves the way for future empirical investigations into the realm of research software, establishing a foundational framework for exploring this under-examined area.

Keywords: Research software · Citation intent · Large Language Models

1 Introduction

Research software is a critical instrument in contemporary scientific environments, as it offers the computational capacity to expand human capacities to observe and investigate phenomena and acquire new knowledge from the increasing amount of data [19]. As a result, software gradually takes more important roles in data-driven science [40] and is regarded as a "first-class research object"

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 80–99, 2024.
https://doi.org/10.1007/978-3-031-65794-8_6

by scientists in a growing list of research domains [6]. During the past decade, there has been significant progress in the development of research infrastructure to support the publishing, using, and crediting research software [2,39], which in turn, supports empirical investigations into the impacts of research software and their roles in scientific research [27,34]. All these efforts are believed to contribute to the construction of a more fair and transparent scientific system [14].

Despite this progress, one major gap in existing research is that there is a lack of a more granular understanding of the links between scientific publications and research software, i.e., how software is cited in scientific publications. This question is central to citation context analysis developed from the field of scientometrics, which examines the different types of contexts in citation sentences, such as sentiment, function, or level of importance of individual citations [43]. This approach is able to reveal more granular reasons behind citations and impact and hence contributes to a deeper understanding of how credit and knowledge flow between publications [9]. On the same page, when this method is applied to research software, we can also understand not only how many times a software object is cited in publications, but the reasons why it is cited. This knowledge is critical for the construction of a new research infrastructure for research software and evaluation of research software and its developers.

A large number of citation context classification schemes have previously been proposed, focusing specifically on citations between scientific publications (Scite [31]). However, we are arguing that these schemes cannot be used to sufficiently understand why software is cited in scientific publications, since there may be distinct reasons why a software package is cited in a paper. In light of this topic, very few schemes have been proposed for citations of research software in scientific publications, with the exceptions of SoftCite [12] and SoMeSci [37]. We believe it is vital to revitalize existing efforts by (1) developing a new classification system for citation intents of research software that builds on existing efforts, and (2) applying and testing the system on new software mentions datasets.

In this paper, we are presenting our preliminary results, including (1) a new classification system for software citation intents in full-text scientific publications and (2) performance assessment of machine learning algorithms, in particular large language models, for classifying the citation intent of software-mention sentences. We compile a new dataset that can be used for software citation intent analysis by aggregating and annotating the SoftCite [12] and SoMeSci [37] datasets. We evaluate the performance of the machine learning models on a subset from a recent large-scale software name mention dataset published by the Chan Zuckerberg Initiative (CZI) [22]. More specifically, we are examining the intent of informal citations to the software. Borrowing previous research [35], formal and informal citation approaches to mentioning software in scientific publications are defined as those mentions with and without an official citation to the software respectively. This project, to our knowledge, is the first effort to identify and classify contexts of software mentions (or informal software citations) from full-text publications. It is our hope that our results will improve existing research infrastructure for empirical studies on research software.

2 Related Work

2.1 Research Software Studies

Software has become a cornerstone of contemporary scientific systems, due to the large quantity of data available to researchers and the computational resources required to analyze such data [10]. Given its elevated importance, it is important to treat research software as a "first-class research object" just like research articles, which requires the support from infrastructure to publish, peer-review, reuse, and cite software entities [6]. Among these requirements, giving and tracing citations to software is central to the assignment of reward to software development activities and promoting researchers' motivation to develop and publish software [13].

Software citation is a highly challenging issue in the scholarly communication system because various empirical studies have found that software is inconsistently cited, if cited at all, in scientific publications [20,28]. In addition, when a software is cited, there is often a complicated relationship between citations and software, which makes it very hard to trace how specific software is cited [26]. These findings have inspired recent efforts to develop software citation principles, especially aligned with the FAIR Principles [2,39].

Despite these progresses to develop a more robust software citation infrastructure, it is commonly accepted that information citations to software, or software mentions, are critical for investigating the links between scientific publications and software [38]. This approach is reflected in recent efforts to publish large-scale software name datasets extracted from full-text publications, especially the CZI dataset that covers close to 4 million open-source scientific publications [22], as well as some similar datasets [12]. These fresh open datasets will undoubtedly promote new empirical research on this critical topic to promote the openness of science, including the present research.

2.2 Citation Intent Classification

Citations have long been regarded as a gold standard to measure one's impacts within the scientific system, as they represent one's intellectual debts to other authors [24,29]. Based on this normative theory, it is possible to construct a systematic evaluation system by collecting the citations between all documents, which is the idea behind the Scientific Citation Index (SCI) as well as many other scientific evaluation systems [16]. Despite the proven effectiveness (at least to some text) of using citations to evaluate one's scientific impacts [4,17], however, one concern with just focusing on the citation count is that the citation itself can bear multiple semantic meanings. For example, Bruno Latour, the famous sociologist of science, made the following argument:

> "[Sources] may be cited without being read, that is perfunctorily; or to support a claim which is exactly the opposite of what its author intended;

> or for technical details so minute that they escaped their author's attention; or because of intentions attributed to the authors but not explicitly stated in the text." [25]

Such challenges to classic citation analysis method gave rise to a new line of research that focuses on the symbolic meanings of citations in the full-text publications, often called *citation context and content analysis* [8]. In their review of this topic, Zhang et al. identified a few important aspects of the symbol that have been analyzed, such as sentiment, function, or level of importance of individual citations [43].

In addition, a few important classification systems have been proposed to classify regular citations (especially those citing research articles) during the past few decades, each with their own categories and considerations. [23,30,31,41]. However, Cohan et al. [7] argue that these classification systems are usually too fine-grained to allow a meaningful application to software citations. Having many fine-grained categories successfully captures rare contexts but hinders a meaningful analysis of the citation impact. More recent efforts, such as SoftCite [12] and SoMeSci [37], are trying to directly address this research problem, by developing citation context categories dedicated to software entities. However, both of these efforts are based on and only tested using limited publication samples. Moreover, the citation context categories are not the same between the two datasets. As a result, we believe there is still a large gap in this field that can be addressed by our efforts to apply advanced data science methods to classify software citation sentences to a common scheme.

3 Methods

We first defined a set of citation intent classes and created mappings for any existing informal software citation datasets with intention annotations. We then used the combined dataset to fine-tune multiple language models.

3.1 Citation Intent Classes

We reviewed multiple schemes for citation context and intention that have been proposed for both regular research publications and software. The respective schemes are listed in Table 1. Both the ACL-ARC and SciCite were proposed for the intent classification of research article citations. For their work on the ACL Anthology Reference Corpus (ARC) [5], Jurgens et al. [23] propose six categories for citation function, unifying several previously proposed schemes. Their scheme focuses on how authors align their work to cited publications and maintains a higher granularity for classifying indirect mentions like background information or contrasting related works than our scheme does. We found that these kinds of mentions currently don't occur with a high enough frequency to warrant further splitting these classes. For SciCite, Cohan et al. [7] similarly argue that previous datasets for citation intent often use too fine-grained schemes. They propose instead to use only three categories, focusing on direct use and comparison of results, and regarding everything else as background information providing more context.

However, these schemes cannot be directly transferred to software citations. Unlike research article citations, software can be cited as research software that has been created as part of the publication. Both the SoftCite [12] and SoMeSci [37] provide annotations for the intent of software citations and propose software-specific schemes. Apart from software creation and usage, they both consider categories related to software publication, namely *sharing* and *deposition*.

Table 1. Citation Intent Schemes.

Source	Target	Categories
ACL-ARC [23]	publications	Background, Motivation, Uses, Extension, Comparison or Contrast, Future
SciCite [7]	publications	Background information, Method, Result comparison
SoftCite [12]	software	Created, Used, Shared
SoMeSci [37]	software	Usage, Mention, Creation, Deposition

For the development of our citation intent scheme, we established the following two guiding principles:

1. The scheme should be able to distinguish the most common and relevant types of citation intent.
2. The scheme should exhibit high inter-annotator agreement so that it can be consistently applied by a human.

The first principle ensures that the resulting scheme is not too fine-grained, as this would hinder a meaningful analysis in future empirical research. The second principle allows multiple annotators to label a potentially large corpus consistently. Based on the principles and the previously described schemes, we propose the following three categories in our scheme for informal research software citation intent:

– **Paper <describes_creation_of> Software:** the paper describes or acknowledges the creation of a research software entity.
– **Paper <describes_usage_of> Software:** the paper describes the use of research software in any part of the research procedure, for any purpose.
– **Paper <describes_related_software> Software:** the paper describes the research software for any other reasons beyond the first two categories. Note that throughout the paper, we refer to this category by using "related" and "mention" interchangeably.

The most relevant distinction in our scheme is that between a sentence in a publication describing the creation of a piece of software as opposed to one citing the usage of software. We make sure not to confound the annotation process or analysis with any rare labels by instead encapsulating any other mention in the third category.

Similar to existing efforts, this scheme only considers *functional* intents, i.e., functional reasons for mentioning the software in publications, instead of other aspects of the intent, such as sentiment and importance [43]. In contrast to the schemes proposed for SoMeSci and SoftCite, we specifically did not include a category for *sharing* or *deposition*. We believe that distinguishing this citation intent from creation and usage is not strongly relevant to the evaluation of impact of software being mentioned in publications, especially considering the case where sharing software is often strongly related to creating the software in the first place.

An important attribute of our scheme is that it is designed to be applied on the sentence level: the evaluation is made based on each sentence where a software entity is mentioned. Hence, a paper-software pair can have multiple citation intents if a software entity is mentioned multiple times in the paper. For creating our datasets, we decided that each sentence could only be classified into one category. In the case where multiple categories were applicable, we chose the category carrying more weight in evaluating impact, where *Paper* $<$ *describes_creation_of* $>$ *Software* has a larger weight than *Paper* $<$ *describes_usage_of* $>$ *Software*, which in turn has a larger weight than *Paper* $<$ *describes_related* $>$ *Software*. Note that since we are considering only one citation intent per sentence, we are making the assumption that even if multiple software are mentioned in the same sentence, they are mentioned with the same intent. In some rare corner cases, such as multiple software being mentioned in the same sentence with multiple intents, this assumption will not hold true. Hence, our findings are not applicable in these cases.

3.2 Data

We wanted to re-use existing datasets as much as possible and build on top of previous work, rather than create new datasets and define new gold standards. This is why we chose to build on the SoftCite [12] and SoMeSci [37] datasets by merging them into one representative dataset that can be used for analyzing software citation intent. The datasets, for the most part, consist of single sentences that contain a software mention (*informal* citation, by means of verbal reference to software, whereas a *formal* citation would be by verbal means in conjunction with an included URI, or an official citation, such as a literature reference) and their corresponding labels, which vary between the datasets. Consolidating these similar yet still slightly different datasets was outside the scope of this work. However, given our decision regarding citation intent classes outlined above, we had to make a few adjustments to the existing labels in the provided datasets. We did this through manual curation. Table 2 shows the mappings between our proposed scheme and software citation intent schemes used in the SoMeSci and SoftCite datasets. From the SoftCite dataset, we were able to transfer labels **Used** and **Created** directly to our **Usage** and **Creation** classes and mapped most of the **Shared** labeled data into **Creation**. After careful consideration and debate between multiple annotators, we moved some records that had multiple labels or no labels at all into our **Mention** category. For the SoMeSci dataset, we transferred the **Usage**, **Mention** and **Creation** labels straight to our own labels. We disregarded entries with a label of **Deposition**.

Table 2. Mappings to other software citation intent schemes.

Ours	SoftCite	SoMeSci
Paper <describes_creation_of> Software	Created	Creation
Paper <describes_usage_of> Software	Used	Usage
Paper <describes_related> Software	n/a	Mention

As part of data curation, we created a pipeline that downloaded all available full text of papers in the two datasets (SoftCite [12] and SoMeSci [37]) via the PMC API [42] in order to augment the existing data at the sentence-level with an expanded citation context of three sentences surrounding the citation: *leading*, *citing* and *trailing sentence*. After all pre-processing, we ended up with a single dataset. The final dataset consists of 3188 software citations, each labeled as **Creation**, **Usage**, or **Mention**, along with the sentence in which the software mention occurs (<*citing sentence*>) and the citation context. The citation context consists of:

- <*leading sentence*><*citing sentence*><*trailing sentence*>

Some examples from the combined training dataset can be found in Table 3. In addition, we augmented the dataset with 1,000 sentences that contained no citation contexts by sampling randomly from the set of sentences that were not tagged with a software mention. This data can be used as negative training examples. The distribution of labels and the number of words in the contexts in the training dataset used can be seen in Fig. 1.

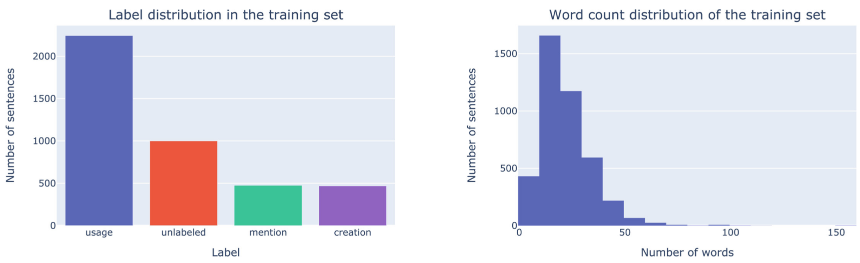

Fig. 1. Training Dataset Data Distribution

Table 3. Examples from the training data

Intent type	Sentence	Full Context
creation	We have developed coXpress as a means of identifying groups of genes that are differentially co-expressed	**We have developed coXpress as a means of identifying groups of genes that are differentially co-expressed.** The utility of coXpress is demonstrated using two publicly available microarray datasets
usage	The resulting trajectories were analyzed using the CPPTRAJ module in AMBERTOOLS 16 [55,56]	To study the dynamic behavior of the proposed ligand-protein complex, we have performed in a short production run of 10 ns using AMBER 14. **The resulting trajectories were analyzed using the CPPTRAJ module in AMBERTOOLS 16 [55,56]**
mention	M-Track provides a valuable and user-friendly interface to streamline the analysis of spontaneous grooming in biomedical research studies	**M-Track provides a valuable and user-friendly interface to streamline the analysis of spontaneous grooming in biomedical research studies**
unlabeled	Developing a new drug is an evidence-based exercise	**Developing a new drug is an evidence-based exercise**

We used the combined dataset of 4,188 sentences to train the language models. We split the dataset 80/20 for training and testing in order to facilitate a reasonable comparison between models. We evaluated the models on the test held-out portion of the data that the model has not seen during training. Moreover, we had an additional dataset of 210 samples curated by Chan Zuckerberg Initiative. This dataset is a subset of the CZI Software Mentions Dataset [21] and was manually curated by CZI annotators by reviewing sentences that contain mentions of software names; the dataset was initially curated using a more granular intent classification which was subsequently mapped to the intent classification described above (**creation**, **used**, **mention**). Note that since the original CZI Software Mentions Dataset was not annotated with intent classification classes, this has been done manually by CZI bio-curators after the initial dataset has been published. Because of the effort required and the size of the initial dataset, we were only able to use a subset for evaluation. One annotator initially classified the sentences, and an additional annotator resolved any conflicts at an ulterior time.

3.3 Training Models

We explored finetuning several BERT [11] models, as well as GPT-3.5 [32] and GPT-4 [33] in various training settings.

BERT Models. We studied four different BERT models, namely BERT [11], DistilBERT [36], SciBERT [3] and PubMedBERT [18]. BERT [11] was pre-trained on BookCorpus [44] and English Wikipedia [15]. It is well suited for fine-tuning downstream tasks that use a full sentence, e.g. for text classification. DistilBERT [36] is a smaller and thus a faster BERT model. It was pre-trained on the same corpus using knowledge distillation with BERT as its teacher. SciBERT was pre-trained on a corpus of scientific texts from Semantic Scholar [1]. It was found to outperform BERT on tasks and datasets in the scientific domain. PubMedBERT [18] was in turn trained on biomedical papers, specifically abstracts from PubMed and full-text articles from PubMedCentral [42]. Hence, it is tailored to tasks in the biomedical domain.

We used the model architecture provided by Hugging Face's library, applying fine-tuning to all four models for text classification tasks. This fine-tuning was consistent across models, employing identical parameters: (epochs=10, learning_rate=2e-5, weight_decay=0.01).

GPT-3.5/4. We also investigated GPT-3.5 and GPT-4 in three different learning settings: *zero-shot learning*, *few-shot learning* and *fine-tuning*. In *zero-shot learning*, the model is only given a description of the task before solving the task. Specifically, we used the system message described in Listing 1.1 to instruct the model. In *few-shot learning*, the model receives a similar instruction followed by a handful of examples for expected interactions between the user and the assistant (i.e. the model). The model is supposed to learn from these few examples how to generalize on new data. For this, we sampled five examples from each class (creation, usage, mention, and none) and provided these to the model. The corresponding prompt is shown in Listing 1.2.

Listing 1.1. Zero-shot prompt. The 'instruction_message' sets the assistant's tone, behavior or persona.

```
instruction_message = (
        "You␣are␣a␣scientist␣trying␣to␣figure␣out␣the␣citation␣"
        "intent␣behind␣software␣mentioned␣in␣sentences␣coming␣"
        "from␣research␣articles.␣"
        "Your␣four␣categories␣are:␣creation,␣usage,␣mention,␣or␣none.␣"
        "The␣definitions␣of␣the␣classes␣are:␣"
        "-␣creation:␣software␣was␣created␣by␣the␣authors␣of␣the␣paper"
        "-␣usage:␣software␣was␣used␣in␣the␣paper␣"
        "-␣mention:␣software␣was␣mentioned␣in␣the␣paper,␣but␣not␣used,␣"
        "nor␣created␣"
        "-␣none:␣none␣of␣the␣previous␣3␣categories␣apply"
        "You␣need␣to␣output␣one␣category␣only."
)
# send instruction as system message to shape the assistant's behaviour
zero_shot_messages = [
    {
        "role": "system",
        "content": instruction_message
    }
]
```

Listing 1.2. Few-shot prompts. When interacting with the OpenAI API, a query can have one of three different roles: (1) **system** – where the behavior, tone, or persona, of the GPT assistant is being defined. We set the behavior of the GPT assistant as in the 'instruction_message' presented in Listing 1.1. The other roles assumed can be (2) **user** – which contains the queries presented by the user to the assistant, and (3) **assistant** – which gives back the GPT assistant's response.

```
# same instruction message as in zero-shot learning
few_shot_messages = [
    {
        "role": "system",
        "content": instruction_message
    }
]

# append examples as messages
for example in few_shot_examples["usage"]:
    # example for user prompt (sentence or context)
    few_shot_messages += [{"role": "user", "content" : example}]
    # example for assistant output (category)
    few_shot_messages += [{"role": "assistant", "content" : "usage"}]
# same for "creation", "mention" and "none"
...
```

Fine-tuning can further improve a model's performance. Instead of a handful of examples, a larger training set is provided. We fine-tuned GPT3.5 on the sentence and full context, using the same dataset as for the BERT models. Both models were fine-tuned for a total of 5 epochs using the OpenAI API and the 'gpt-3.5-turbo' model. The OpenAI API does not allow much hyper-parameter tuning besides the number of epochs, so we used the API's default settings. Since the model can give back answers that do not fit into one of the provided classes, we post-process the answers by lowercasing and stripping punctuation marks.

4 Results

We evaluated the models on a 20% test split of the training data (Table 4), as well as on the additional CZI Validation Dataset (Table 5). The evaluation metrics used to assess model performance were precision, recall, F1-score, and overall accuracy, both at the individual label level and for the aggregate performance. In Table 4 we also attach the metrics reported for classification of intent classes by SoftCite [12], as well as SoMeSci [37]. These metrics are extracted from the corresponding papers and are evaluated on different datasets then the ones in this paper. Note that since the classifiers are not evaluated on the same data or even trained to predict the same citation intent classes, the results are not necessarily comparable. Nonetheless, we report the metrics where applicable. For example, the SoftCite paper [12] only reports the performance of a trained citation intent classifier for the **used** and **not used** categories. Hence, we show the metrics for the **used** category, mapping it to our **Usage** category. For the SoMeSci [37] dataset, the paper reports metrics for the following classes: **Allusion**, **Usage**,

Creation and **Deposition**. We map the **Allusion** to our **Mention** category, and the **Usage** and **Creation** classes directly. We don't report the **Deposition** metrics, since we have discarded this category in our own scheme.

4.1 Results of BERT Models

As shown in Table 4, all models achieve high scores across the metrics and categories on the test split. PubMedBERT outperforms the others by a small degree. As seen in Table 5, on the CZI Validation Dataset, model performance drops across the board, DistilBERT, however, outperforming the other BERT models. Considering the moderate sample size of the training dataset, it may imply that a light architecture such as DistilBERT performs better in this dataset, achieving a more balanced result, especially when compared with the original BERT.

For the category-specific results, the classification is related to the availability of the labels in the training and validation dataset - all the models perform best in the **Usage** label, which is the most frequent in both the training and validation set, and perform the worst on the **Creation** label. Notably though, DistilBERT is the only model to achieve non-zero scores in the **Creation** category across all three metrics, highlighting its unique capability to identify and classify this particularly challenging category. For the **Mention** category, the performance drops for all BERT models, with PubMedBERT outperforming the other BERT models. The same trend can be observed for the test split in Table 4. By definition, this category encompasses more varied instances than the other two, which might be why the models struggle to consistently recognise it. In the **Usage** category, SciBERT and PubMedBERT perform the highest.

4.2 Results of GPT-3.5/GPT-4

In general, **fine-tuning** the model at the sentence level seems to achieve the best results for the GPT-3.5 model on both the test split and the CZI validation dataset. The GPT3.5 model fine-tuned on the sentence level achieves the highest performance on the challenging CZI validation set (P = 0.571, R = 0.531, F1 = 0.545, Accuracy = 0.881) surpassing all BERT models as well. Fine-tuning GPT-3.5 on the entire context (containing the leading, citing, and trailing sentence) leads to a decrease in performance. This tells us that feeding the entire context around the sentence is actually not helping the model learn more information about the intent on citing the software in the sentence itself. The same observation holds for both GPT3.5/4 few-shot models. None of the **few-shot** and **zero-shot** approaches for both GPT-3.5 and the GPT-4 models achieved close to the performance of the fine-tuned models, which means that despite these models' generalizable power, they still don't hold enough information to be able to classify software citation intent without additional training data. We observe in general that GPT-4 models tend to outperform GPT-3.5 models both in zero and few-shot contexts. Notably, however, both GPT-3.5 and GPT-4 few-shot models generally tend to do worse than the zero-shot models counterparts.

Table 4. Evaluation of Different Models on the Test Split. We assessed the overall Precision (P), Recall (R), F1-score (F1), and Accuracy (Acc) across the entire validation dataset. Additionally, we analyzed the Precision, Recall, and F1-score for each label within every model. For comparison, we attach the metrics reported for classification of intent classes by SoftCite [12], as well as SoMeSci [37]. These metrics are extracted from the corresponding papers and are evaluated on different datasets than the ones in this paper.

Model	Training Methodology	Overall				Creation			Mention			Usage			Unlabeled		
		P	R	F1	Acc	P	R	F1	P	R	F1	P	R	F1	P	R	F1
BERT	Fine-tuning	0.866	0.859	0.862	0.903	**0.90**	**0.87**	**0.88**	0.71	0.69	0.70	0.93	0.94	0.94	0.92	0.94	0.93
DistilBERT	Fine-tuning	0.823	0.826	0.824	0.884	0.86	0.81	0.84	0.59	0.62	0.61	0.95	0.93	0.94	0.90	0.95	0.92
SciBERT	Fine-tuning	0.85	0.846	0.846	0.906	0.89	0.77	0.82	0.63	0.68	0.65	0.96	0.95	0.95	0.93	0.99	0.96
PubMedBERT	Fine-tuning	**0.867**	**0.891**	**0.88**	**0.919**	0.88	0.88	0.88	**0.69**	**0.78**	**0.73**	**0.97**	**0.93**	**0.95**	0.94	0.97	0.96
GPT-3.5	Zero-Shot	0.7	0.608	0.627	0.717	0.77	0.46	0.57	0.24	0.47	0.32	0.84	0.86	0.85	0.96	0.65	0.77
	Few-Shot (sentence)	0.617	0.556	0.54	0.612	0.81	0.46	0.59	0.28	0.24	0.26	0.93	0.57	0.71	0.45	0.95	0.61
	Few-Shot (context)	0.546	0.512	0.463	0.5	0.72	0.57	0.64	0.21	0.15	0.17	0.88	0.35	0.5	0.38	0.98	0.55
	Fine-tuning (sentence)	0.839	0.867	0.851	0.9	0.8	0.91	0.85	0.65	0.69	0.67	**0.97**	**0.93**	**0.95**	0.94	0.93	0.94
	Fine-tuning (context)	0.766	0.808	0.783	0.857	0.66	0.88	0.76	0.49	0.52	0.51	0.96	0.88	0.92	**0.95**	**0.95**	**0.95**
GPT-4	Zero-Shot	0.684	0.662	0.664	0.815	0.73	0.70	0.71	0.26	0.11	0.15	0.83	0.96	0.89	0.92	0.88	0.90
	Few-Shot (sentence)	0.746	0.736	0.738	0.839	0.74	0.62	0.67	0.46	0.44	0.45	0.93	0.91	0.92	0.86	0.97	0.91
	Few-Shot (context)	0.716	0.73	0.716	0.832	0.68	0.82	0.74	0.43	0.26	0.33	0.92	0.91	0.92	0.83	0.93	0.87
SoftCite	BidGRU x 10	–	–	–	–	–	–	–	–	–	–	0.965	0.992	0.979	–	–	–
	SciBERT	–	–	–	–	–	–	–	–	–	–	0.956	0.995	0.975	–	–	–
SoMeSci	Bi-LSTM-CRF (Test)	–	–	–	–	0.87	0.51	0.64	0.68	0.18	0.29	0.77	0.84	0.8	–	–	–
	Bi-LSTM-CRF (Devel)	–	–	–	–	0.97	0.5	0.66	0.68	0.28	0.4	0.78	0.8	0.79	–	–	–

We haven't investigated in detail why that might happen, but it is an interesting observation to note that for this task, learning from a few examples is detrimental, whereas learning from a lot (i.e. finetuning) is helpful.

Inspecting per-class performance, we observe that, similarly to BERT models, GPT models tend to do very well on predicting **Usage** and **Unlabeled** labels, achieving precision, recall and F1 score >0.9 for both test splits and CZI Validation Dataset and struggle the most with predicting the **Mention** class. This makes sense given that any software will be, after all, mentioned in the paper. While we instruct the model to predict this label if none of the **Usage** or **Creation** might apply, it is still ambiguous. Some examples of model mistakes on the CZI Validation dataset can be seen in Table 6.

5 Data and Code Availability Statement

Training scripts are available on our GitHub repository: https://github.com/karacolada/SoftwareImpactHackathon2023_SoftwareCitationIntent. BERT fine-tuning scripts can be found under **BERT_finetuning** and all the GPT scripys, including zero-shot, few-shot and fine-tuning under **GPT_models**. The merged dataset, together with training, validation and test splits, as well as GPT-3.5 formatted data and the CZI Validation Dataset can also be found under the **data** folder, together with extra documentation and a README file.

Table 5. Evaluation of Different Models on the CZI Validation Dataset. We assessed the overall Precision (P), Recall (R), F1-score (F1), and Accuracy (Acc) across the entire validation dataset. Additionally, we analyzed the Precision, Recall, and F1-score for each label within every model.

Model	Training Methodology	Overall				Creation			Mention			Usage		
		P	R	F1	Acc	P	R	F1	P	R	F1	P	R	F1
BERT	Fine-tuning	0.323	0.368	0.335	0.771	0.00	0.00	0.00	0.15	0.29	0.20	0.94	0.85	0.90
DistilBERT	Fine-tuning	0.481	0.412	0.443	0.801	0.71	0.50	0.59	0.29	0.26	0.27	0.94	0.88	0.91
SciBERT	Fine-tuning	0.302	0.308	0.306	0.80	0.00	0.00	0.00	0.27	0.35	0.30	0.95	0.90	0.92
PubMedBERT	Fine-tuning	0.319	0.392	0.342	0.81	0.00	0.00	0.00	0.28	0.39	0.33	0.94	0.91	0.93
GPT-3.5	Zero-Shot	0.464	0.511	0.478	0.8	0.55	0.6	0.57	0.35	0.61	0.44	0.96	0.84	0.89
	Few-Shot (sentence)	0.525	0.291	0.373	0.457	0.5	0.3	0.37	0.6	0.39	0.47	1.00	0.47	0.64
	Few-Shot (context)	0.454	0.195	0.269	0.338	0.5	0.2	0.29	0.33	0.22	0.26	0.98	0.36	0.53
	Fine-tuning (sentence)	**0.571**	**0.531**	**0.545**	**0.881**	0.71	0.5	0.59	**0.59**	**0.7**	**0.64**	**0.98**	**0.93**	**0.95**
	Fine-tuning (context)	0.553	0.503	0.509	0.819	0.83	0.5	0.62	0.41	0.65	0.5	0.97	0.86	0.91
GPT-4	Zero-Shot	0.473	0.544	0.495	0.8	0.58	0.70	0.64	0.35	0.65	0.45	0.96	0.82	0.89
	Few-Shot (sentence)	0.473	0.385	0.421	0.614	**0.7**	**0.7**	**0.7**	0.21	0.17	0.19	0.98	0.67	0.79
	Few-Shot (context)	0.399	0.206	0.269	0.471	0.5	0.2	0.29	0.11	0.09	0.1	0.99	0.54	0.7

Table 6. Examples of mistakes the GPT3.5 model fine-tuned at the sentence level makes on the CZI Validation dataset

Sentence	True Label	Predicted Label
Very recently, one research group applied Mask R-CNN on cervix segmentation tasks, the obtained (Dice, IoU) score is (0.8711, 0.765) on "Kaggle Dataset" as reported in [26]	mention	usage
Identifying major effect QTL underlying single or multiple traits in various populations determines the successful use of QTL through MAS [8]	mention	none
In Fig. 11, p miss of SVM-SMP is nearly equal to 0, which is much better than SVM-LA	usage	none
FVA can be set up in COBRA toolbox using the function fluxVariability()	usage	mention

6 Discussion

The determination of software citation intent requires a system to not only identify the entity but understand the semantic relationships provided by the context around the entity. In this study, we focused on the latter part of said system, by investigating which model types can effectively learn the intent of authors by their reporting of software. Prior work done by the SoftCite [12] and SoMeSci [37] groups offered valuable datasets that were combined and normalized to a simple scheme. This corpus allowed for the fine-tuning and experimentation of various flavors of BERT, GPT-3.5 and GPT-4 models. Our intuition was that these models would be able to accurately characterize these intents. One interesting finding

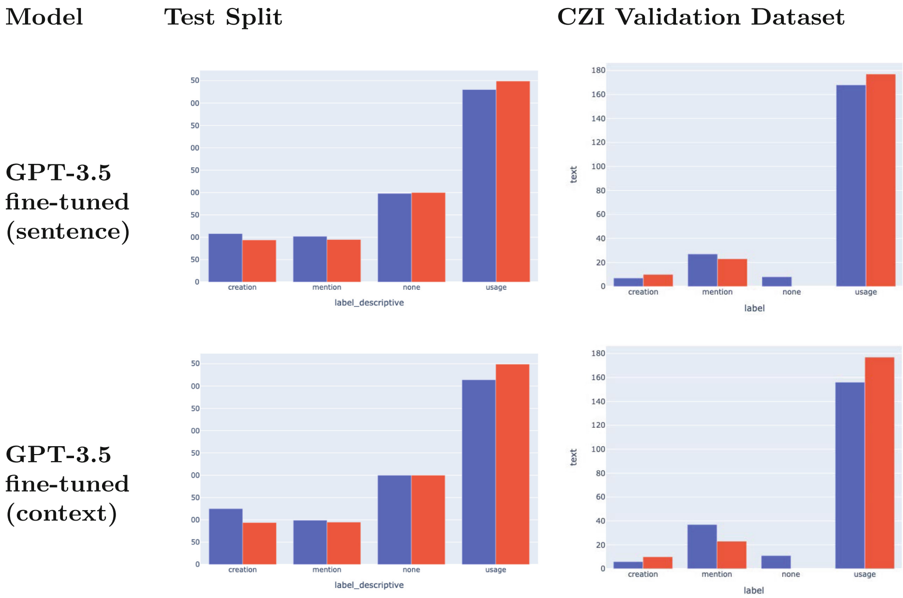

Fig. 2. GPT-3.5 fine-tuned Comparison of True and Predicted Labels Distributions. Y axis represents the counts, and X axis the label categories. pred_gpt3.5 true

was that including the full context in classification seemed to hurt model performance. This insensitivity towards extra contextual clues indicates that intent can typically be determined in close proximity to the mention of the software entity. Further text analysis may elucidate exactly what type of language is characteristic of each intent class. A quick word frequency count in sentences that are of the "creation" class identify "software", "available", "http", "developed", and "source" as the most common. A more proper analysis for all intent types to identify word patterns could improve the classification system (Fig. 2).

To test the full breadth of capabilities of the GPT models, we employed various experiments including zero-shot, few-shot and fine-tuned approaches. None of the **few-shot** and **zero-shot** approaches for both GPT-3.5 and the GPT-4 models achieved performance comparable to the fine-tuned models, which means that software citation intent classification is not a task that these models can do out of the box, without additional training. Adding example cases to the prompts in a few-shot setting yielded a decrease in both precision and recall over the evaluation sets compared to the zero-shot setting. Beyond this preliminary work, future experiments will have to test different versions of prompts to further probe this unexpected behavior, as prompt engineering was not an extensive part of this work. Fine-tuning the GPT-3.5 model generated results comparable to the BERT models. We did not experiment with fine-tuning a GPT-4 model because this process is closed to the general public at the moment of paper publication, but we would expect that a GPT-4 fine-tuned model would achieve even higher performance. The easiest category to predict was **Usage**, followed by **Creation**. The **Mention** category was the hardest for models to learn to predict well, which makes sense given that software falling in the other classes can automatically fall under this category as well. While trained on different intent categories and data, our best models surpass the metrics reported by SoftCite and SoMeSci on the **Creation** and **Mention** categories and are comparable for the **Usage** category.

Fine-tuned models generally exhibited high performance on the test set. However, despite our best efforts to identify and resolve systematic differences in the CZI validation set from our test set, models were not able to achieve similar performance. The only competitive performance can be observed using the fine-tuned GPT-3.5 model. Given this is a challenging dataset coming from a different distribution than the training data, this speaks to the power of the GPT family of models to generalize and find nuance in ambiguous text, compared to BERT models. Further experiments necessitate a proper quality and error analysis of the validation set.

7 Conclusion

In conclusion, this preliminary work presents a new system for the classification of software citation intent in scholarly research and insights into the use of large language models for classifying scientific software citation intent. Building on prior work in this research space, we offer an aggregated and normalized corpus that can be used to train and evaluate the performance of machine learning models tasked on the classification of mentions, usage, and creation of software in text. We present, to the best of our knowledge, the first study on using large language models on predicting software citation intent. The identification of these entities contributes to the link between research software and scientific publications. We believe this work establishes a foundational framework for exploring the under-examined area of studying scientific software citation intent.

Acknowledgements. The authors would like to acknowledge the Chan Zuckerberg Initiative for hosting the Mapping the Impact of Research Software in Science hackathon that led to the development of this work. We would also like to thank Michaela Torkar for help with curating the CZI validation dataset and valuable expert feedback. Kara Moraw was supported by the UK Research Councils through grant EP/S021779/1.

Appendix

(See Figs. 3, 4 and 5).

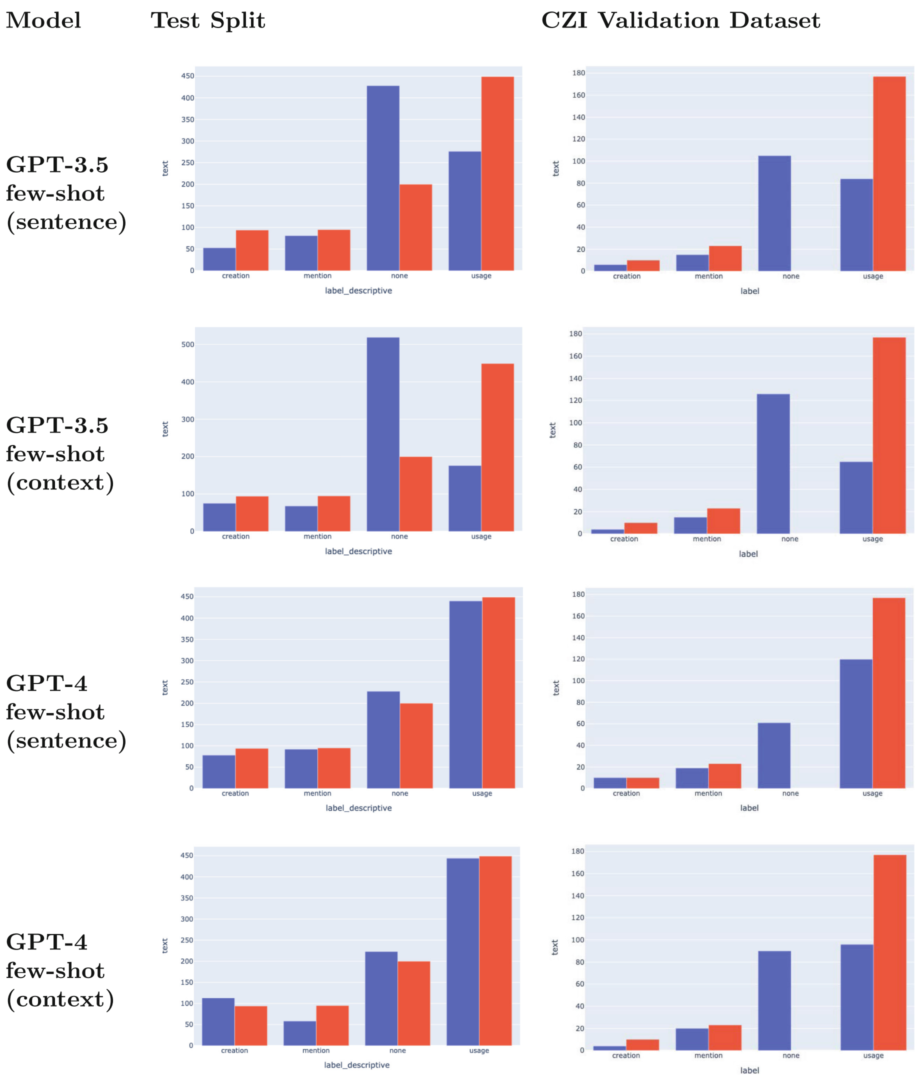

Fig. 3. GPT-3.5/GPT-4 few-shot Comparison of True and Predicted Labels Distributions for few-shot GPT-3.5/GPT-4 models, trained both on the sentence and the full context. Y axis represents the counts, and X axis the label categories. pred_gpt3.5/4 true

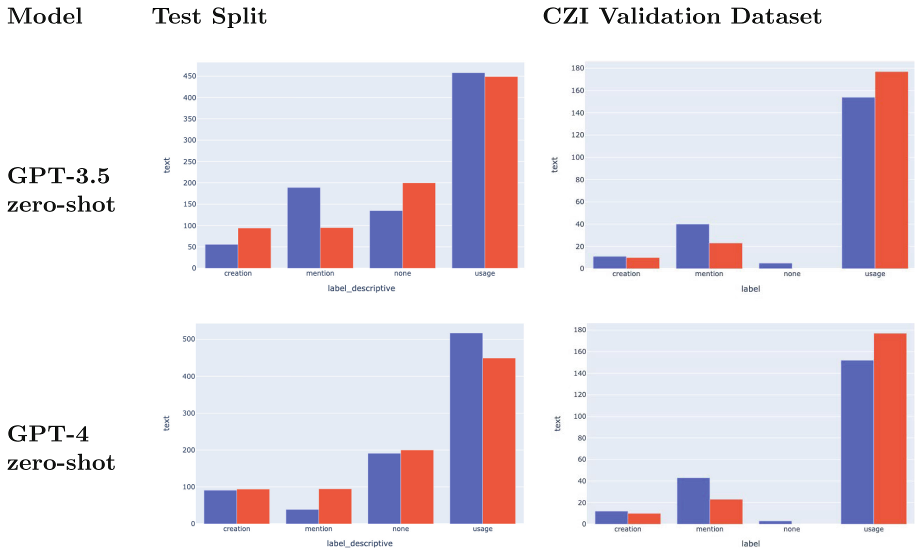

Fig. 4. GPT-3.5/GPT-4 zero-shot Comparison of True and Predicted Labels Distributions for zero-shot GPT-3.5/GPT-4 models. Y axis represents the counts, and X axis the label categories. pred_gpt3.5/4 true

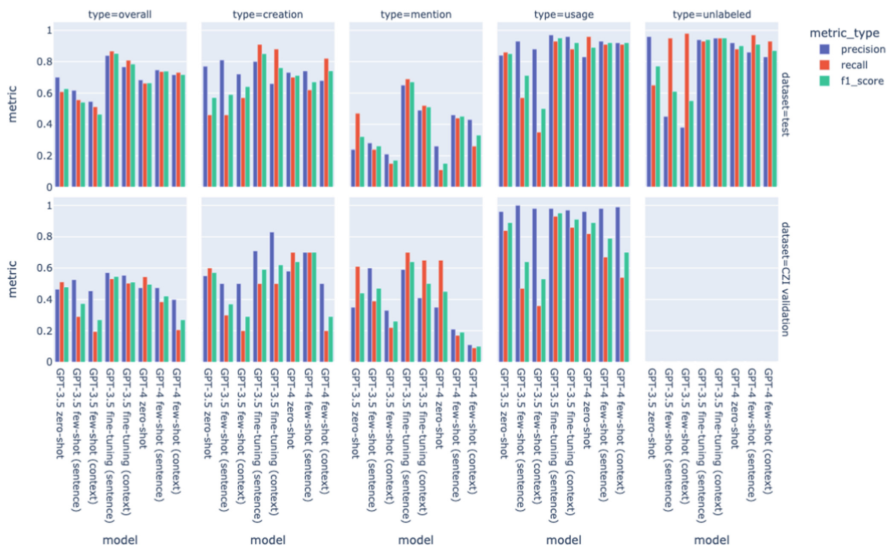

Fig. 5. Performance of GPT-3.5 and GPT-4 models on the test split and the CZI validation dataset. Blue bars correspond to Precision, red bars to Recall and green bars to the F1 score. First row corresponds to results on the test split of the training data (0.2 of the training data. Second row corresponds to the CZI validation dataset. The first column shows overall model performance, aggregating the labels in a macro fashion. Each subsequent column represents model performance for that particular intent class. Note that we do not have any 'unlabeled' sentences in the CZI validation dataset, which is why all metrics will be 0.

References

1. Ammar, W., et al.: Construction of the literature graph in semantic scholar. arXiv preprint arXiv:1805.02262 (2018)
2. Barker, M., et al.: Introducing the fair principles for research software. Sci. Data **9**(1), 622 (2022)
3. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3615–3620. Association for Computational Linguistics, Hong Kong, China (2019). https://doi.org/10.18653/v1/D19-1371. https://aclanthology.org/D19-1371
4. Bensman, S.J.: Garfield and the impact factor: the creation, utilization, and validation of a citation measure. Ann. Rev. Inf. Sci. Technol. (ARIST) **42** (2008)
5. Bird, S., et al.: The ACL anthology reference corpus: a reference dataset for bibliographic research in computational linguistics. In: LREC (2008)
6. Chassanoff, A., Altman, M.: Curation as "interoperability with the future": preserving scholarly research software in academic libraries. J. Am. Soc. Inf. Sci. **71**(3), 325–337 (2020)
7. Cohan, A., Ammar, W., Van Zuylen, M., Cady, F.: Structural scaffolds for citation intent classification in scientific publications. arXiv preprint arXiv:1904.01608 (2019)
8. Cronin, B.: The need for a theory of citing. J. Documentation **37**(1), 16–24 (1981)
9. Cronin, B.: The citation process. Role Significance Citations Sci. Commun. **103** (1984)
10. Crouch, S., et al.: The software sustainability institute: changing research software attitudes and practices. Comput. Sci. Eng. **15**(6), 74–80 (2014)
11. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding (2019)
12. Du, C., Cohoon, J., Lopez, P., Howison, J.: Softcite dataset: a dataset of software mentions in biomedical and economic research publications. J. Am. Soc. Inf. Sci. **72**(7), 870–884 (2021)
13. Du, C., Cohoon, J., Priem, J., Piwowar, H., Meyer, C., Howison, J.: Citeas: better software through sociotechnical change for better software citation. In: Companion Publication of the 2021 Conference on Computer Supported Cooperative Work and Social Computing, pp. 218–221 (2021)
14. Easterbrook, S.M.: Open code for open science? Nat. Geosci. **7**(11), 779–781 (2014)
15. W. Foundation: Wikimedia downloads. https://dumps.wikimedia.org
16. Garfield, E.: "Science citation index"—a new dimension in indexing: this unique approach underlies versatile bibliographic systems for communicating and evaluating information. Science **144**(3619), 649–654 (1964)
17. Garfield, E.: Is citation analysis a legitimate evaluation tool? Scientometrics **1**, 359–375 (1979)
18. Gu, Y., et al.: Domain-specific language model pretraining for biomedical natural language processing. ACM Trans. Comput. Healthcare **3**(1), 1–23 (2021)
19. Horsfall, D., Cool, J., Hettrick, S., Pisco, A.O., Hong, N.C., Haniffa, M.: Research software engineering accelerates the translation of biomedical research for health. Nat. Med. 1–4 (2023)
20. Howison, J., Bullard, J.: Software in the scientific literature: problems with seeing, finding, and using software mentioned in the biology literature. J. Am. Soc. Inf. Sci. **67**(9), 2137–2155 (2016)

21. Istrate, A.M., et al.: SoftwareImpactHackathon2023: Software Citation Intent (2023). https://github.com/karacolada/SoftwareImpactHackathon2023_SoftwareCitationIntent
22. Istrate, A.M., Li, D., Taraborelli, D., Torkar, M., Veytsman, B., Williams, I.: A large dataset of software mentions in the biomedical literature. arXiv preprint arXiv:2209.00693 (2022)
23. Jurgens, D., Kumar, S., Hoover, R., McFarland, D., Jurafsky, D.: Measuring the evolution of a scientific field through citation frames. Trans. Assoc. Comput. Linguist. **6**, 391–406 (2018)
24. Kaplan, N.: The norms of citation behavior: prolegomena to the footnote. Am. Doc. **16**(3), 179–184 (1965)
25. Latour, B.: Science in Action: How to Follow Scientists and Engineers Through Society. Harvard University Press (1987)
26. Li, K., Chen, P.Y., Yan, E.: Challenges of measuring software impact through citations: an examination of the lme4 R package. J. Informet. **13**(1), 449–461 (2019)
27. Li, K., Yan, E.: Co-mention network of R packages: scientific impact and clustering structure. J. Informet. **12**(1), 87–100 (2018)
28. Li, K., Yan, E., Feng, Y.: How is R cited in research outputs? Structure, impacts, and citation standard. J. Informet. **11**(4), 989–1002 (2017)
29. Merton, R.K.: The Sociology of Science: Theoretical and Empirical Investigations. University of Chicago Press (1973)
30. Moravcsik, M.J.: Citation context classification of a citation classic concerning citation context classification. Soc. Stud. Sci. **18**(3), 515–521 (1988)
31. Nicholson, J.M., et al.: Scite: a smart citation index that displays the context of citations and classifies their intent using deep learning. Quant. Sci. Stud. **2**(3), 882–898 (2021)
32. OpenAI: Models - OpenAI API. https://platform.openai.com/docs/models/gpt-3-5-turbo
33. OpenAI: GPT-4 technical report. arXiv preprint arXiv:2303.08774 (2023)
34. Pan, X., Yan, E., Cui, M., Hua, W.: Examining the usage, citation, and diffusion patterns of bibliometric mapping software: a comparative study of three tools. J. Informet. **12**(2), 481–493 (2018)
35. Park, H., You, S., Wolfram, D.: Informal data citation for data sharing and reuse is more common than formal data citation in biomedical fields. J. Am. Soc. Inf. Sci. **69**(11), 1346–1354 (2018)
36. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter (2020)
37. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: Somesci-a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4574–4583 (2021)
38. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: The role of software in science: a knowledge graph-based analysis of software mentions in pubmed central. PeerJ Comput. Sci. **8**, e835 (2022)
39. Smith, A.M., Katz, D.S., Niemeyer, K.E.: Software citation principles. PeerJ Comput. Sci. **2**, e86 (2016)
40. Symons, J., Alvarado, R.: Can we trust big data? applying philosophy of science to software. Big Data Soc. **3**(2), 2053951716664747 (2016)
41. Teufel, S., Siddharthan, A., Tidhar, D.: Automatic classification of citation function. In: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, pp. 103–110 (2006)

42. Wheeler, D.L., et al.: Database resources of the national center for biotechnology information. Nucleic Acids Res. **36**(suppl_1), D13–D21 (2007)
43. Zhang, G., Ding, Y., Milojević, S.: Citation content analysis (CCA): a framework for syntactic and semantic analysis of citation content. J. Am. Soc. Inform. Sci. Technol. **64**(7), 1490–1503 (2013)
44. Zhu, Y., et al.: Aligning books and movies: towards story-like visual explanations by watching movies and reading books. In: The IEEE International Conference on Computer Vision (ICCV) (2015)

RepoFromPaper: An Approach to Extract Software Code Implementations from Scientific Publications

Aleksandar Stankovski(✉) and Daniel Garijo

Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
a.stankovski@alumnos.upm.es, daniel.garijo@upm.es

Abstract. An increasing amount of scientists link to their research software code implementations in their academic publications in order to support the reusability of their results. However, research papers usually contain many code links (e.g., from reused tools or existing competing efforts) making it challenging to automatically establish clear links between papers and their corresponding implementations. This paper presents RepoFromPaper, an approach for automatically extracting the main code implementation associated with a research paper, based on the context in which that link is mentioned. Our approach uses fine-tuned language models to retrieve the top candidate sentences where a code implementation may be found, and uses custom heuristics to link candidate sentences back to their corresponding URL (footnote, reference or full-text mention). We evaluated RepoFromPaper on 150 research papers, obtaining an F1 score of 0.94. We also run our approach on nearly 1800 papers from the CS.AI Arxiv category, discovering 604 paper-repository links and making them available to the community.

Keywords: Information extraction · Research Software · Software repository · Open Science

1 Introduction

Research Software, i.e., the *source code files, algorithms, scripts, computational workflows and executables that were created during the research process* [2] is becoming recognized as a first class citizen in scientific curricula.[1] In order to support the results described in academic publications, scientists often include a link to a code repository (e.g., GitHub, Gitlab) with their technical implementations details.

While efforts have been made by the scientific community to establish principles [16] and formats for software citation [5], detecting the code repository link associated with a publication has two main challenges. First, authors often cite research software inconsistently, employing diverse formats and locations such as

[1] https://sfdora.org/read/.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 100–113, 2024.
https://doi.org/10.1007/978-3-031-65794-8_7

full-text repository mentions (cases where the link is written in the paragraphs), footnotes, or references to refer to a software component [8]. Second, a publication may contain several code repository links (from tools that are reused, or competing with the proposed approach) making it challenging to automatically detect the right code implementation.

This paper introduces a methodology designed to address these challenges by automatically extracting the software implementation repository link associated with a research paper, based on the context in which the link is mentioned. The core contributions of our work include:

1. **Training and validation datasets** of labeled sentences designed to fine-tune and evaluate our approach [17]. The training dataset includes 61 research papers related to software engineering available on the PapersWithCode[2] platform. The validation dataset includes 150 software engineering research articles extracted from Arxiv. Both datasets encompass various types of implementation mention sentences to cover the diverse ways authors reference the implementation repository.
2. **RepoFromPaper**[3], a tool to automatically extract the code implementation repository from a research paper, including PDF-to-Text conversion, sentence extraction, sentence classification and link search, as well as three fine-tuned models.
3. **The results** of the application of our approach on nearly 1800 Arxiv research papers, capturing links between research papers and their software implementations [20].

The rest of the paper is structured as follows. Section 2 describes related work efforts, while Sect. 3 describes the steps followed by RepoFromPaper to detect implementation links. Section 4 describes the metrics used in our evaluation and Sect. 5 presents our assessment results on 150 papers. Next, Sect. 6 describes how we applied our results to nearly 1800 papers, Sect. 7 discusses the limitations of our approach and Sect. 8 concludes the paper.

2 Related Work

The landscape of research papers mentioning software is vast and continually expanding. Platforms such as PapersWithCode actively promote the citation and linking of software source code in research papers. The FORCE11 Software Citation Working Group[4] has put forth software citation principles [15], and efforts from Katz et al. have analysed software citation implementation challenges [9], software citation in theory and practice [10], as well as provided a software citation guide [11] for researchers. These initiatives highlight the importance of proper software citation in research.

[2] https://paperswithcode.com/.
[3] https://github.com/StankovskiA/RepoFromPaper.
[4] https://force11.org/group/software-citation-working-group/.

Researchers have taken on the challenge of automatically detecting software mention intent, as exemplified by the work available on GitHub[5] which uses data from SoftCite (Du et al., 2021) [6] and SoMeSci (Schindler et al., 2020) [14]. Their focus is on classifying software mentions based on intent, categorizing them into "Creation" (i.e., a tool is proposed in a paper), "Usage" (i.e., a tool is used in a publication to conduct research), and "Related" (i.e., a tool is mentioned as a related competitive effort). While this work shares similarities with ours, they aim to detect software tool mentions and understand their intent. Instead, our objective is to identify the repository code implementations associated with a research publication.

Lin et al. [12] present a methodology for automatically extracting software source code URLs, reporting a high model accuracy of 0.939. However, their approach has three limitations. First, their methodology does not consider URLs in references. Second, their approach relies on GROBID,[6] a PDF parser that structures nicely the contents of a paper, but may overlook footnotes. Third, their reliance on a regex search for sentences containing URLs may overlook indirect mentions, such as those within references or footnotes.

Finally, our previous work[7] [7] focuses on identifying bi-directional URL mentions between a paper and a repository, i.e., papers which mention a source code repository, and the repository reciprocates by mentioning the paper. While this approach holds the potential for high precision, it falls short in capturing unidirectional repository mentions (i.e., those publications that refer to a code repository but without a link back to that paper), which we aim to address in this work.

3 RepoFromPaper: Methodology

Our approach consists of six steps. Figure 1 provides an overview of the data flow within the pipeline, starting with an input PDF file and concluding with either the discovery of a relevant code implementation link or an empty response. We elaborate on each step of the pipeline below, providing insights into the rationale, processes, and integration of essential components within our methodology.

3.1 PDF-to-Text Conversion

We start with the conversion of PDFs of research papers into text using the Apache Tika PDF reader[8], known for its speed and accurate extraction of text. This initial step enables subsequent text-based processing, facilitating the application of heuristic rules for sentence extraction and input into the models for identifying repository implementation mentions. Although alternatives like

[5] https://github.com/karacolada/SoftwareImpactHackathon2023_SoftwareCitationIntent.

[6] https://github.com/kermitt2/grobid/.

[7] https://github.com/SoftwareUnderstanding/RSEF.

[8] https://github.com/chrismattmann/tika-python.

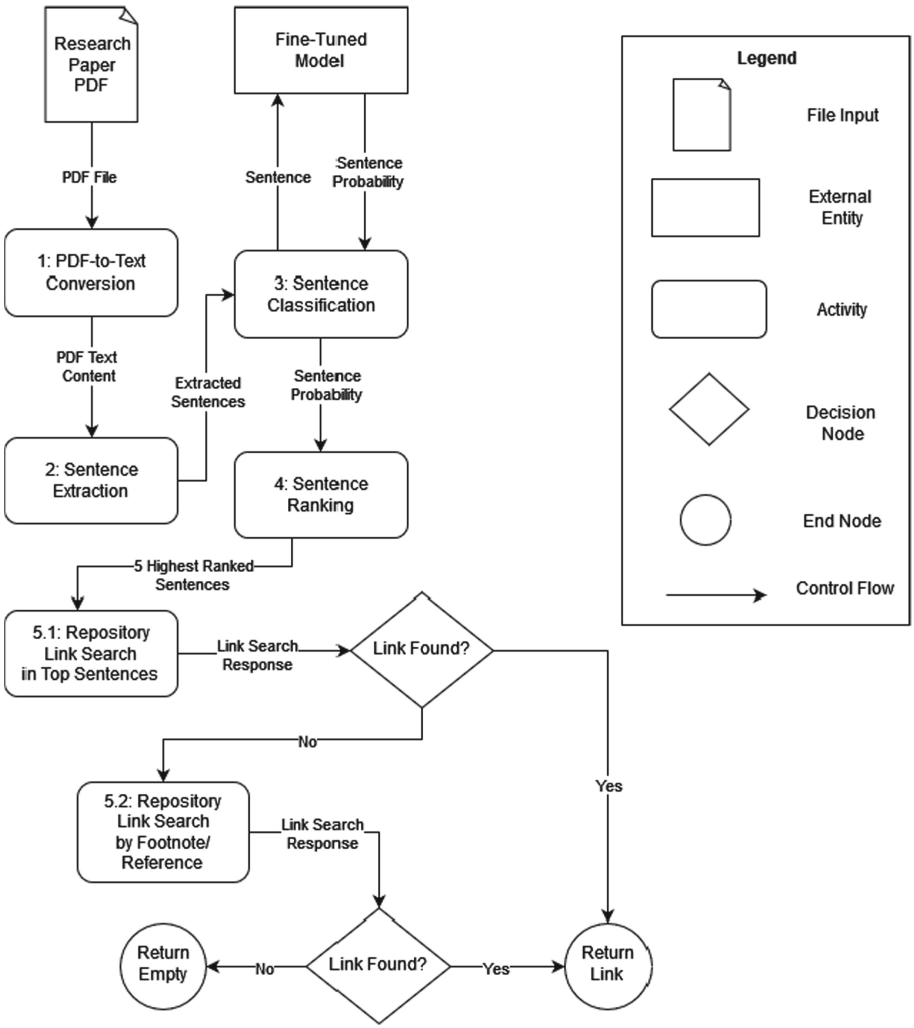

Fig. 1. RepoFromPaper Methodology Flowchart

GROBID are available, we selected Apache Tika due to its robust performance in handling various PDF formats, processing speed, and accurate representation of footnote content, which is a critical factor in our methodology. The source code for our package, including the integration with Apache Tika, is available online under an MIT license[9], providing transparency and reproducibility for our approach.

[9] https://github.com/StankovskiA/RepoFromPaper.

3.2 Sentence Extraction

The Sentence Extraction phase preprocesses the input PDF text and extracts complete sentences for subsequent analysis. Within this phase, various functions contribute significantly to refining and organizing the PDF text, effectively segmenting it into well-formatted sentences (e.g., removing end-of-line dashes) each ready for input into the fine-tuned models to classify them. The decision to extract sentences instead of paragraphs is driven by our findings that paragraphs introduce significant noise, while sentence-level extraction enhances model learning by focusing on key information.

Text cleaning is crucial for ensuring uniformity and clarity in the extracted sentences. This involves removing newline characters, word breaks, extra white space, and inconsistencies in links. Additionally, we extract reference number - reference text pairs, as well as footnote number - footnote text pairs if the footnote text is a link. This approach enables us to utilize this information effectively in the subsequent link search step.

Due to the diverse formats and layouts of research papers, sentences are frequently split across multiple lines, possibly spanning different pages and encountering footnotes in between, oftentimes including hyphenation at the end of lines as a line break. To address this issue, we consolidate fragmented sentences by considering factors such as sentence beginning and ending, new line characters and white spaces, and hyphenation, ensuring the formation of cohesive and complete sentences from fragmented text.

3.3 Sentence Classification

In this section, we delve into the process of classifying sentences extracted from research papers in order to identify implementation links. The classification is performed using fine-tuned BERT, SciBERT and RoBERTa models, chosen for their effectiveness in processing textual data.

Training Data and Fine-Tuning. We approach the problem of distinguishing between implementation mentions sentences and non-implementation sentences as a text classification problem, more specifically a sentence classification problem. Based on the context of the sentence, we aim to assess whether it is proposing an implementation of the paper. For this purpose, we assembled a training corpus [17] consisting of sentences extracted from 61 research papers related to software engineering sourced from the PapersWithCode platform. These sentences encompass various ways of mentioning repositories that authors tend to use, including full-text mentions, footnotes, and references. Each sentence in the corpus was annotated with a binary label indicating whether it mentions an implementation repository (1) or not (0). This annotation process involved one annotator initially labeling the sentences, followed by a review by another annotator. Any conflicts about the annotations were resolved through discussion until agreement was reached, particularly regarding whether to include an implementation mention if it was lacking clear context.

The models were fine-tuned using a binary sequence classification setup, where the objective was to classify sentences as either implementation mentions or non-implementation sentences. We employed the 'bert-base-uncased' model for BERT [4], 'allenai/scibert_scivocab_uncased' for SciBERT [1] and the 'roberta-base' model for RoBERTa [13], initializing them with pre-trained weights. The fine-tuned BERT[10], SciBERT[11] and RoBERTa[12] models are available on the HuggingFace platform.[13]

3.4 Sentence Ranking

Using our fine tuned models, we classify all the sentences available in an input publication. The models then predict the probability of each sentence belonging to the class of implementation or non-implementation sentences. Based on these probabilities, the sentences are ranked, allowing us to identify the sentences which are most likely to contain implementation mentions. We then retrieve the five sentences with the highest probability as candidates to find implementation links. Our rationale for selecting the top five sentences is that, while the model predicts the probability of each sentence belonging to class 1, we observed that the correct proposal sentence may not consistently have the highest probability. To mitigate this, we opt for a more inclusive strategy, extracting the top five sentences based on their probability scores. This increased the chances of capturing the correct proposal sentence from 80% to 94%, accounting for potential variations in model predictions.

3.5 Repository Link Search

The final step of our methodology aims to link the top ranked sentences with the corresponding link containing the code implementation. We divide this step in two stages:

1. **Repository link search in top sentences**: We use regular expressions to retrieve any code repository links (GitHub, GitLab) that may be found within the candidate sentence itself. As described in Howison & Bullard (2015) [8], inline references are among the common practices for citing software in publications. The rationale behind this step is to establish a direct connection between the predicted proposal sentences and their corresponding repositories. By searching for links within the sentences with the highest probability of being implementation mentions, we aim to streamline the extraction process and efficiently link research papers to their associated repositories. If multiple repository links are present in the top-ranked sentences, we return the first identified link.

[10] https://huggingface.co/oeg/BERT-Repository-Proposal.
[11] https://huggingface.co/oeg/SciBERT-Repository-Proposal.
[12] https://huggingface.co/oeg/RoBERTa-Repository-Proposal.
[13] https://huggingface.co/.

2. **Repository link search in footnotes and references**: This step aims to broaden the search scope by examining the sentences that may contain relevant numbers or special characters representing footnote or reference numbers. These characters are indicative of footnote or reference numbers commonly associated with research papers. The order of appearance of these numbers is retained to prioritize classified sentences with higher probability. Once potential footnote or reference numbers are identified, our methodology proceeds to search for candidate sentences containing these numbers. To maximize the chances of finding the correct sentence, we consider both the original appearance of the numbers and variations with or without brackets, particularly when reference numbers are enclosed in brackets.

4 Evaluation Methods

To assess the performance of our methodology, we employ two main evaluation methods, each providing valuable insights into the effectiveness of our approach.

4.1 Mean Reciprocal Rank (MRR)

Mean Reciprocal Rank (MRR) [3] serves as a key evaluation metric for gauging the individual performance of the fine-tuned models. MRR is calculated based on the position of the correct proposal sentence within the list of the top five highest ranked sentences. This metric offers an understanding of how well the models rank the correct proposal sentence relative to other potential candidates. A higher MRR indicates better model performance in isolating and prioritizing the most relevant sentences.

The formula for Mean Reciprocal Rank is given by:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\text{rank}_i} \tag{1}$$

where N is the number of instances, and rank_i is the position of the correct proposal sentence in the ranked list for the ith instance.

4.2 Precision, Recall, and F1 Score

To comprehensively evaluate the overall performance of RepoFromPaper, we employ precision, recall, and F1 score metrics. These metrics are calculated based on the following definitions:

- **True Positive (TP):** The pipeline returns a correct repository implementation link for the target paper.
- **False Positive (FP):** The pipeline returns an incorrect repository implementation link for the target paper.
- **False Negative (FN):** The pipeline fails to identify any repository implementation link, but one is present.

– **True Negative (TN):** The model finds no repository implementation link, and there is no link present in the paper.

Precision measures the accuracy of the identified repository links, recall assesses the ability of the methodology to capture all relevant links, and the F1 score provides a balanced evaluation considering both precision and recall.

4.3 Training and Testing Corpora

In the training corpus for our models, we included 75 implementation sentences and approximately 2500 non-implementation sentences from 61 research papers sourced from the PapersWithCode platform. To evaluate the performance of our method, we assembled a separate evaluation corpus [19] consisting of 150 software engineering research papers obtained from Arxiv.org. These papers were carefully selected to ensure heterogeneity and avoid repetitiveness, representing a diverse range of implementation mention styles, authored by various authors. We manually tagged these papers to create a validation set specifically for evaluating our methodology. Importantly, none of the papers included in this validation set were used for training the models, ensuring the integrity of our evaluation process.

We utilized the entire text contents of these papers in our evaluation process. To ensure merit and diversity of repository implementation types, both the training corpus and evaluation set are consisted of papers that encompass the three main mention types: "Full-text" (i.e., inline URLs), "Footnote" and "Reference" mentions. Figure 2 shows the number of mention types present in the training and evaluation set, which follow a similar distribution.

Figure 3 shows the frequency distribution of repository links found in the papers. This distribution sheds light on the challenges associated with automatically identifying and extracting the correct implementation repository links from research papers, as a nearly half of the papers have two or more code links.

Finally, papers that only used hyperlinks to link the implementation repositories were excluded from the set as the link text was not present in the PDF text.

5 Results

Table 1 describes the results obtained from the evaluation of our methodology using the Fine-Tuned BERT, SciBERT, and RoBERTa models over our test set. Our results present an accurate identification of implementation mentions within research papers (0.94 F1), indicating the efficacy of employing fine-tuned language models. The SciBERT model exhibits superior learning capabilities, as reflected in its elevated precision, recall, and F1 score. This implies a more sophisticated grasp of implementation mentions. The model's enhanced performance is in line with its advanced architecture, emphasizing the significance of employing state-of-the-art language models for intricate tasks.

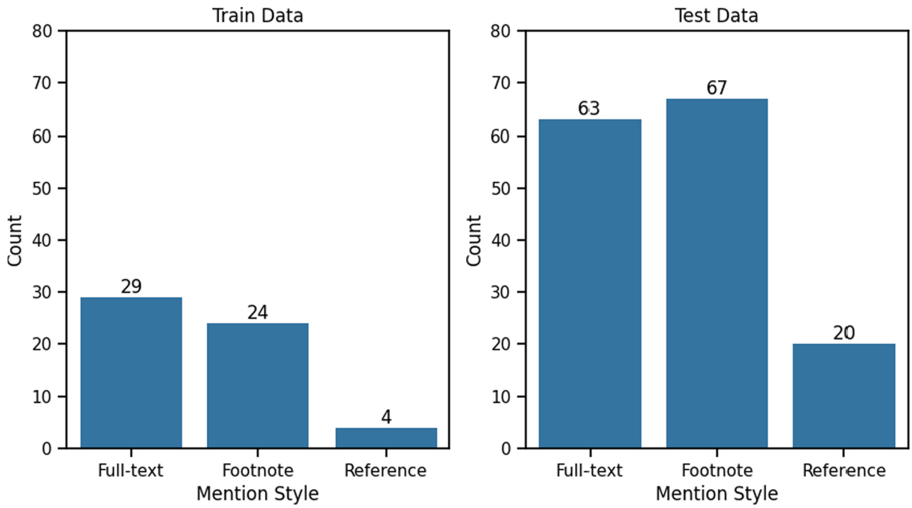

Fig. 2. Distribution of implementation links and their style (within the full text, as a footnote or in a reference) in training and testing corpora

Table 1. Evaluation results for the fine-tuned models

Model	Precision	Recall	F1 Score	MRR
BERT	0.864	0.871	0.867	0.55
RoBERTa	0.942	0.929	0.936	0.753
SciBERT	**0.944**	**0.95**	**0.947**	**0.85**

These results provide insights into the performance of the fine-tuned models, as well as the overall methodology, in identifying implementation mentions within the extracted sentences. To better understand the significance of the results, we compare our results in the test set against a baseline method achieved by selecting the most frequent code repository (using a regular expression) in a publication (the first code repository is returned if all code links appear just once).

The comparison between the regex baseline and our best performing model can be seen in Table 2. Our method outperforms the baseline by achieving 15% higher precision and 6% increase in F1 Score, while having a 5% lower recall. We consider this an adequate trade off for the problem at hand.

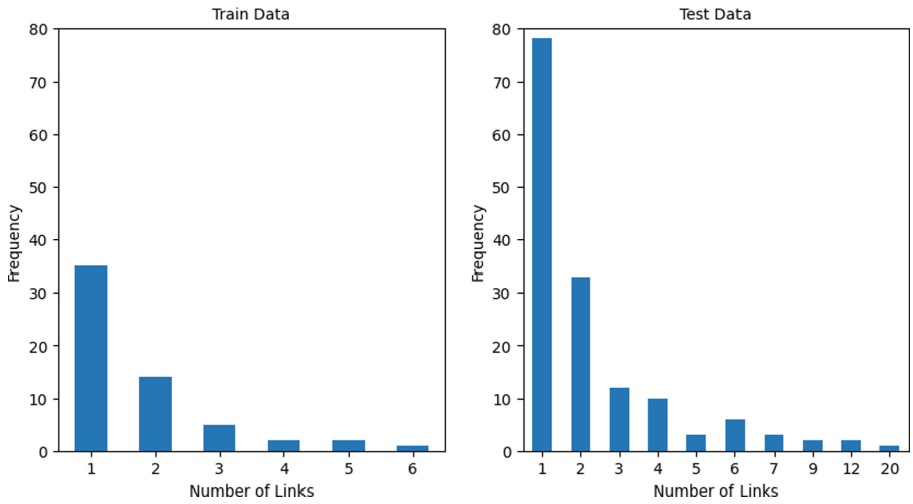

Fig. 3. Number of code repository links in training and test papers

Table 2. Comparison with baseline performance

Model	Precision	Recall	F1 Score
Regex Baseline	0.793	1	0.88
SciBERT	**0.944**	**0.95**	**0.947**

6 A Corpus of Papers and Their Corresponding Implementations

We packaged our method into RepoFromPaper[14] [21] and applied it on nearly 1800 research papers submitted in the years 2022 and 2023 found on the Artificial Intelligence section on Arxiv.org.[15] We applied our method both with the fine-tuned RoBERTa and SciBERT models over the papers. The leading performance of the SciBERT model approach was once again confirmed as it led to detection and extraction of 604 implementation repository links, while the RoBERTa model approach detected 585. We make the outcomes of this application of our method public [20].

Finally, we compared our approach using the SciBERT model against a method that detects bi-directional links between papers and code repositories [7] on 150 research papers from the Software Engineering category on Arxiv.org[16]. These 150 articles were selected randomly, from the year 2023 (selected papers may or may not include a code implementation link). In summary, the number

[14] RepoFromPaper is available at https://github.com/StankovskiA/RepoFromPaper.
[15] https://arxiv.org/list/cs.AI/recent.
[16] https://arxiv.org/list/cs.SE/recent.

of implementation links found only by the bi-directional approach was 4, the number of implementation links found only by RepoFromPaper was 41 and the number of implementation links found by both approaches was 16. In 89 publications none of the approaches found an implementation link. The results of the comparison are available online [18].

Our findings show that our approach is able to extract 25% more implementation repository links when compared to the bi-directional approach. Furthermore, we observe an expected overlap in the extracted links between both approaches, i.e., both approaches successfully extract the same implementation link. However, we also observe unique links extracted by each method. The reason for this divergence is twofold. On the one hand, while our approach is able to detect uni-directional links between a research paper and a repository, the bi-directional approach requires the repository to also point back to the paper, therefore missing these links. On the other hand, the bi-directional approach is able to return multiple confirmed links (many authors separate code implementation and evaluation results in different repositories) whereas our approach currently returns only one implementation repository link. The ability of both approaches to detect unique links suggests that they can complement each other, aiming to extract as many correct implementation repository links from the research paper as possible.

7 Discussion

Our approach produces high evaluation results, presents several limitations. Firstly, the model training dataset was limited to the 75 implementation mention sentences present in 61 research papers, which may restrict its ability to generalize in other domains. Additionally, when the proposal link was not found in the top-ranked sentences, our methodology searched for footnote or reference mentions, but the abundance of numbers in some sentences introduced potential noise. Moreover, the order of sentences containing footnote/reference numbers posed complexity, occasionally leading to false positive links. Another limitation is that our methodology currently returns only one implementation link, even when multiple correct links may exist in a publication. Our approach also does not extract links embedded as hyperlinks, defined in tables or present in metadata. Despite these challenges and limitations, our methodology demonstrates robust performance by effectively detecting and extracting implementation repository links from PDFs of research papers, irrespective of their formats or the diverse ways in which implementation repositories are mentioned.

8 Conclusions and Future Work

In this paper we introduced RepoFromPaper, a methodology and tool for the automatic extraction of implementation repository links from research papers. Our evaluation demonstrates promising results, showcasing the efficacy of using fine-tuned language models. The achieved precision, recall, and F1 scores, particularly with the SciBERT model, signify a considerable success in identifying implementation mentions within research papers.

However, while our approach has shown good performance, there remain areas for improvement, particularly in the pre-processing step of converting PDFs to text. Enhancements in this phase may lead to more accurate sentence extraction, reducing noise and further refining the pipeline's effectiveness.

Moving forward, there are several avenues for future work and improvements. First, expanding the training dataset and fine-tuning the models with a more extensive range of proposal mention variations may enhance their ability to recognize diverse ways of mentioning repositories. Second, investigating more advanced PDF-to-Text conversion techniques may contribute to better sentence extraction, overcoming challenges posed by varied PDF formats. Finally, applying our method on research papers from different domains will help us generalizing our approach, gaining better insights into the current practices regarding code and data repository mentions in disciplines other than Computer Science, ranging from Astronomy to Geology or Computational Biology.

Acknowledgements. This work is supported by the Madrid Government (Comunidad de Madrid, Spain) under the Multiannual Agreement with Universidad Politécnica de Madrid in the line Support for R&D projects for Beatriz Galindo researchers, in the context of the VPRICIT, and through the call Research Grants for Young Investigators from Universidad Politécnica de Madrid. The authors would also like to acknowledge European Union's Horizon Europe Programme under GA 101129744 – EVERSE – HORIZON-INFRA-2023-EOSC-01-02.

References

1. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. arXiv preprint arXiv:1903.10676 (2019)
2. Chue Hong, N.P., et al.: FAIR Principles for Research Software (FAIR4RS Principles) (2022). https://doi.org/10.15497/RDA00068
3. Craswell, N.: Mean reciprocal rank. In: Liu, L., Özsu, M.T. (eds.) Encyclopedia of Database Systems. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-39940-9_488
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
5. Druskat, S., et al.: Citation File Format (2021). https://doi.org/10.5281/zenodo.5171937
6. Du, C., Cohoon, J., Lopez, P., Howison, J.: Softcite dataset: a dataset of software mentions in biomedical and economic research publications **72**(7), 870–884 (2021). https://doi.org/10.1002/asi.24454

7. Garijo, D., Arroyo, M., Gonzalez, E., Treude, C., Tarocco, N.: Bidirectional paper-repository tracing in software engineering. In: 21st International Conference on Mining Software Repositories (MSR 2024). ACM, Cham (2024). https://doi.org/10.1145/3643991.3644876
8. Howison, J., Bullard, J.: Software in the scientific literature: problems with seeing, finding, and using software mentioned in the biology literature. J. Am. Soc. Inf. Sci. **67**(9), 2137–2155 (2016). https://doi.org/10.1002/asi.23538
9. Katz, D.S., et al.: Software citation implementation challenges. arXiv preprint arXiv:1905.08674 (2019)
10. Katz, D.S., Chue Hong, N.P.: Software citation in theory and practice. In: Davenport, J.H., Kauers, M., Labahn, G., Urban, J. (eds.) ICMS 2018. LNCS, vol. 10931, pp. 289–296. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96418-8_34
11. Katz, D.S., et al.: Recognizing the value of software: a software citation guide. F1000Research **9** (2020). https://doi.org/10.12688/f1000research.26932.1
12. Lin, J., Wang, Y., Yu, Y., Zhou, Y., Chen, Y., Shi, X.: Automatic analysis of available source code of top artificial intelligence conference papers. Int. J. Softw. Eng. Knowl. Eng. **32**(07), 947–970 (2022). https://doi.org/10.1142/S0218194022500358
13. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
14. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: Somesci- a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM 2021, pp. 4574–4583. Association for Computing Machinery, New York (2021). https://doi.org/10.1145/3459637.3482017
15. Smith, A.M., Katz, D.S., Niemeyer, K.E.: Software citation principles. PeerJ Comput. Sci. **2**, e86 (2016). https://doi.org/10.7717/peerj-cs.86
16. Smith, A.M., Katz, D.S., Niemeyer, K.E.: FORCE11 software citation working group: software citation principles. PeerJ Comput. Sci. **2**, e86 (2016). https://doi.org/10.7717/peerj-cs.86
17. Stankovski, A.: PapersWithCode-Corpus Repository Proposal Sentences Training Dataset (2024). https://doi.org/10.5281/zenodo.10701846
18. Stankovski, A.: Repofrompaper comparison with Bidir method (2024). https://doi.org/10.5281/zenodo.10988947
19. Stankovski, A.: RepoFromPaper Repository Implementation Link Testing Dataset (2024). https://doi.org/10.5281/zenodo.10980368
20. Stankovski, A.: RFP output on CSAI papers from 2022/23 (2024). https://doi.org/10.5281/zenodo.10975879
21. Stankovski, A.: Stankovskia/repofrompaper: v1.0.1 (2024). https://doi.org/10.5281/zenodo.10988913

Automated Extraction of Research Software Installation Instructions from README Files: An Initial Analysis

Carlos Utrilla Guerrero[1,2(✉)], Oscar Corcho[1], and Daniel Garijo[1]

[1] Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
carlos.utrilla.guerrero@alumnos.upm.es, {ocorcho,dgarijo}@upm.es
[2] Research Data and Software (RDS) Department in Delft University of Technology, Delft, the Netherlands

Abstract. Research Software code projects are typically described with a README files, which often contains the steps to set up, test and run the code contained in them. Installation instructions are written in a human-readable manner and therefore are difficult to interpret by intelligent assistants designed to help other researchers setting up a code repository. In this paper we explore this gap by assessing whether Large Language Models (LLMs) are able to extract installation instruction plans from README files. In particular, we define a methodology to extract alternate installation plans, an evaluation framework to assess the effectiveness of each result and an initial quantitative evaluation based on state of the art LLM models (`llama-2-70b-chat` and `Mixtral-8x7b-Instruct-v0.1`). Our results show that while LLMs are a promising approach for finding installation instructions, they present important limitations when these instructions are not sequential or mandatory.

Keywords: Research/Scientific Knowledge Graphs · Natural Scientific Language Processing · Information Extraction

1 Introduction

Research Software [5] is becoming increasingly recognized as a means to support the results described in scientific publications. Researchers typically document their software project in code repositories, using README files (i.e., `readme.md`) with instructions on how to install, setup and run their software tools. However, software documentation is usually described in natural language, which makes it challenging to automatically verify whether the installation steps required to make the software project work are accurate or not. While seemingly arbitrary, it can be challenging for researchers to follow instructions from different document standards and make sure they work harmonically and consistently.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 114–133, 2024.
https://doi.org/10.1007/978-3-031-65794-8_8

In this work we aim to address these issues by exploring and assessing the abilities of state of the art Large Language Models (LLMs) to extract installation methods (*Plans*) and their corresponding instructions (*Steps*) from README files. LLMs such as GPT-4 [21] and MISTRAL [12] have been firmly established as state of the art approaches in various natural scientific language processing (NSLP) tasks related to knowledge extraction from human-like scientific sources such as software documentation from public sharing code hosting services. LLMs have also shown promise in following instructions [26] and learning to use tools [25]. However, existing research in the field is still quite novel.

Our goal in this work is twofold: given a README file, we aim to 1) detect all the available *Plans* (e.g., installation methods for different platforms or operative systems) and, 2) for each Plan, detect what steps are required to install a software project, as annotated by the authors. Our contributions[1] include:

1. *PlanStep*, a methodology to extract structured installation instructions from README files;
2. An evaluation framework to assess the ability of LLMs to capture installation instructions, both in terms of *Plans* and *Steps*;
3. An annotated corpus of 33 research software projects with their respective installation plans and steps.

We implement our approach by following our methodology to evaluate two state of the art LLMs (LLaMA-2 [31] and (MIXTRAL [12]) on both installation instruction tasks with our corpus of annotated projects.

The reminder of the paper is structured as follows. Section 2 discusses relevant efforts to ours, while Sect. 3 describes our approach. Section 4 describes our experimental setup and early results, Sect. 5 addresses our limitations and Sect. 6 concludes the paper.

2 Related Work

The goal of extracting relevant information from scientific software documentation forms the foundation of complex knowledge extraction with Natural Language Processing (NLP) models, all of which use machine-learning-based (ML) approaches as basic building blocks [10].

Extracting action sequences from public platforms (e.g. Github, StackOverflow) or README files is an instance of a complex planning tasks class of problems. Remarkably, the field of automated software engineering has rapidly developed novel approaches using LLMs on important problems, for instance, integrating tool documentation [23], detecting action sequence from manuals [18], testing software [39], traceability and generating software specifications [42].

LLMs such as GPT-4 and others follow an architecture using encoder-decoder structure [37], and have been shown to perform well on simple-plans extraction

[1] The code and corpus are publicly available at: https://github.com/carlosug/READMEtoP-PLAN/ [44].

and procedure mining [24], as well as mining to support scientific material discovery [2,38]. The fundamental constraint of multi-step reasoning abilities, however, remains [19,33,34].

In the Knowledge Extraction (KE) field, foundational work builds on general-propose metadata extractor and domain-specific have been successfully applied in a variety of tasks including scientific mentions [6], software metadata extraction [17,32], and scientific knowledge graph creation [15]. The automated planning community has also continued to push the boundaries of approaches that learn how to extract plans [20] and action sequences from text in domain-specific [4,8,13], and general domains (i.e., [18,35]). Recently, [22,43] and [26] have made an impressive advance in the feasibility of connecting LLMs with massive APIs documentation. However, in most cases installation instructions, specifically for plans and steps are absent from the corresponding studies.

Recent work [9,11] has achieved significant improvements in multi-step extraction tasks by using different prompt strategies [34,36]. In these prompt strategies, the number of operations required to extract entities or events from text grows. This makes it more difficult to learn semantics between the inputs as the instructions are not self-actionable, especially when several steps are involved. Early approaches also discussed the missing descriptions in generated plans [30], and composed learning a mapping from natural language instructions to sequences of actions for planning proposes [27]. In this experiment, this is reduced to a number of formal definitions, albeit at cost of reduced effective resolution due to natural language problem, an effect we plan to counteract with improved prompt variations using formal representation [7,18] as described in Sect. 3.6.

To the best of our knowledge, this is the first approach relying entirely on LLMs to extract installation instructions from research software documentation. We focus on eliciting multi-step reasoning by LLMs in a zero-shot configuration. We ask LLMs to extract both the installation plans and step instructions, effectively decomposing the complex task of installing research software into two kinds of sub-problems: 1) extraction of installation methods to capture various ways of installing research software as *Plan*(s) from unstructured documentation, and 2) extraction of installation instructions to identify sequential actions for each method as *Step*(s) (i.e., Step(s) per Plan).

3 *PlanStep*: Extracting Installation Instructions from README Files

In this section we present *PlanStep*, our proposed approach designed to address limitations briefly outlined in Sect. 2. First, we describe the core goal and problem we attempt to solve. Next, we describe the *PlanStep* architecture and building blocks. Finally, we describe the data generation and corpus.

3.1 Classical Planning: Software Installation Instructions

The central objective of planning tasks for an intelligent assistant is to autonomously detect the sequence of steps to execute in order to accomplish a task. In classical planning domain, this procedure relies on a formal representation of the planning domain and the problem instance, encompassing actions, and their desired goals [9].

In our case, a problem instance within the installation instruction activity is illustrated in Fig. 1. This instance features research software with two alternate plans for installation available in the README: `"Install from pip"` and `"Install from source"`. Each plan is defined fairly briefly, but detailed in the corresponding headers of the markdown file. Subsequently, installation steps outlining the requirements for the setup and execution, are displayed. For instance, the *Plan 1* (categorised as `"Package Manager"`) includes *Steps* contains three steps (or actions). *Plan 2*, classified as *"Source"*, involves one step. If we ask an intelligent assistant to autonomously decide on what sequence of steps to execute this software, we might use an LLM to mine its documentation and break down the installation objective into smaller sub-tasks: first detecting the requirements, then identifying the plan available, and finally execute the necessary commands. Many installation procedures may not need planning. For example, the `"Package Manager"` plan usually entails a one-step with code block, showing exactly what others need to type into their install software from a command line. However, in complex installation plans such as `"from Container"` (i.e., `Docker compose-up`, create virtual environments, configure public keys, etc.,..) planning allows the assistant to decide dynamically what steps to take. If we want an assistant to consider a software component and install it following its instructions, the task may be decomposed into different steps: 1) detect alternate plans which are available as installation methods and, 2) for each installation method, detect its corresponding sequence of steps.

3.2 *PlanStep* Methodology

Consistency in the extracted installation methods across different software versions is key for researchers to accurately reproduce experiments, regardless of when and how to access a README file. Therefore, our method aims to consistently connect human-readable instructions to installation *Plans* and *Steps.*

PlanStep receives as input the entire README file, and aims to extract action sequences from text in natural language, representing tasks with two distinct levels of granularity: 1) alternate installation methods (e.g., installation instructions for a separate operative system, installation instructions from source code, etc.), and 2) for each installation method, the sequence of steps associated with it.

Figure 2 depicts the methodology we followed for developing PlanStep, which comprises five stages: the first stage is to collect a set of research software for our study. Second, for each software component in the corpus, we retrieve the link for the code repository, if present, and extract the installation instruction

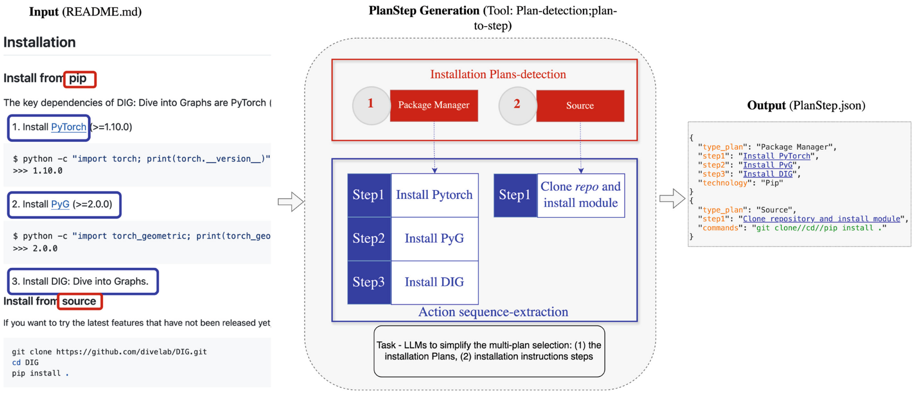

Fig. 1. An example of our experimental approach for *PlanStep*. A research software project includes two installation methods: a simple installation plan *i.e. Package Manager* and a complex one (*i.e. Source*). Each installation method has a different number of steps and configuration details.

text from its README file. Then, we inspect the original README and represent the alternate installation plans for each README in a structured format. Afterwards, for each entry, we prompt Large Language Models in order to detect plans and their corresponding steps. Finally, we design an evaluation framework to assess the quality of our results.

We limit ourselves to tasks that can be characterized as research software installation activities and involve a reasonable or necessary order of steps to be executed, such as manually setting up a software project component, installing additional libraries using package managers, running from isolated containers, or building from source.

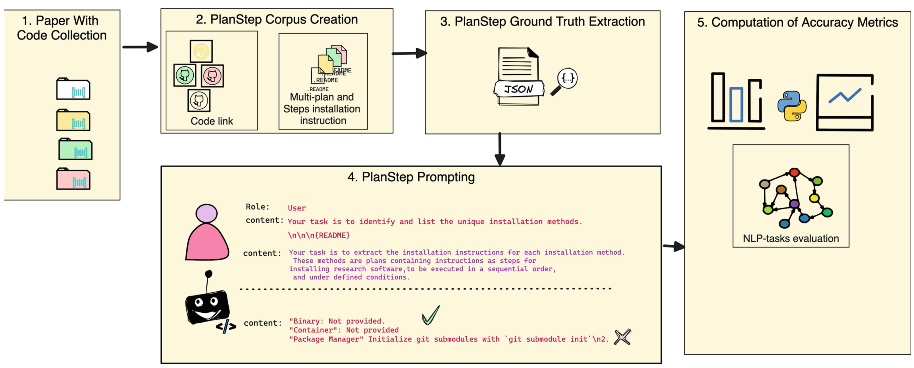

Fig. 2. Overview of the methodology followed to collect research software and design an evaluation framework to assess PlanStep

3.3 *PlanStep* Corpus Creation

To systematically evaluate LLM performance on extracting installation plans with steps across varying setups, complexity, and domains, we started by selecting a corpus of research papers with their code[2] implementations from diverse Machine Learning (ML) areas and across different task categories. For this evaluation, we excluded, however, papers with no link to their public repository available on Github or Gitlab.

All annotations were made by the authors of this paper separately, and subsequently compared until consensus was achieved. We discussed each entry to determine the final set of steps and plans for each research software. Very rarely, agreement on specific properties remained elusive even after evaluation, and these cases were manually resolved through additional discussion. In summary, our corpus has 33 fully active and maintained open-source research software projects.

3.4 Ground Truth Extraction for *PlanStep*

The 33 research software projects in our corpus were selected as study subjects. In a manual co-annotation process, we tasked annotators with identifying both the installation plans and steps associated with each project's README. The installation plans varied in complexity and description style, with some, like `from pip` typically comprising a single-sentence step (excluding requirements), while others, such as 'from source,' included multiple steps, considering various user environments and requirements. Additionally, we defined specific properties for each plan type, taking into account technology-specific support, such as package repositories like `npm` or `PyPI`. Further elaboration on these definitions is provided below:

A. Plan: represents the concept of an installation method available in a README, which is composed of steps, that must be executed in a given order. For instance, a "Source" is an instance of a Plan concept. A README can include one or multiple Plans in the installation instructions section. A brief explanation of the plans and examples is provided in Table 1.

B. Step: represents the concept of a planned action as part of a '*Plan*' to be executed sequentially. It may consist of either a single action or a group of actions. We define a 'Step' based on the original README text, where consecutive actions mentioned together are annotated as one step. For instance, Listing 1.1 illustrates this concept with a simple JSON example. In the example, the authors' original text describes the first step (`Step1`) as `'Clone this repository and install requirements,'` which encompasses two distinct actions: `'Clone this repository'` and `'install requirements.'`. The

[2] We made use of Paper with Code platform: https://paperswithcode.com/ as it links together articles with software repository.

Table 1. Definition of plan types and examples found in our corpus

Plan Type	Short definition	Examples
Binary	Installing by directly downloading and running precompiled files	GitHub releases
Container	Installing by packaging software components and its dependencies using containerization	Docker, Podman, Singularity
Package Manager	Downloading and installing from official repositories	Conda, Homebrew, Pip, npm
Source	Installing by manually compiling original code into machine-readable binaries	Clone from repository, create virtual environment

second step (`Step2`) simply involves one action `‘iRun the container with docker - compose’`

```
{
 "id": "4",
 "name": "utiasASRL/steam_icp",
 "url": "https://raw.githubusercontent.com/utiasASRL/steam_icp/master/README.
     md",
 "plans": [
  {
   "type": "Container",
   "steps": [
    {
     "text": "Clone this repository and install requirements.",
     "seq_order": 1,
     "is_optional": false
    },
    {
     "text": "Run the container with docker-compose",
     "seq_order": 2,
     "is_optional": false
    }
 ],
 "README_instructions": "## Installation Clone this repository and its
     submodules.[....]"
}
```

Listing 1.1. JSON snippet showing Step 1 including two actions (Steps) *“clone”* and *“install”*

We manually examined cases where annotators disagreed. For example, significant confusion arose from overly complicated instructions detected in README files, particularly in cases where installation instructions were included in the markdown subheadings, such as `#Step1: Download the files`, followed by a paragraph like `Step1: Download the files with the following commands`. We resolved these conflicts by removing the content of these subheadings and providing detailed annotations for the subsequent paragraph.

Next, we faced challenges when describing plan types and steps across supported technologies. For instance, while instructions for the package manager plan typically involve running pip install, the `TorchCP` library offered alternative installation methods like the `TestPyPI` server. To resolve this, we created a distinct plan named "package manager" and specified `TestPyPI` as the associated technology property.

Lastly, conflicts emerged concerning the inclusion of installation requirements. Some cases listed requirements within the installation instructions, while others deposited them in a separate section, traditionally before the installation instruction content begins. We decided to include these software requirements specifications only when they were part of the installation instruction section content.

3.5 Distribution of the Installation Instructions of README Files

Table 2 shows descriptive statistics of the selected research software projects based on our annotations. We reported four distinct installation Plans: binary, source, package manager and container. Notably, over half of our corpus exclusively relied on the "source" method for installing research software via README files. While "from source" was the most prevalent standalone method (66%), container and package manager plans were observed in only two and one cases, respectively. As anticipated, the "binary" method was not reported at all, indicating its rarity on open source general repositories such as Github. Unsurprisingly, the most popular research software tools e.g., `tensorflow` or `langchain` incorporated the instructions to install with package manager, typically consisting of up to two steps. The Plans vary widely in their number of steps. For example, *"simple"* Plans e.g., Package Manager and Container consists of 2–3 steps, while *"complex"* featured 10 (see Table 2 col (Total Steps)). This diversity in the number of steps impacts installation Plan length in two ways: 1) more steps introduce more complexity, and 2) additional instructions can serve as obstacles, needing further action for installation.

Approximately 44% of our samples offered multiple plans or combinations for software installation, suggesting a diverse landscape of installation approaches. Further analysis of these combinations revealed redundant information across many instruction sections, highlighting potential challenges for LLMs in accurately identifying plans and steps. For instance, the maximum length of installation instructions for the source plan reached approximately 1,765 tokens, underscoring the complexity and variability of these instructions. This diversity not only reflects the varied nature of installation plans but also poses challenges for LLMs in accurately parsing and selecting relevant instructions, potentially leading to errors in plan and step detection. The total average length of installation instructions across all subjects was 130.79 tokens.[3]

3.6 *PlanStep* Prompting

This section introduces our PlanStep prompt templates and its explanations for such tasks, as depicted in PROMPT101 and PROMPT201.

[3] Additional details about the corpus and its data exploration are available in our GitHub repository: https://github.com/carlosug/READMEtoP-PLAN/blob/main/RESULTS/corpus-explore-data.ipynb.

Table 2. Statistics of plan and steps in the corpus. We report the number and average of "ids" per plan type, multiple plans, maximum total steps in a plan, and the length of installation instructions with parameters (TokenInstall.)

Plan Types(*Combinations*)	Counts(Prop. %)	Max. Total Steps	Max.Tokens$_{\text{Install_md}}$
Source	22 (66.66%)	10	253
Container	2 (6.06%)	3	158
Package Manager	1 (3.03%)	2	102
Binary	– (–%)	–	–
Mult. Binary	1 (3.03%)	1	108
Mult. Source	1 (3.03%)	3	303
Mult. Package Manager	3 (9.09%)	3	348
Package Manager & Source	1 (3.03%)	3	217
Package Manager & Binary	1 (3.03%)	2	187
Container & Source	1 (3.03%)	9	725
ALL	33 (100%)		

We directly instruct the LLMs with prompt design to describe the installation methods (Plan) and their corresponding installation instructions (Step) for each README. That is, the usual zero-shot prompt is set to ask LLM two tasks, Plan and Step, respectively. Since the LLM contains no information about these terms, we describe the terms and their respective meanings next to the task of the prompt. Consequently, the prompts used in our experiment can be categorised as follows:

Plan Prompting: This task is about extracting the installation method as Plans described in a README. We named it the PROMPT101, and it contains the four unique Plans and its definitions.

Plan Task (PROMPT101)

Plan Task (PROMPT101):
Given the following README, your task is to identify and list the unique installation methods. These methods are plans containing instructions for installing research software, to be executed in a specific order and under defined conditions. Exclude code commands. Be concise.

1. Binary: Install via download and run precompiled files. For example, GitHub releases.
2. Container: Install the software and its dependencies via isolated environments. For example, Docker, Podman, or Singularity.
3. Package Manager: Install via tools and indexed repositories. For example, Conda, Homebrew, or Pip.
4. Source: Run via command-line, manage and install dependencies, compile source code to a target machine, build, and run. For example, download raw source code, clone repositories, and install dependencies from code repositories.

Step Prompting: This task asks for detecting the installation instructions as Steps found in a README. We named it the PROMPT201 and it requests a list of Steps for a given installation plan.

Step Task (PROMPT201)

Step Task (PROMPT201):
Given the following README, extract the installation instructions for each installation method. These methods are plans containing instructions as steps for installing research software, to be executed in a sequential order, and under defined conditions. Exclude code commands. Be concise.

1. Binary:[....] 4. Source[a].

[a] We insert the same definitions as states in PROMPT101

4 Experiments

In order to evaluate the effectiveness of our approach, we conducted experiments to test the ability of LLMs to capture plans and the sequence of tasks required to install different software.

4.1 Experimental Setup

We employed `Mixtral-8x7b-Instruct-v0.1` [12] and `LLaMA2-70b-chat` [31], which are two of the most widely-used open-source LLMs with public access.[4] Both models demonstrate moderately good instruction-following capabilities [43]. Throughout our experiments, we maintained a `temperature` of 0 (argmax sampling) to ensure reproducible results. The ground-truth annotations and study subjects used to compare LLM's predicted responses in the experiments were those presented in Sect. 3.3 and Sect. 3.4.

4.2 Evaluation Metrics

To assess our proposed *PlanStep* method, we employed the following metrics to assess the performance of LLMs on NLP-oriented tasks (as proposed by [3]):

- **F1-scores**: these scores are computed to compare the performance of LLMs in extracting plans with the ground truth annotations.
- **Recall-Oriented Understudy for Gisting Evaluation (ROUGE)** [16]: we report ROUGE-1 (R1), ROUGE-2 (R2), and ROUGE-L to evaluate the quality of the results by comparing the LLMs extracted steps with the ground-truth dataset.

[4] Accessible through a public Python API Library: https://pypi.org/project/groq/.

Table 3. Results obtained on the Plan detection task.

LLM	Zero-shot		
	Precision	Recall	F1 score
`llama-2-7b-chat`	0.4615	0.8333	**0.5941**
`Mixtral-8x7b-Instruct-v0.1`	0.4068	0.6667	0.5053

Table 4. Evaluation results for detecting task steps for each plan. The scores (%) for Rouge-1 (R1), Rouge-2 (R2), and Rouge-L (RL) for the generated step descriptions compare our results against the ground truth steps.

LLM	Zero-shot		
	R1 ↑	R2 ↑	RL
`llama-2-7b-chat`	29.48	18.88	27.75
`Mixtral-8x7b-Instruct-v0.1`	46.42	37.53	45.27

4.3 Evaluation Results

The results of our evaluation are shown in Table 3 and Table 4 for Plan and Step tasks respectively. We present the performance of open-source LLM on two task with the standard (PROMPT101 and PROMPT201) zero-shot prompt templates.

Plan-Task: To evaluate the effectiveness of the LLMs, we tested the models in different ways, measuring the change in performance on plan task by comparing their generated response plans with ground truth annotations. We used zero-shot approach. Table 3 summarises our results on plan tasks, and compares both LLMs' performance.

Both LLMs in zero-shot prompting achieved roughly a performance of more than 50% F1-score. LLaMA-2 exhibits superior performance over MIXTRAL in plan task with the LLaMA-2 outperforming the best of our experiment by 9% compared to MIXTRAL.

Step-Task: We evaluated the performance with three metrics to measure the quality in analyzing step-task. Table 4 shows the performance of the models (e.g., MIXTRAL and LlaMA).

We further observe that while MIXTRAL consistently outperforms LLaMA-2 across all ROUGE scores (R1, R2, and RL) in the *Step-task*, achieving approximately 15% higher scores, both models demonstrate similarly poor performance in adhering to optimal step orderings, with scores ranging from 0.46 to 0.29. These findings suggest that both models struggle with the task of sequentially ordering steps in a installation Plan.

4.4 Analysis

Results of Plan-Step task. Experimental results indicate that both LLMs scored an average of around 55% F1-score for plan-task, and 37% ROUGE scores for step-task. This suggest that LLMs intrinsically vary in their abilities to solve complex tasks and reason efficiently, which are crucial for extracting plans and detecting steps more accurately.

Error Analysis. We performed a detailed analysis on specific cases where the detection performance of the LLMs differ significantly from the annotations to understand why certain steps were falsely detected. We manually studied all errors made by LLMs and classify them into four categories. Table 5[5] shows the count of each error type on Plan-Step tasks: **E1:** means models call Plans and Steps installation instructions wrongly by reusing prompt input `e.g., "Binary": ["Step 1: Definition Prompt."`; **E2:** indicates cases where models include notes and code commands in their responses, resulting in falsely imputed new steps to a wrong Plan; **E3** refers to situations where models extract steps correctly but assign them to the wrong Plan Types due to a mixture of verbs or words associated with the different methods, and lack of context e.g., if the word `"pip"` appears, the LLM directly assigns the corresponding step to directly "Package Manager"[6]; **O** represents errors in an unclassified category (e.g., summarizing steps, incorporating steps from a previous README, splits steps or invented sentences as hallucinations). Further tables, plots and error responses examples can be found in the Appendix.

Our results suggest that different Plans exhibit a wide diversity in error types: simple installation tasks with few actions ("Package" and "Binary") primarily encounter issues related to **E3**; notably, "Source" faces more issues with **E1**, indicating a significant impact of prompts on model performance. Across Plan types, we observe nearly identical results suggesting a possible explanation: concise instructions in README files may significantly reduce these incorrect behaviors, leading to successful execution of installation steps. Additional experiments are needed to assess this hypothesis.

Effect of Prompts. Figure 3 shows an overview of the steps detected by each LLM. Both MIXTRAL and LLaMA-2 perform on par *(score: 10 vs. 7)* in detecting steps correctly and incorrectly (score: 8 vs. 9). However, the latter exhibits

[5] The raw material we used to calculate the counts are listed in our repository [44]: `qualitative_error_analysis.md`.

[6] Install python packages from a git repository has been classified as step of "Source" plan.

Table 5. Counts of *PlanStep* on different Plans. E1: wrong Plans category but correct Steps; E2: wrong order of steps but correct number; E3: wrong sequential order; O: others

	LLAMA				MISTRAL			
Error Type:	E1	E2	E3	O	E1	E2	E3	O
Binary	0	0	1	0	0	0	0	1
Source	15	4	8	1	0	6	4	2
Package Manager	0	0	1	0	1	0	4	0
Container	0	1	1	1	0	0	1	0

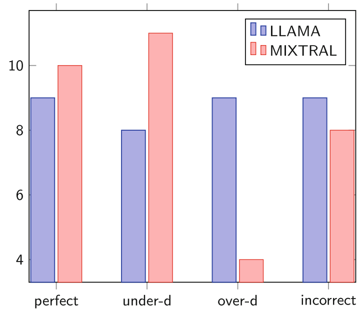

Fig. 3. Total count of steps detected for each Plan per LLM, in comparison with the ground truth. If a LLM detected fewer steps than the annotations, we consider it under-detection **(under-d)**, while if it detected more, it indicate over-detection **(over-d)**. A correct step detection (**(perfect)** indicates the number of steps agree with those in the ground truth. The **(incorrect)** detection counts steps in a plan that are falsely detected i.e., the LLM model detected a plan with steps that are not part of our annotations)

slightly worse performance compared to the former in over-detection cases *(score: 4 vs. 9)*, which is likely due to the ability of the model to insert prompts information into the responses. The definitions seem to inadvertently lead LLMs to incorrectly detect plans and steps by copying extra steps added solely to the prompt definition i.e., **E1:** call non-existing Plans by adding prompt's input. This observation suggests the need for additional testing with zero-shot prompts for different installation plans (and reduce the definitions used in the prompt). More advanced zero-shot prompting methods [40] as well as chain-of-thought prompt strategies [41] to effectively guide LLMs in translating steps into smaller sub-tasks will be investigated in our future work.

Few-shot prompt strategies such as LLM4PDDL [28] together with chain-of-thought prompts, may provide an expressive and extensible vocabulary representation for semantically writing and describing plans to machines. We plan to investigate this approach further in future work.

5 Discussion

This work aims to automatically extract all available installation information from research software documentation. Our experiments demonstrate that while LLMs show promising results, there is substantial room for improvement. During our analysis, we prioritized extracting concise plans and steps of software installation text using two LLMs. LlaMA-2 generally demonstrates the fewest errors in plan-task, indicating a higher accuracy in predicting the installation methods. The LlaMA-2, however, shows a progressively higher number of errors when dealing with steps. MIXTRAL exhibits the opposite. We observe that MIXTRAL outputs are significantly more truthful than LlaMA-2 with less randomness and creativity in their responses. Notably, the more steps involved, the more frequent errors across both models, indicating the challenges faced in accurately predicting parameters for tools.

Moreover, the reliance on LLM for the evaluation of plans and step instructions introduce new challenges. As LLM's ability in planning tasks in under scientific scrutiny [14,29], there is a crucial need for further validation and fine-tuning of its capabilities in this specific context.

We are in our initial phase of the experimental research project, and consequently, components from *PlanStep* approach will certainly be updated and revised. First, we believe that designing combinations of few-shot prompt standards with the addition of strict formal language will improve the ability of LLMs to detect plans, and their instructions for installation consistently. Second, additional evaluations are needed to validate the insights obtained in our experiments. For plan tasks, our approach may be compared with baseline models, measuring the change in performance. Third, increasing the size of our annotated corpus is notably advantageous, providing a broader exploration of alternative semantic approaches formal representations. However, the manual nature of our instruction writing process limits our capacity to scale this work significantly.

6 Conclusion and Future Work

In this work we presented an evaluation framework and initial experimentation for using LLMs as a means to extract alternate research software installation plans and their corresponding instructions. Our approach involves equipping the LLM with essential documentation tailored to installation instructions, enabling them to refine their accuracy when using the README and improve their performance in automating the detection of installation instructions. As part of our evaluation framework we have proposed an annotated corpus, which collects different research software with their installation instructions, to systematically evaluate LLMs in extracting tasks, including plans and steps belonging to those plans.

Our experiments show promising results for both plan detection and step detection, although we are still a long way from our goal. We are currently extending our approach in different directions. First, we are augmenting the annotation corpus to consider additional README files of increasing complexity in order to create a comprehensive benchmark, distinguishing READMES of different complexity. Second, we aim to improve the prompting strategies used in our approach, including few-shot examples to better equip the model with the goal of each PlanStep task. Our central goal is to create an assistant that aids in installing research software while addressing issues that may currently exist in the installation instructions. Investigating further the addition of executable instructions in formalised and machine-readable language from classical planning research community *i.e., Domain Definition Language (PDDL)* [1] and beyond i.e., *P-Plan Ontology* [7] is another research goals of ours.

Acknowledgements. This work is supported by the Madrid Government (Comunidad de Madrid - Spain) under the Multiannual Agreement with Universidad Politécnica de Madrid in the line Support for R&D projects for Beatriz Galindo researchers, in the context of the VPRICIT, and through the call Research Grants for Young Investigators from Universidad Politécnica de Madrid. The authors would also like to acknowledge European Union's Horizon Europe Programme under GA 101129744 – EVERSE – HORIZON-INFRA-2023-EOSC-01-02.

Appendix

Examples of Error Types Produced by LLMs

```
Question: PROMPT201
Correct Answer:
    },
    "7": {
        "Source": [
            "### Dependencies \n Initialize git submodules with",
            "### Python environment \n Install the specific versions of every
     package from ‘requirements.txt‘ in a new conda environment:",
            "To ensure that Python paths are properly defined, update the
    ‘~/.bashrc‘ by adding the following lines"
        ]
    },
LLAMA Answer:
    "7": {
        "Binary": [
            "Step 1: Download the precompiled files from GitHub releases."
        ],
        "Container": [
            "Step 1: Initialize git submodules with ‘git submodule init‘ and
    ‘git submodule update‘.",
            "Step 2: Create a new Conda environment with the specific
    versions of every package from ‘requirements.txt‘ using ‘conda create --
    name gsft python=3.9‘.",
            "Step 3: Activate the environment using ‘conda activate gsft‘.",
            "Step 4: Install the packages listed in ‘requirements.txt‘ using
    ‘pip install -r requirements.txt‘."
        ],
        "Source": [
            "Step 1: Clone the repository.",
            "Step 2: Initialize git submodules with ‘git submodule init‘ and
    ‘git submodule update‘.",
            "Step 3: Create a new Conda environment with the specific
    versions of every package from ‘requirements.txt‘ using ‘conda create --
    name gsft python=3.9‘.",
            "Step 4: Activate the environment using ‘conda activate gsft‘.",
            "Step 5: Install the packages listed in ‘requirements.txt‘ using
    ‘pip install -r requirements.txt‘."
        ]
    },
```

Listing 1.2. Demonstration id: 7 (Error Type. 1 - Adding definition in the LLM answer)

```
Question: PROMPT201
Correct Answer:
    "14": {
        "Source": [
            "Install the requirements with:",
            "or set up TPU VM with:"
        ]
    },
MISTRAL Answer:
    "14": {
        "Package Manager": [
            "1. Create a new conda environment with Python 3.10.",
            "2. Activate the new conda environment.",
            "3. Install Jax with CUDA 12 support: ‘pip install -U ‘jax[
    cuda12_pip]==0.4.23‘ -f https://storage.googleapis.com/jax-releases/
    jax_cuda_releases.html‘",
            "4. Install the required packages: ‘pip install -r requirements.
    txt‘"
        ]
    },
```

Listing 1.3. Demonstration id: 14 - Error Type. 2 (Mixture of steps from code commands and notes)

```
Question: PROMPT201
Correct Answer:
   },
    "13": {
        "Container": [
            "First, install Docker on your machine. Then move to the original
     directory ‘learning_to_fly‘ and build the Docker image:",
            "If desired you can also build the container for building the
    firmware:",
            "After that you can run it using e.g.:",
            "Navigate to [https://0.0.0.0:8000](https://0.0.0.0:8000) with
    your browser, and you should see something like in the screenshot above (
    after starting the training).",
            "Navigate to [https://0.0.0.0:6006](https://0.0.0.0:6006) with
    your browser to investigate the Tensorboard logs."
        ],
        "Source": [
            "Clone this repository:",
            "Then instantiate the ‘RLtools‘ submodule:",
            "Then instantiate some dependencies of ‘RLtools‘ (for
    conveniences like checkpointing, Tensorboard logging, testing, etc.):",
            "#### Install dependencies on Ubuntu",
            "#### Install dependencies on macOS",
            "Going back to the main directory (‘learning_to_fly‘), we can now
     configure the build of the code:",
            "Finally, we can build the targets:",
            "After successfully building the targets, we can run the code (in
     the original directory ‘learning_to_fly‘):",
            "If not already installed:",
            "Then from the original directory ‘learning_to_fly‘:",
            "To run the training with the UI, we download the JavaScript
    dependencies in the form of the two files ‘three.module.js‘ and ‘
    OrbitControls.js‘:",
            "After that we can execute the UI binary from the root folder:",
            "Now you should be able to navigate to [http://0.0.0.0:8000](http
    ://0.0.0.0:8000) in your browser and start the training."
        ]
    },
LLAMA Answer:
    "13": {
        "Container": [
            "Step 1: Install Docker on your machine.",
            "Step 2: Move to the original directory ‘learning_to_fly‘ and
    build the Docker image: ‘docker build -t arpllab/learning_to_fly .‘",
            "Step 3: Run the container using ‘docker run -it --rm -p
    8000:8000 arpllab/learning_to_fly‘.",
            "Step 4: Navigate to [https://0.0.0.0:8000](https://0.0.0.0:8000)
     with your browser to access the training program."
        ],
        "Source": [
            "Step 1: Clone the repository: ‘git clone https://github.com/
    arplaboratory/learning-to-fly‘.",
            "Step 2: Instantiate the ‘RLtools‘ submodule: ‘git submodule
    update --init -- external/rl_tools‘.",
            "Step 3: Instantiate some dependencies of ‘RLtools‘: ‘git
    submodule update --init -- external/cli1 external/highfive external/json/
     external/tensorboard tests/lib/googletest/‘.",
            "Step 4: Install dependencies on Ubuntu or macOS.",
```

```
        "Step 5: Configure the build: ‘cmake .. -DCMAKE_BUILD_TYPE=
    Release -DRL_TOOLS_BACKEND_ENABLE_OPENBLAS:BOOL=ON‘ (for Ubuntu +
    OpenBLAS) or ‘cmake .. -DCMAKE_BUILD_TYPE=Release -
    DRL_TOOLS_BACKEND_ENABLE_MKL:BOOL=ON‘ (for Ubuntu + MKL) or ‘cmake .. -
    DCMAKE_BUILD_TYPE=Release‘ (for macOS).",
        "Step 6: Build the targets: ‘cmake --build . -j8‘.",
        "Step 7: Run the code: ‘cd .. ./build/src/training_headless‘.",
        "Step 8: To run the training with the UI, download the JavaScript
     dependencies and execute the UI binary: ‘cd src/ui ./get_dependencies.sh
    ‘ and ‘cd ../../ ./build/src/ui 0.0.0.0 8000‘.",
        "Step 9: Navigate to [http://0.0.0.0:8000](http://0.0.0.0:8000)
    in your browser to access the training program."
      ]
    },
```

Listing 1.4. Demonstration id: 13 - Error Type. O (correct sequential order but llm split and invented sentences)

Detailed Tables

- Figure 4 illustrates the distribution of total length of readme instructions for each study subject in our corpus.
- Figure 5 aggregates the study subjects per distinct Plan type and its technology properties.
- Plots where each bar represents an ID research software, and within each bar, different colored segments represent the ratio of system-detected steps to reference steps for each method. Ratio of LLM Detected steps to Annotations steps. A value around 1 indicates a good match between LLM and Annotations (Figs. 6 and 7).

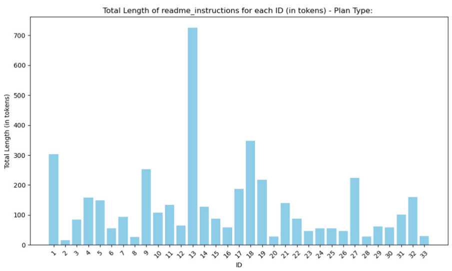

Fig. 4. Length (Tokens) of the README for each "id" research software

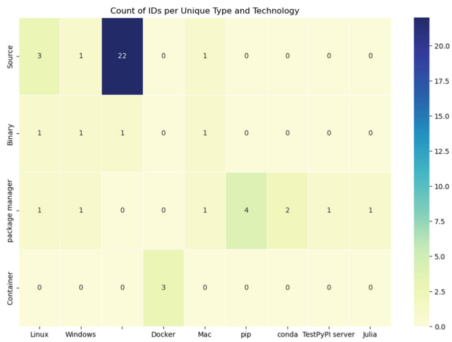

Fig. 5. Heatmap of our corpus

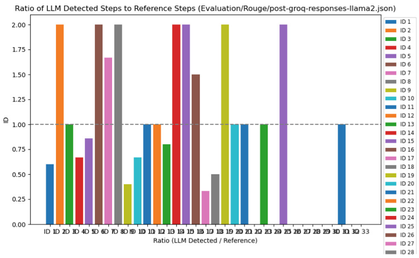

Fig. 6. Ratio of LLM detected steps by LlaMa versus ground truth steps

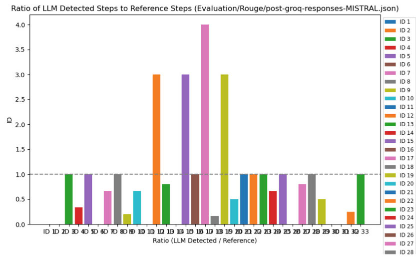

Fig. 7. Ratio of LLM detected steps by MIXTRAL versus ground truth steps

References

1. Constructions Aeronautiques et al.: PDDL—the planning domain definition language. Technical report (1998)
2. Microsoft Research AI4Science and Microsoft Azure Quantum. "The impact of large language models on scientific discovery: a preliminary study using GPT-4". arXiv:2311.07361 (2023)
3. Blagec, K., et al.: A global analysis of metrics used for measuring performance in natural language processing. arXiv:2204.11574 (2022)
4. Boiko, D.A., et al.: Autonomous chemical research with large language models. Nature **624**(7992), 570–578 (2023). https://doi.org/10.1038/s41586-023-06792-0. https://www.nature.com/articles/s41586-023-06792-0. ISSN 1476-4687. Accessed 31 Dec 2023
5. Hong, N.P.C., et al.: FAIR Principles for Research Software (FAIR4RS Principles). Version 1.0 (2022). https://doi.org/10.15497/RDA00068
6. Du, C., et al.: Softcite dataset: a dataset of software mentions in biomedical and economic research publications. J. Assoc. Inf. Sci. Technol. **72**(7), 870–884 (2021). https://doi.org/10.1002/asi.24454. ISSN 2330-1635
7. Garijo, D., Gil, Y.: Augmenting PROV with plans in PPLAN: scientific processes as linked data. In: Second International Workshop on Linked Science: Tackling Big Data (LISC), Held in Conjunction with the International Semantic Web Conference (ISWC), Boston, MA (2012)
8. Hirsch, E., Uziel, G., Anaby-Tavor, A.: What's the plan? Evaluating and developing planning-aware techniques for LLMs. arXiv:2402.11489 (2024)
9. Hirsch, E., Uziel, G., Anaby-Tavor, A.: What's the plan? Evaluating and developing planning-aware techniques for LLMs (2024). arXiv:2402.11489 [cs]. Accessed 14 Mar 2024
10. Hou, X., et al.: Large language models for software engineering: a systematic literature review (2023). http://arxiv.org/abs/2308.10620. Accessed 05 Sept 2023
11. Huang, X., et al.: Understanding the planning of LLM agents: a survey. arXiv:2402.02716 (2024)
12. Jiang, A.Q., et al.: Mixtral of experts. arXiv:2401.04088 (2024)
13. Jin, Q., et al.: GeneGPT: augmenting large language models with domain tools for improved access to biomedical information (2023). arXiv:2304.09667 [cs, q- bio]. Accessed 14 Mar 2024

14. Kambhampati, S., et al.: LLMs can't plan, but can help planning in LLM-modulo frameworks. arXiv:2402.01817 (2024)
15. Kelley, A., Garijo, D.: A framework for creating knowledge graphs of scientific software metadata. Quant. Sci. Stud. 1–37 (2021). https://doi.org/10.1162/qss_a_00167. ISSN 2641-3337
16. Lin, C.-Y.: ROUGE: a package for automatic evaluation of summaries. In: Text Summarization Branches Out, Barcelona, Spain, pp. 74–81. Association for Computational Linguistics (2004). https://www.aclweb.org/anthology/W04-1013
17. Mao, A., Garijo, D., Fakhraei, S.: SoMEF: a framework for capturing scientific software metadata from its documentation. In: 2019 IEEE International Conference on Big Data (Big Data), pp. 3032–3037 (2019). https://doi.org/10.1109/BigData47090.2019.9006447
18. Miglani, S., Yorke-Smith, N.: NLtoPDDL: one-shot learning of PDDL models from natural language process manuals. In: ICAPS 2020 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS 2020) (2020)
19. Mondorf, P., Plank, B.: Beyond accuracy: evaluating the reasoning behavior of large language models–a survey. arXiv:2404.01869 (2024)
20. Olmo, A., Sreedharan, S., Kambhampati, S.: GPT3- to-plan: extracting plans from text using GPT-3 (2021). arXiv:2106.07131 [cs]. Accessed 17 Jan 2024
21. OpenAI. GPT-4 Technical Report (2023). arXiv:2303.08774 [cs]. Accessed 24 Sept 2023
22. Qin, Y., et al.: InFoBench: evaluating instruction following ability in large language models (2024). arXiv:2401.03601 [cs]. Accessed 16 Feb 2024
23. Qin, Y., et al.: ToolLLM: facilitating large language models to master 16000+ real-world APIs (2023). arXiv:2307.16789 [cs]. Accessed 16 Feb 2024
24. Rula, A., D'Souza, J.: Procedural text mining with large language models. In: Proceedings of the 12th Knowledge Capture Conference 2023, K-CAP 2023, pp. 9–16. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3587259.3627572
25. Schick, T., et al.: ToolFormer: language models can teach themselves to use tools (2023). arXiv:2302.04761 [cs]. Accessed 21 Sept 2023
26. Shen, Y., et al.: TaskBench: benchmarking large language models for task automation (2023). arXiv:2311.18760 [cs]. Accessed 14 Mar 2024
27. Shridhar, M., et al.: ALFRED: a benchmark for interpreting grounded instructions for everyday tasks (2020). arXiv:1912.01734 [cs]. Accessed 16 Jan 2024
28. Silver, T., et al.: PDDL planning with pretrained large language models. In: NeurIPS 2022 Foundation Models for Decision Making Workshop (2022)
29. Stechly, K., Valmeekam, K., Kambhampati, S.: On the self-verification limitations of large language models on reasoning and planning tasks. arXiv:2402.08115 (2024)
30. Tenorth, M., Nyga, D., Beetz, M.: Understanding and executing instructions for everyday manipulation tasks from the World Wide Web. In: 2010 IEEE International Conference on Robotics and Automation (ICRA 2010), Anchorage, AK, pp. 1486–1491. IEEE (2010). https://doi.org/10.1109/ROBOT.2010.5509955. ISBN 978-1-4244-5038-1. Accessed 02 Feb 2024
31. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models. arXiv:2307.09288 (2023)
32. Tsay, J., et al.: AIMMX: artificial intelligence model metadata extractor. In: Proceedings of the 17th International Conference on Mining Software Repositories, MSR 2020, pp. 81–92. Association for Computing Machinery, New York (2020). https://doi.org/10.1145/3379597.3387448. ISBN 978-1-4503-7517-7. Accessed 20 Sept 2023

33. Valmeekam, K., et al.: Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change)
34. Valmeekam, K., et al.: On the planning abilities of large language models-a critical investigation. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
35. Valmeekam, K., et al.: PlanBench: an extensible benchmark for evaluating large language models on planning and reasoning about change (2023). arXiv:2206.10498 [cs]. Accessed 18 Jan 2024
36. Valmeekam, K., et al.: PlanBench: an extensible benchmark for evaluating large language models on planning and reasoning about change. In: Advances in Neural Information Processing Systems, vol. 36 (2024)
37. Vaswani, A., et al.: Attention is all you need. In: Guyon, I., et al. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
38. Wang, H., et al.: Scientific discovery in the age of artificial intelligence. Nature **620**(7972), 47–60 (2023). https://doi.org/10.1038/s41586-023-06221-2. ISSN 1476-4687. Accessed 07 Sept 2023
39. Wang, J., et al.: Software testing with large language models: survey, landscape, and vision. IEEE Trans. Softw. Eng. (2024)
40. Wang, L., et al.: Plan-and-solve prompting: improving zero-shot chain-of- thought reasoning by large language models (2023). arXiv:2305.04091 [cs]. Accessed 14 Mar 2024
41. Wei, J., et al.: Chain-of-thought prompting elicits reasoning in large language models. In: Advances in Neural Information Processing Systems, vol. 35, pp. 24824–24837 (2022)
42. Xie, D., et al.: Impact of large language models on generating software specifications (2023). arXiv:2306.03324 [cs]. Accessed 11 Sept 2023
43. Yuan, S., et al.: EASYTOOL: enhancing LLM-based agents with concise tool instruction (2024). arXiv:2401.06201 [cs]. Accessed 06 Feb 2024
44. Carlos, Z.: Carlosug/READMEtoP-PLAN: READMEtoP-PLAN First Release (2024). https://doi.org/10.5281/zenodo.10991890

A Technical/Scientific Document Management Platform

Melissa Lemos[1(✉)], Grettel M. García[1], Yenier T. Izquierdo[1], Cleber Oliveira[1], Jefferson Alves de Sousa[1], Bernardo Florindo Mortari Rezende[1], Bruno Novelli[1], and Marco A. Casanova[1,2]

[1] Instituto Tecgraf, PUC-Rio, Rio de Janeiro, RJ 22451-900, Brazil
{melissa,ggarcia,ytorres,cleberoli,alvesjefferson, bernardofmr,bnovelli}@tecgraf.puc-rio.br

[2] Departamento de Informática, PUC-Rio, Rio de Janeiro, RJ 22451-900, Brazil
casanova@inf.puc-rio.br

Abstract. This paper first presents DANKE, a data and knowledge management platform that allows users to submit keyword queries to a centralized database. DANKE uses a knowledge graph to provide a semantic view of the centralized database in a vocabulary familiar to the users. The paper then describes DANKE-U, a specialized module that enables DANKE to handle unstructured data, including scientific and engineering documents. Lastly, the paper presents a real use case from the oil and gas industry, involving technical/scientific documents.

Keywords: Database Keyword Search · Knowledge Graphs · Data Integration · Engineering Data · Scientific Data · Unstructured Data

1 Introduction

This paper first presents DANKE, a data and knowledge management platform that allows users, without technical training, to submit keyword queries to a centralized database. DANKE's primary motivation is to democratize access to data and documents, originally dispersed across different data sources, without requiring users to write scripts or depend on the development of applications that provide forms for querying this data.

DANKE helps construct a centralized database through a data integration process with the following major steps: (i) data extraction from the original data sources; (ii) transformation and enrichment of these data; (iii) loading the data into the centralized database; (iv) indexing the data. DANKE uses a knowledge graph to provide a semantic view of the centralized database in a vocabulary familiar to the users. The knowledge graph is the basis of the search engine, which matches the keywords users submit with the concepts of the graph, and uses the matchings to compile automatically an SQL (or RDF) query on the centralized database.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 134–146, 2024.
https://doi.org/10.1007/978-3-031-65794-8_9

The paper then describes DANKE-U, which enables DANKE to handle unstructured data, including scientific and engineering documents. DANKE-U is a collection of components that process unstructured data, extracting metadata, thumbnails, texts, images, named entities, and tables. With the help of this extension, DANKE's users can seamlessly search unstructured documents and relate them to structured data.

Lastly, the paper presents a real use case from the oil and gas industry, where DANKE is applied to technical/scientific documents and structured data about subsea production systems, that play a critical role in offshore oil and gas extraction. In this context, technical/scientific documents are challenging to interpret: they may contain data organized in non-conventional tables that merge cells (between rows and columns); or they may take the form of engineering drawings, etc. Thus, off-the-shelf tools are of limited help to analyze such documents.

The paper is organized as follows. Section 2 contains a very brief discussion on related work. Section 3 introduces DANKE. Section 4 outlines DANKE-U. Section 5 covers a use case. Finally, Sect. 6 contains the conclusions.

2 Related Work

This section briefly covers related work in areas directly connected to data integration and database keyword search.

Data Integration. Classic data integration is usually divided into the major sub-problems of data retrieval, data fusion, schema alignment, and entity linkage [3]. The progress the data integration community has made in addressing these challenges in the context of big data integration was explored in [5]. The ties between machine learning and data integration were discussed in [4,20].

In the context of engineering data, Nguyen et al. [17] described the development of a framework for data integration to optimize the remote operations of offshore wind farms. Espinola et al. [6] presented an approach that integrates data from mixed/augmented reality tools and embedded intelligent maintenance systems to support operators/technicians during maintenance tasks, providing easier access, understanding, and comprehension of information from different systems. Urbina Coronado et al. [22] described how to integrate data from machine tools with production data collected by a Manufacturing Execution System (MES) to monitor process output, consumable usage, and operator productivity.

Database Keyword Search Engines. Early relational keyword-based query processing tools [1,18,19] explored the foreign/primary keys declared in the relational schema to compile a keyword-based query into an SQL query with a minimal set of join clauses based on the notion of candidate networks.

State-of-the-art content-based image retrieval strategies [9,13] assume that high-dimensional vectors represent images, created using Deep Learning techniques. Alternatively, the tool might transform both the text and the images (or, in fact, any other media as well) into a single high-dimensional vector, as in cross-modal retrieval techniques [2,25].

Tautkute et al. [21] proposed a multimodal search engine in the fashion domain that retrieves items aesthetically similar to a query composed of image and textual inputs. Vo et al. [23] addressed a similar problem, where a query is specified in the form of an image and a text description of desired modifications to the input image. Yu et al. [24] introduced a multimodal model for understanding commerce topics, where the content can be an image, a text, or a combination of both.

DANKE [12] is an evolution of earlier tools [8,10,11]. DANKE-U enables DANKE to handle unstructured data, including scientific and engineering documents, as mentioned in the introduction.

3 A Brief Overview of DANKE

A *keyword query* in DANKE is just a list of terms, or *keywords*, that the user wants to search the database for, and may include reserved terms, such as "<", ">", "*between*". In relational jargon, an *answer* to a keyword query is formatted as a table whose columns (or column names) contain the keyword matches. The answer may be the result of joining several database tables, that is, an answer to a keyword query does not need to be constructed out of a single table.

DANKE has three main components (see Fig. 1): *Storage Module*; *Preparation Module*; and *Knowledge Extraction Module*.

The *Storage Module* houses a relational (or RDF) database, constructed from various data sources. The database is described by a knowledge graph (KG), which is independent of the data model of the underlying database. The *Storage Module* also holds data indices required to support keyword search and other services.

The *Preparation Module* provides tools for creating the knowledge graph and for constructing and updating the centralized database through a pipeline responsible for a typical data integration process. This includes indexing data, collecting data from data sources, transforming and enriching data, and ingesting data into the database. This module enables customization of the pipeline in the most convenient manner for each case. The knowledge graph KG is defined by de-normalizing the schemas of the underlying databases, enriching KG with metadata that help interpret the data, and indicating which properties will have their values indexed.

The *Knowledge Extraction Module* uses the technology described in [7,8,10] and explores the knowledge graph and the data indices to compile a keyword query into an SQL (or SPARQL) query that returns data that best match the keywords. It features an algorithm [8] that accepts as input a keyword query Q_K, the knowledge graph KG, and the indices, and: (1) finds matches with the keywords in Q_K; (2) creates a conceptual query Q_C by exploring the keyword matches found and KG; (3) compiles Q_C into an SQL (or SPARQL) query Q_S, which is then executed.

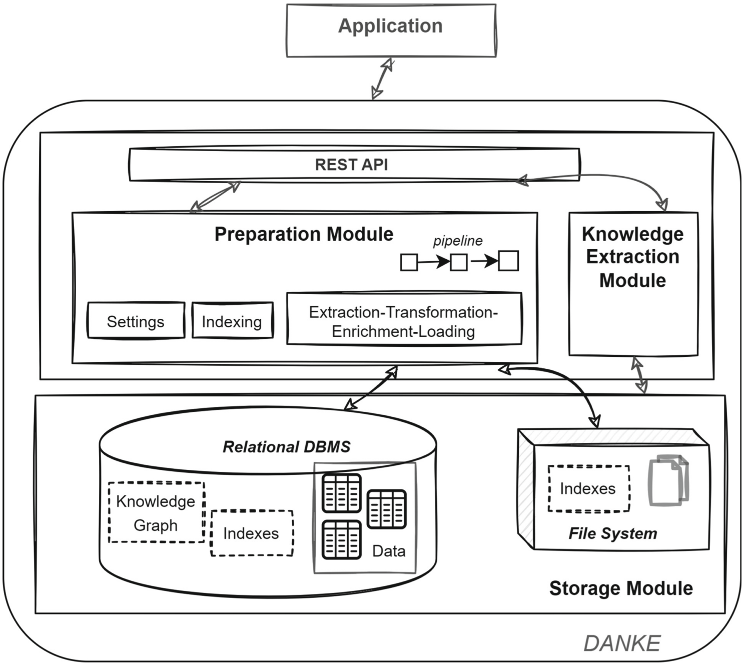

Fig. 1. Architectural Overview of the DANKE Platform

4 DANKE-U

To handle unstructured data, such as technical/scientific documents, DANKE creates pipelines using DANKE-U, as illustrated in Fig. 2. DANKE-U comprises a set of components that process unstructured data, extracting metadata, thumbnails, texts, images, named entities and tables, for example.

A pipeline encompasses several forms of text document processing, automatically extracting information and enriching the database with new data and relationships. If it is necessary to search through the textual content of the file, the pipeline must include text indexing, enabling the search engine to match keywords with the textual content of the file.

In relational mode, documents and the associated data are recorded in a table, including their identification, file path, thumbnail, textual content, and metadata such as title, author, date, and type. Furthermore, other tables that relate documents to additional information may also be used.

An example of text document processing is extracting attribute values from specific entities using advanced table processing techniques facilitated by the "Table Extraction" component of DANKE-U. This component can be tailored to extract data from various types of tables, as illustrated on the right-hand side of Fig. 2, and executes the processing in three stages: (i) Collect and Settings: involves reading a folder containing PDF documents and a configuration file defining the metadata and tables to be investigated in the documents. The configuration determines domain-specific structures (e.g., applying regular expressions to identify entity and attribute names) used to select tables and attributes for extraction, along with formatting options for each attribute in the output. (ii) Extractor: extracts the attributes and tables defined in the previous stage. The process utilizes the Camelot library[1], capable of extracting tables with column and line separators (lattices, illustrated in Figs. 4 and 5 of the appendix) as well as those without clear separators (stream, illustrated in Fig. 6 of the appendix). From the Camelot output, the component identifies and extracts the attributes based on the configuration. (iii) Cleaner and Formatter: transforms the extracted data into tabular format, which can be stored using spreadsheets, CSV, or database tables. The user can personally and configure the output format through the configuration file.

Another relevant example of text processing is Named Entity Recognition. Assuming the closed-world hypothesis, where there is a known set of values representing an entity, this technique can identify which documents mention a specific entity in their text. Hence, it is possible to relate documents to entities, storing this relationship in the database.

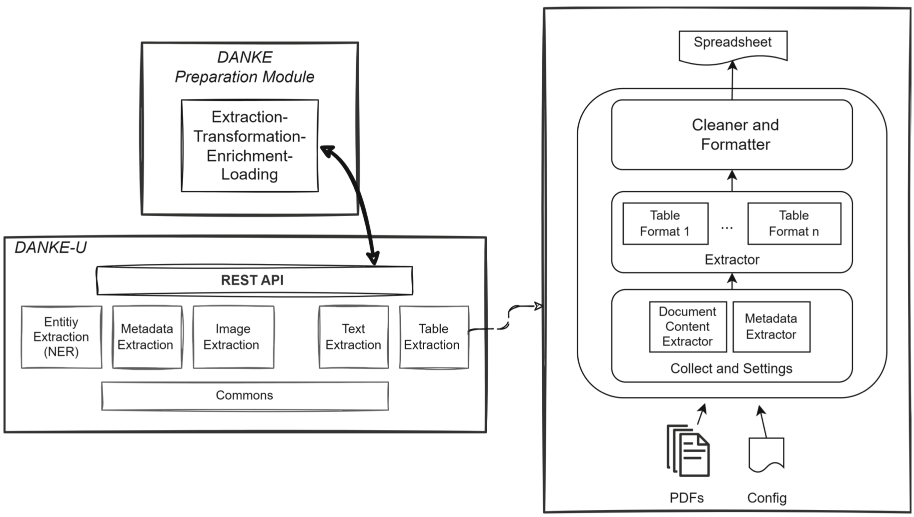

Fig. 2. Architectural Overview of the DANKE-U Module

[1] https://camelot-py.readthedocs.io/en/master/index.html.

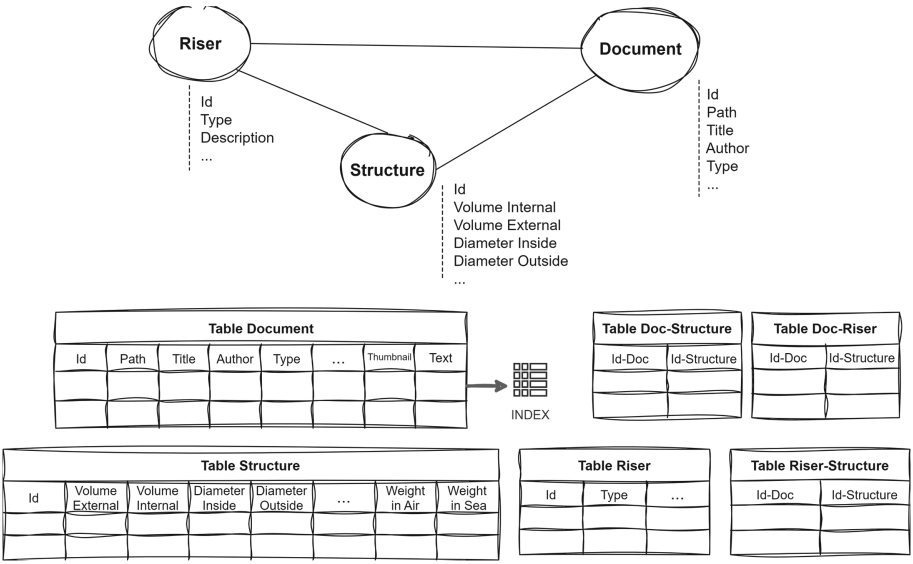

Fig. 3. Extracting data from tables

5 Use Case

This section introduces the domain in which DANKE is employed to handle scientific and technical documents, illustrating the approach with real-life cases in subsea production systems of offshore oil and gas extraction.

Subsea production systems are composed of increasingly complex structures to meet the growing demands of large deep-water oil fields. Moreover, maintaining the physical integrity of several subsea components faces an additional challenge from their remote and hard-to-access locations. Nonetheless, tracking their operational health is crucial to guarantee environmental safety, reduce the high costs of an unplanned operation shutdown, and meet regulatory demands.

To accomplish these tasks, subsea experts need efficient access to consistent and up-to-date information about equipment designs, layouts and locations, inspection reports, maintenance activities, and other daily-generated data. However, throughout the life cycle of the subsea system, a large variety of data are generated and stored in various places, creating a complex environment. In this context, accessing, searching, navigating, and integrating this data becomes critical for decision-making. However, these actions are often costly and require substantial time from experts, technicians, and engineers.

Figure 3 illustrates a fragment of the knowledge graph that represents the subsea domain, encompassing: (i) a riser, a conductor pipe connected to offshore production platforms and the subsea flowlines, manifolds, and wellhead; (ii) a structure detailing the specific attributes for which a riser is designed, such as volume, diameter, and weight; and (iii) documents containing engineering and scientific data related to the risers and structures. Figure 3 also displays the resulting relational database, featuring tables to store data about risers, structures, documents, and the relationships among them. The appendix contains three illustrations of tables with a layout commonly found in PDFs in a real-world scenario; however, for confidentiality reasons, the values of the structures are fictitious.

The following use cases illustrate the approaches used in this domain for searching the documents.

Use Case 1 - A Keyword Query Over Document Metadata. Suppose the user enters the keyword query “document type technical drawing”. DANKE compiles this keyword query into an SQL query involving the table “Document” and its attribute “type”, filtering for the attribute value “technical drawing”, and responds with documents of the type technical drawing. Document metadata may have been automatically extracted from the PDF file using the “Metadata Extraction” component of DANKE-U, or it could have been manually entered by a specialist using an Electronic Document Management (EDM) system.

Use Case 2 - A Keyword Query Over Document Text. Suppose the user enters the keyword query “document diagram yyyy.xxxx”. DANKE compiles this keyword query into an SQL query to find documents of type “diagram” that contain the text “yyyy.xxxx”. To locate the text “yyyy.xxxx” within diagrams, represented as images in PDFs, it is necessary to extract the text from the images using OCR, a task performed by the “Text Extraction” component of DANKE-U. Additionally, the pipeline created in DANKE’s Preparation Module must store and index the text extracted from the PDF.

Use Case 3 - A Keyword Query Relating Data and Documents. Suppose the user enters the keyword query “structure document diagram yyyy.xxxx”. DANKE compiles this keyword query into an SQL query similar to the previous example but includes the relationships between structures and documents. To establish this relationship, document texts must be processed to identify those that mention structure identifiers. This task is accomplished by the “Entity Extraction” component of DANKE-U. Using the structure identifiers, previously known and stored in the table “Structure”, the component “Entity Extraction” processes the document texts and returns a table associating the structure identifiers with the document identifiers.

Use Case 4 - A Keyword Query Involving Attribute Values Defined in Documents. Suppose the user enters the keyword query “riser xxxx-yyyy structure diameter.” DANKE compiles this keyword query into an SQL query that retrieves the diameter of the structures for the riser “xxxx-yyyy”. Some attributes of the structure are not originally present in a structured form (e.g., in a relational database) and had to be extracted through a sophisticated process that interprets various table formats found in documents. This extraction is performed by the “Table Extraction”component of DANKE-U, customized to process complex tables. Figures 4, 5 and 6 in appendix illustrates a real-life scenario. However, for confidentiality reasons, the values presented in the tables are fictitious. The classical format of a table places each attribute in a separate cell, as in Fig. 4. In this example, the figure shows a table with diameter *4* for the structure identified as *001.00024*. However, in some cases, when the attribute value is the same for different structures, the attribute value is presented only once by merging cells. Figure 5 presents a slightly more complex table to process, as it contains the same diameter value (*4*) in a merged cell, for three different structures: *001.00021, 001.00022, and 001.00023*. Figure 6 depicts a table with a different layout but also containing the diameter value 4 of structure *001.00024*. Furthermore, a page break may occur within the same table.

6 Conclusions and Directions for Future Work

The paper first presented the overall architecture of DANKE, a data and knowledge extraction platform, based on a search engine. Then, it described DANKE-U, that enables DANKE to handle unstructured data, including scientific and engineering documents. Finally, the paper presented a use case from the oil and gas industry, involving technical/scientific documents, processed using DANKE.

DANKE is currently being equipped with a Natural Language (NL) interface, constructed with the help of a Large Language Model (LLM), to help users locate data using NL questions, covering language constructs not supported by keyword queries [14–16]. The final goal is to explore the combination of controllable database keyword search with LLM freedom to create a *trustworthy application* that generates *reliable answers*.

As for future work, the application area involves a large variety of documents, including complete engineering drawings. This demand calls for the development of other specialized set of components to be included in DANKE-U.

Acknowledgements. This work was partly funded by FAPERJ under grant E-26/202.818/2017; by CAPES under grants 88881.310592-2018/01, 88881.134081/2016-01, and 88882.164913/2010-01; by CNPq under grant 302303/2017-0; and by Libra Consortium (Petrobras, Shell Brasil, Total Energies, CNOOC, CNPC, and PPSA) within the ANP R&D levy as a commitment to research and development investments.

A Appendix

Structure number	**001.00024**
Structure type	Flowline
Pipe Family	Family III
Application (Sweet or Sour)	Sour
Internal diameter (")	4
Internal fluid	Gas / Diesel / Dead Oil / Water
Maximum differential design pressure (bar)	621
Maximum internal design pressure (bara)	819
Minimum internal design pressure (bara)	1
Maximum internal accidental overpressure (bara)	883
Temperature associated to accidental overpressure (°C)	60
Maximum number of pressure extreme cycles	300
Maximum number of temperature extreme cycles	300
Maximum pressurization rate (MPa/min)	10
Maximum depressurization rate (MPa/min)	(2)
Maximum design temperature (°C)	90
Minimum design temperature (ºC)	-20
Maximum CO_2 content (% mol)	85
Maximum H_2S content (ppm)	120
Maximum design water depth (m)	2500
Service life (years)	30

Fig. 4. Example 1 of table format (with fictitious data)

Structure number	001.00021	001.00022	001.00023
Structure type	Riser top section	Riser bottom section	Flowline
Pipe Family	Family III		
Application (Sweet or Sour)	Sour		
Internal diameter (“)	4		
Internal fluid	Gas / Diesel / Dead Oil / Oil		
Maximum differential design pressure (bar)	345		
Minimum internal design pressure (bara)	1		
Temperature associated to accidental overpressure (°C)	40	3	3
Maximum number of pressure extreme cycles	300		
Maximum number of temperature extreme cycles	300		
Maximum pressurization rate (MPa/min)	10		
Maximum design temperature (°C)	90		
Minimum design temperature (ºC)	-20		
Maximum CO_2 content (% mol)	85		
Maximum H_2S content (ppm)	120		
Maximum chlorides content (mg/l)	161900		
Maximum design water depth (m)	2250		2500
Service life (years)	25		30

Fig. 5. Example 2 of table format (with fictitious data)

Structure datasheets

Structure 001.00021 – Riser top section

CHARACTERISTICS	IMPERIAL	METRIC
DIAMETER inside	4.00 in	101.60 mm
DIAMETER outside	7.80 in	198.00 mm
VOLUME internal	0.096 cf/ft	8.88 l/m
VOLUME external	0.331 cf/ft	30.79 l/m
WEIGHT in air empty	49.15 lbf/ft	73.14 kgf/m
WEIGHT in air full of seawater	55.26 lbf/ft	82.24 kgf/m
WEIGHT in seawater empty	27.94 lbf/ft	41.58 kgf/m
WEIGHT in seawater full of seawater	34.06 lbf/ft	50.68 kgf/m
SPECIFIC GRAVITY in sea water empty	2.32	2.32
PRESSURE Nominal bursting	9369 psi	646 bars
HYDROSTATIC collapse pressure lay 2	4554 psi	314 bars
DAMAGING PULL in straight line	803346 lbf	3573.46 kN
MINIMUM BENDING RADIUS for STORAGE	4.22 ft	1.29 m
BENDING STIFFNESS at 20°C	46080 lbf.ft2	19.04 kN.m2
RELATIVE ELONGATION at design pressure	0.039 %	0.039 %
RELATIVE ELONGATION for 100 kN	0.017959 %	0.017959 %
THERMAL EXCHANGE COEFFICIENT at 20°C	2.39 Btu/hftF	4.14 W/(m.K)

Fig. 6. Example 3 of table format (with fictitious data)

References

1. Bergamaschi, S., Guerra, F., Interlandi, M., Trillo-Lado, R., Velegrakis, Y.: Combining user and database perspective for solving keyword queries over relational databases. Inf. Syst. **55**, 1–19 (2016). https://doi.org/10.1016/j.is.2015.07.005
2. Costa Pereira, J., et al.: On the role of correlation and abstraction in cross-modal multimedia retrieval. Trans. Pattern Anal. Mach. Intell. **36**(3), 521–535 (2014). https://doi.org/10.1109/TPAMI.2013.142
3. Doan, A., Halevy, A.Y., Ives, Z.G.: Principles of Data Integration, 1st edn. Morgan Kaufmann, San Francisco (2012). https://doi.org/10.1016/C2011-0-06130-6
4. Dong, X.L., Rekatsinas, T.: Data integration and machine learning: a natural synergy. In: Proceedings of the 2018 International Conference on Management of Data. p. 1645-1650. SIGMOD 2018, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3183713.3197387
5. Dong, X.L., Srivastava, D.: Big data integration. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 1245–1248 (2013). https://doi.org/10.1109/ICDE.2013.6544914
6. Espíndola, D.B., Fumagalli, L., Garetti, M., Pereira, C.E., Botelho, S.S., Ventura Henriques, R.: A model-based approach for data integration to improve maintenance management by mixed reality. Comput. Ind. **64**(4), 376–391 (2013). https://doi.org/10.1016/j.compind.2013.01.002
7. García, G.M.: A Keyword-based Query Processing Method for Datasets with Schemas. Ph.D. thesis, Thesis presented to the Graduate Program in Informatics, PUC-Rio (2020)
8. García, G.M., Izquierdo, Y.T., Menendez, E., Dartayre, F., Casanova, M.A.: RDF keyword-based query technology meets a real-world dataset. In: Proceedings of the International Conference on Extending Database Technology, pp. 656–667. OpenProceedings.org (2017). https://doi.org/10.5441/002/edbt.2017.86
9. Hameed, I.M., Abdulhussain, S.H., Mahmmod, B.M.: Content-based image retrieval: a review of recent trends. Cogent Eng. **8**(1), 1927469 (2021). https://doi.org/10.1080/23311916.2021.1927469
10. Izquierdo, Y.T., García, G.M., Menendez, E.S., Casanova, M.A., Dartayre, F., Levy, C.H.: *QUIOW*: a keyword-based query processing tool for RDF datasets and relational databases. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) DEXA 2018. LNCS, vol. 11030, pp. 259–269. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98812-2_22
11. Izquierdo, Y.T., et al.: Keyword search over schema-less RDF datasets by SPARQL query compilation. Inf. Syst. **102**, 101814 (2021). https://doi.org/10.1016/j.is.2021.101814
12. Izquierdo, Y.T., et al.: A platform for keyword search and its application for covid-19 pandemic data. J. Inf. Data Manage. **12**(5) (2021). https://doi.org/10.5753/jidm.2021.1904, https://sol.sbc.org.br/journals/index.php/jidm/article/view/1904
13. Li, X., Yang, J., Ma, J.: Recent developments of content-based image retrieval (CBIR). Neurocomputing **452**, 675–689 (2021). https://doi.org/10.1016/j.neucom.2020.07.139
14. Nascimento, E.R., et al.: Text-to-SQL meets the real-world. In: (Accepted to the 26th International Conference on Enterprise Information System) (2024)
15. Nascimento, E.R., et al.: A family of natural language interfaces for databases based on chatGPT and langchain (short paper). In: Companion Proceedings of

the 42nd International Conference on Conceptual Modeling: Posters and Demos co-located with ER 2023, Lisbon, Portugal, November 06-09, 2023. CEUR Workshop Proceedings, vol. 3618 (2023). https://ceur-ws.org/Vol-3618/pd_paper_1.pdf
16. Nascimento, E.R., et al.: My database user is a large language model. In: (Accepted to the 26th International Conference on Enterprise Information System) (2024)
17. Nguyen, T.H., Prinz, A., Friiső, T., Nossum, R., Tyapin, I.: A framework for data integration of offshore wind farms. Renew. Energy **60**, 150–161 (2013). https://doi.org/10.1016/j.renene.2013.05.002
18. de Oliveira, P., da Silva, A., de Moura, E.: Ranking candidate networks of relations to improve keyword search over relational databases. In: 2015 IEEE 31st International Conference on Data Engineering, pp. 399–410 (2015). https://doi.org/10.1109/ICDE.2015.7113301
19. Ramada, M.S., Silva, J.C., Leitão-Júnior, P.S.: From keywords to relational database content: a semantic mapping method. Inf. Syst. **88**, 101460 (2020). https://doi.org/10.1016/j.is.2019.101460
20. Stonebraker, M., Ilyas, I.F.: Data integration: the current status and the way forward. IEEE Data Eng. Bull. **41**, 3–9 (2018). https://api.semanticscholar.org/CorpusID:49407081
21. Tautkute, I., Trzciński, T., Skorupa, A.P., Brocki, Ł, Marasek, K.: Deepstyle: multimodal search engine for fashion and interior design. IEEE Access **7**, 84613–84628 (2019). https://doi.org/10.1109/ACCESS.2019.2923552
22. Urbina Coronado, P.D., Lynn, R., Louhichi, W., Parto, M., Wescoat, E., Kurfess, T.: Part data integration in the shop floor digital twin: mobile and cloud technologies to enable a manufacturing execution system. J. Manuf. Syst. **48**, 25–33 (2018). https://doi.org/10.1016/j.jmsy.2018.02.002. special Issue on Smart Manufacturing
23. Vo, N., Jiang, L., Sun, C., Murphy, K., Li, L.J., Fei-Fei, L., Hays, J.: Composing text and image for image retrieval-an empirical odyssey. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6439–6448 (2019). https://doi.org/10.1109/CVPR.2019.00660
24. Yu, L., et al.: Commercemm: large-scale commerce multimodal representation learning with omni retrieval. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 4433–4442 (2022). https://doi.org/10.1145/3534678.3539151
25. Zeng, D., Yu, Y., Oyama, K.: Deep triplet neural networks with cluster-cca for audio-visual cross-modal retrieval. ACM Trans. Multimedia Comput. Commun. Appl. **16**(3), 1–23 (2020). https://doi.org/10.1145/3387164

Research Knowledge Graphs

The Effect of Knowledge Graph Schema on Classifying Future Research Suggestions

Dimitrios Alivanistos[1,2(✉)], Seth van der Bijl[1], Michael Cochez[1,2], and Frank van Harmelen[1]

[1] Department of Computer Science, Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands
seth.vanderbijl@icloud.com, {d.alivanistos,m.cochez,frank.van.harmelen}@vu.nl

[2] Discovery Lab. Elsevier, Radarweg 29a, 1043 NX Amsterdam, The Netherlands
https://discoverylab.ai

Abstract. The output of research doubles at least every 20 years and in most research fields the number of research papers has become overwhelming. A critical task for researchers is to find promising future directions and interesting scientific challenges in the literature. To tackle this problem, we hypothesize that structured representations of information in the literature can be used to identify these elements. Specifically, we look at structured representations in the form of Knowledge Graphs (KGs) and we investigate how using different input schemas for extraction impacts the performance on the tasks of classifying sentences as future directions. Our results show that the MECHANIC-Granular schema yields the best performance across different settings and achieves state of the art performance when combined with pretrained embeddings. Overall, we observe that schemas with limited variation in the resulting node degrees and significant interconnectedness lead to the best downstream classification performance.

Keywords: Information Extraction · Scientific Knowledge Graphs · Scientific Discourse · Classification

1 Introduction

Scientific papers often discuss future research directions and challenges, suggesting potential areas for further exploration. These are commonly found in the *future work* and *conclusion* sections and for multiple venues, they are a requirement for publication. Given a potentially infinite number of research trajectories, future research statements in papers attempt to guide researchers into promis-

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 149–170, 2024.
https://doi.org/10.1007/978-3-031-65794-8_10

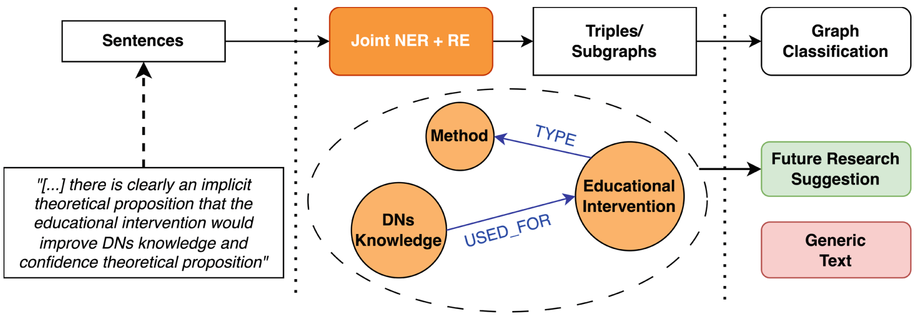

Fig. 1. Illustration of the pipeline with an example. First, a sentence (either research suggestion or random) is fed to a joint NER and RE model. The generated triples form graph(s). Lastly, these graphs form the input to the graph classification which classifies them as containing a *future research suggestion* or not.

ing directions. Knowledge gaps or unresolved questions may be identified and recommended as the most impactful directions.[1]

Multiple approaches have attempted to utilise Machine Learning (ML) and Natural Language Processing (NLP) to automatically identify such suggestions. In this work, we look at the problem of classifying future research suggestions when modelling the discourse not as text, but as a knowledge graph.

1.1 Problem Statement

Utilizing the future research directions to guide research becomes increasingly difficult due to the volume of published research. Several studies indicate that the growth of the size of research papers in terms of references, statistics, participants and tables [49] and especially the total volume of published studies in a given period increases continuously, leading to exponential growth [7,17,35,42,63]. Some studies suggest that research output currently doubles at least every 20 years with some periods of the 20th century seeing research output double every 7 years [34]. Meta-analysis can help condense information in certain disciplines but meta-analyses are subject to an even stronger volume increase [17], with specific fields seeing explosive growth [29]. The increasing volume in academic publications is an inspiring indicator of progress. However, the information overload is time-consuming [1] and can lead to a diminished quality in the interaction between researcher and research papers [44].

With the volume in research output increasing, the amount of potential future research directions becomes intractable. Thus, efficiently collecting and comparing future research indications becomes overly tedious for the average researcher. Furthermore, important directions might be overlooked, and researchers may

[1] Code, documentation and demo for exploring extracted triples: https://github.com/JoshuaSeth/Schema_Subgraph_Classification/tree/master and https://future-research.netlify.app/.

fail to combine research directions that appear in separate sources [16]. Groundbreaking discoveries might be missed as the right questions or directions remain hidden in the large amount of data [24]. With knowledge being mostly published in natural language, distributed among research papers in multiple journals and via diverse media, the question of effective communication in academic research is of paramount importance [27].

Although modelling scientific discourse has received increased attention, there have been few attempts focused on structuring future research suggestions, and summarize and communicate these efficiently.

In this work, we introduce an architecture that takes as input scientific sentences that contain future research suggestions (or not), transforms them into graphs via triple extraction, and evaluates the suitability of the extracted graphs for the downstream task of predicting whether a (sub)graph contains a research challenge or direction recommendation. The produced graphs are further analysed based on topological metrics, allowing for a better comparison of the schemas used to generate them.

1.2 Research Questions

Our main research questions (RQs) examine how schema choice influences triple extraction and the resulting graph topology (RQ1) and its subsequent impact on downstream graph classification (RQ2).

For RQ1, we explore and analyze the effect of schema choice on both local and global KGs in terms of topological features. In RQ2, we investigate how different schema characteristics impact the classification of (sub)graphs containing research recommendations. We also study the relationship between classification performance and graph topology. Lastly, we assess the effectiveness of graph classifiers for research suggestion classification.

2 Related Work

Scientific Paper Segmentation and Argumentation Extraction. Substantial research has been performed in extracting (semi-)structured data from research papers, such as scholarly argumentation mining (SAM) [5,38,41], research paper segmentation of text and figures [8,37], and parsing the figures of research papers [55]. Scientific metadata extraction focuses on extracting title, authorship and other metadata from articles [47]. Other approaches attempt to link the extracted concepts to articles or sentences from articles [32], also known as entity linking. Related research proposes KG based systems for recommending a scientific method or technique for a scientific problem [39]. Automated hypotheses generation generates promising hypotheses from research papers [56,65]. Furthermore, the shared task 11 of SemEval 2022 [43] of recognizing contributions of a paper brought increased attention to the problem of extracting knowledge from publications. Identifying the contributions of a paper can be a valuable task for researchers looking to build on existing work as it can lead to better recommendations. Lastly, subjectivity analysis has received attention, since filtering out the subjective sentences from a paper appears to improve Information Extraction (IE) [50,64].

KG Construction for Scientific Discourse. Knowledge Graphs are graph structures (networks of labelled nodes and edges) [15] with additional semantics [9]. Entities and relations in a KG are expressed in the form of triples of *subject, predicate, object* that correspond to an edge (predicate) between two nodes (subject and object). Entities and relations are often typed with a class.

KGs can be constructed using several methods. Automated approaches attempt to model research content into a KG by employing pipelines involving Natural Language Processing (NLP) methods, rather than hand-engineering. These automated Information Extraction (IE) approaches can model scientific knowledge in the form of a graph of (scientific) triples. Initially, IE methods employed domain-specific, rule-based systems while recent years have seen approaches employing Machine Learning (ML) [52] and Deep Learning (DL) take over [24]. For these approaches, a ML model is trained based on a "high-quality" dataset, often manually annotated. The trained model is then applied to a larger text dataset to automatically extract entities and relations to build a KG. Examples of such KGs are the COVID-19 themed CORD-KG (generated with DyGIE++ trained on MECHANIC data, [25]), the material sciences themed MatKG (build with an LLM transformer, [60]), or the AI-based Intelligence Task Ontology (ITO) KG [6]. Employing AI/ML for the generation of KGs allows for automatic generation on a larger scale. However, automatic generation can be less robust to erroneous input data and the resulting graph can be of low quality. Different disciplines tend to exhibit large discrepancies in adopting IE [24].

Alternatively, KGs can be constructed by hand or by extracting data from semi-structured datasets (e.g. patient records), without employing ML approaches. These KGs are often high-quality but small-scale as human annotations are time-consuming and depend on the expertise of the annotator. Some examples include the chemical protein interactions of ChemProt [58] and adverse drug events (ADE) [19].

3 Methodology

The goal of this research is to construct a graph from sentences suggesting future research, observe and analyse the topological features of said graph, and test the influence of the schema on the downstream task of graph classification. To achieve this we designed the following pipeline. Starting from the dataset created by [25], we take future research suggestion sentences (either a *challenge* or a *direction*) from papers and extract their corresponding triples using a joint Named Entity Recognition (NER) and Relation Extraction (RE) model pre-trained on different selected schemas. Although we understand the importance of NER/RE model performance on the resulting graphs and hence downstream task, in this work we are focusing on the effect of their underlying schema on downstream performance, since more generic schemas can for example result in higher recall but lower precision. We then analyse the resulting graphs both locally (subgraph-level, i.e. collection of triples) and globally (full KG) in terms of topological features. Finally, we employ Relational Graph Convolutional Networks (R-GCNs) to perform graph classification on the resulting graphs.

The rest of this section describes the input data and schema choices and the graph classification using R-GCNs, including details around the architecture and experimental setup. The pipeline is illustrated by Fig. 1.

3.1 Schema and Dataset

There are multiple options when choosing a schema for modelling research content. An extensive selection can be found in Table 5 of the Appendix A. Amongst the listed options, six candidates were considered fit for this work. These are the MECHANIC (Coarse-Granular), SciERC, ACE05, ACE-Event and GENIA. The criteria for the selection were based on granularity and generalizability. For example SciERC has several entity and relationship types, whereas MECHANIC-Coarse only has a single entity type and two relationship types. Concerning generalizability, MECHANIC-Coarse is about general science, while GENIA is specifically about biology. Another motivation for the chosen schemas was their accessibility in the pretrained DyGIE++ joint NER and RE model, which allowed for a straight-forward comparison. The performance of DyGIE++ for NER/RE can be found in the original DyGIE++ paper [61]. The results of our analysis focused on the schemas found in Table 1. A comparison of the different entity and relation types defined by each schema can be found in the Appendix A in Table 7.

Table 1. The selected datasets and schemas investigated in this work. The difference between 0 and N/A for the number of entities or relations is that for the former no entities or relations were defined (like the ACE-Event schema) while for the latter it was OpenIE.

Dataset + Schema	Domain	Size	Entities	Relations	Characteristics
SciERC [40]	CS	500 abstracts	6	7	Coarse-grained, domain-specific
MECHANIC-Coarse [25]	Bio	1000 sentences	1	2	Coarse-grained, domain-specific
MECHANIC-Granular [25]	Bio	1000 sentences	N/A	N/A	Fine-grained, domain-specific
ACE05 [62]	Various	511 documents	7	6	Complex, general
ACE-Event [61]	Various	599 documents	6	18	Complex, general
GENIA [30]	Bio	2,000 abstracts	6	5	Fine-grained, domain-specific

SciERC is a dataset accompanied by a simple schema aimed at extracting methods, tasks and metrics from Computer Science and Artificial Intelligence (CS/AI) abstracts. Nevertheless, it shows good generalisability when detecting concepts and relations in future research sentences outside of CS/AI. GENIA was meant for DNA, RNA and proteins, hence it performs well when dealing with biomedical data. Since it lacks generalisability, it would not be a perfect fit as the basis of a scientific modelling schema, but could be beneficial as an addition to another, more general schema. The ACE05 dataset was available in the ACE-Event and ACE05 format, where the ACE-Event is a filtered version of the ACE05 dataset focused on identifying events. As such, the ACE-Event dataset had different entity types and extended (sub)relation types. ACE05 relations and entities were designed for capturing news events involving people, organizations, locations, movements, and concepts that are physical in nature. This makes it interesting to experiment with for scientific content. Finally, the two MECHANIC schemas were created by [25] in the context of designing a knowledge base of mechanisms extracted from COVID-19 papers. A coarse-grained and fine-grained version were defined, with the coarse-grained version detecting entities of type *Entity* and relations of type *mechanism* (direct mechanisms) and *effect* (indirect mechanism) in sentences. The fine-grained version was closer to Open Information Extraction (OpenIE), where the verb of the sentence would denote the relation (e.g. "COVID-19 influences diabetes" results in "influences" as the relation type [25]).

3.2 Graph Classification of Challenges and Directions

Motivation. Graph Convolutional Networks (GCNs) [31], are a class of neural architectures that operate on graph-structured data and leverage its topological features. GCNs can be applied to graph classification tasks, i.e. classifying a graph given its nodes and edges. An advantage of GCN architectures for graph classification lies in their ability to integrate both node attributes and graph topology into the node representations, which can in turn encode global and local characteristics of the graphs.

GCNs can learn node representations that are more expressive and discriminative than random walk-based methods, which solely rely on the graph structure and ignore features like node attributes. GCNs can handle graphs with varying sizes and structures. Furthermore, they can parse different types of graphs, such as directed, weighted, or heterogeneous graphs, by using the appropriate convolution operators. Therefore, for this work we considered GCNs for graph classification.

Architecture and Implementation. Initially we pretrained the GCN with a Link Prediction (LP) objective to obtain meaningful node representations. In absence of pre-existing node features, GCN models use a random initialisation. With the use of a pretraining routine involving a separate GCN model trained on a Link Prediction (LP) objective we obtain meaningful node representations, which are subsequently used to initialise the graph classification models. We investigate the effect of this initialisation on the performance in the graph classification task. Below we describe the architecture in further detail.

The architecture for pretraining consists of 2 GCN layers. The first layer takes randomly initialized node embeddings (5 channels) while outputting 128 channel encodings. The second GCN layer takes the 128 hidden channels and learns embeddings with a dimension of 64 channels from the 128 hidden channels. In the decoding phase, the decoder layer operates on the node embeddings created by the preceding GCN layers. For each edge $e = (i, j)$, the decoder computes the score for that edge as the dot product of the embeddings of nodes i and j. The score is a scalar which is interpreted as the likelihood of the existence of an edge between nodes i and j. For each positive edge e we sample one negative edge $\hat{e}$ where either i or j is replaced by a node sampled uniformly at random, i.e. corrupted. After training on LP the encoder part of the model was employed to get the initial embeddings for all nodes in the (sub)graphs of the graph classification task.

For graph classification we experimented with both a GCN and a Relational-GCN (R-GCN, [54]). At each layer the GCN model applies a linear transformation to the node features and aggregates the features from the neighboring nodes using a first-order approximation of spectral graph convolutions. In contrast, the R-GCN model extends the conventional convolution operation introducing relation-specific weights. Here, at each layer, for each node j the representations of neighboring nodes i (those with an incoming edge (i, j)) are aggregated after passing them through a relation specific linear layer. In our experiments we investigate the influence of this model choice.

For both GCN and R-GCN, the architecture as illustrated in Fig. 2 is equivalent by swapping the GCN layer(s) with R-GCN layer(s). Each (sub)graph is initially passed to three (relational) graph convolution layers. Then, we obtain a single embedding for each graph by performing a mean pooling over the node representations. Finally, we apply a dropout layer for regularization and a final linear layer to perform the graph classification.

3.3 Experimental Setup

In this section we describe our experimental setup, including the dataset, the extraction and finally the classification, including training and pretraining of the model.

The Dataset. The gold labeled sentences are the ones provided by [33]. In the original dataset these sentences were classified as either a research challenge, a research direction, both, or neither. In that work, the focus was on building a search engine that would distinguish between the two classes and allow for their retrieval separately. However, in the current work we do not make the same distinction, because we are interested in discovering all future research recommendations, which include both challenges and directions. Hence, we cast the problem into a binary classification problem, where each sentence is labeled positive if it contains either a research challenge, a research direction, or both, and negative otherwise.

Joint NER and RE. Entities and relations forming subgraphs were extracted from the sentences employing the DYGIE++ model. The pretrained models

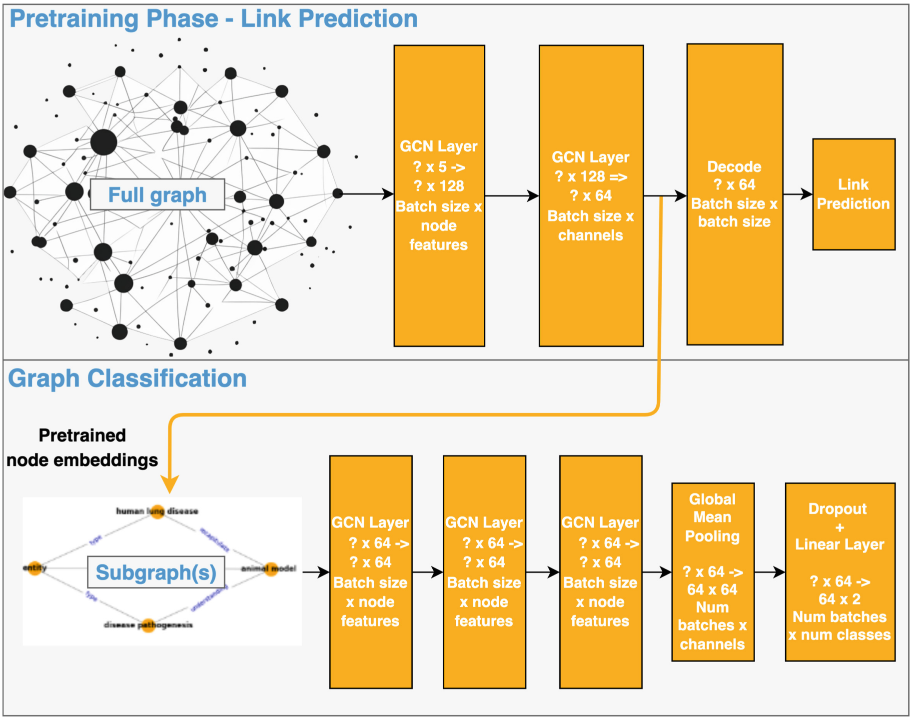

Fig. 2. Illustration of using pretrained embeddings as initial node embeddings for the local subgraphs per sentence. Embeddings are learned by LP on the full graph using entities and relations from all sentences. The learned embeddings are used as initial node embeddings for the entities of the smaller graphs for each sentence which are classified using the (R)-GCN.

hosted in the DYGIE++ repository[2] [61] were used. To improve performance over scientific text, we replaced BERT with SciBERT [4] as our encoder, which is fine-tuned on scientific data. Apart from this, default settings were used. For each schema, the DYGIE++ model extracted entities and relations. From the resulting entities and relations a (global) graph representing the full body of sentences was constructed.

Graph Classification and Pretraining. The pipeline for the LP task begins by loading and preprocessing each extracted graph dataset. These graph datasets are then randomly divided into training, validation, and test sets, with 5% of edges set aside for validation and 10% for testing. The models are parameterized based on the feature dimensions of the nodes in the graph and are optimized using the Adam optimizer with a learning rate of 0.001. The loss function used for the pretraining task is the Binary Cross-Entropy (BCE) with Logits Loss. The models are then trained and subsequently evaluated on the test data, with the

[2] https://github.com/dwadden/dygiepp.

Table 2. Local graph topology metrics under different schemas. Relatively high values are in bold and relatively low values are in italics. Some statistics are first aggregated per subgraph and then averaged.

	ACE-Event	ACE05	GENIA	MECHANIC-Coarse	MECHANIC-Granular	SciERC
Entities	*4.730* ± 3.240	5.131 ± 3.551	*3.386* ± 2.223	**9.118** ± 9.513	6.206 ± 3.535	**7.992** ± 6.389
Relations	*2.243* ± 1.517	2.430 ± 1.617	*1.625* ± 1.033	**4.230** ± 3.542	3.062 ±± 1.727	**3.951** ± 3.143
Degrees	0.948 ± 0.729	*0.947* ± 0.784	0.960 ± 0.669	*0.928* ± 1.346	**0.987** ± 1.099	**0.989** ± 1.032
Clusterings	0.012 ± 0.103	*0.011* ± 0.100	*0.000* ± 0.000	**0.204** ± 0.374	**0.275** ± 0.434	0.080 ± 0.251
Modularities	0.185 ± 0.236	**0.201** ± 0.243	0.071 ± 0.174	*0.029* ± 0.075	*0.023* ± 0.060	**0.290** ± 0.229

Table 3. Global graph topology metrics under different schemas. Relatively high values are in bold while relatively low values are in italics.

	ACE-Event	ACE05	GENIA	MECHANIC-Coarse	MECHANIC-Granular	SciERC
Entities	*1568*	1645	*1065*	**7119**	4240	**6918**
Relations	*2677*	2827	*1873*	**11583**	6367	**9282**
Degrees	3.415 ± 21.106	**3.437** ± 21.986	**3.517** ± 20.209	3.254 ± 72.537	*3.003* ± 61.855	*2.683* ± 47.125
Clusterings	*0.054* ± 0.193	0.057 ± 0.199	*0.039* ± 0.154	**0.508** ± 0.444	**0.557** ± 0.482	0.175 ± 0.348
Modularities	**0.708**	**0.699**	0.669	*0.510*	*0.397*	0.635

Area Under the ROC Curve (AUC) score being computed as a measure of model performance. The subsequent graph classification divided the subgraphs in 75% training graphs, 12.5% validation graphs and 12.5% test graphs. The classes were roughly balanced. The models were similarly optimized with the Adam optimizer with a learning rate of 0.001 using Binary Cross-Entropy (BCE) with Logits Loss. In the graph classification case precision, recall and F1 were used as measure of performance.

4 Results

The following section analyses the results as obtained with quantifiable metrics. Main patterns are noted, interpretation and contextualization of these results follows in the discussion section. We chose to focus on the resulting entities and relations, the node degree, clustering coefficient and modularity as they characterize numerically the topology of the generated graphs. The analysis is performed on both local and global level, i.e. over each subgraph and over the entire extracted graph.

Joint NER and RE. As expected, the joint NER and RE for the different schemas resulted in different graphs. These are described and summarized through network graph metrics. We observe that the resulting graphs differ greatly in terms of topologies both visually and using metrics. The global full interconnected graph statistics are summarized in Table 3. In parallel, the local graph statistics for the subgraph of each sentence are presented in Table 2.

Entities and Relations. The MECHANIC and SciERC schemas yield a higher number of detected entities and relations for both the global graph and local subgraphs. From our experiments we observe that a higher number of entities and relations is associated with better downstream task performance.

Node Degree. Certain schemas result in high variation in terms of node degrees in the KG. Most schemas result in the majority of nodes having a low degree and a long tail of extremely connected nodes. GENIA has a mean degree on the higher side of the spectrum whereas MECHANIC-Granular and SciERC are positioned in the lower end of the spectrum. In terms of local subgraphs, SciERC and MECHANIC-Granular appear more interconnected, with higher degrees than other schemas, although the differences in terms of local degrees are subtle. In general, the standard deviation of these degrees increases with the average degree for local subgraphs. A higher standard deviation in the degree appears to be related to better performance in the downstream task in most of the settings. This phenomenon appears sensible since more densely connected subgraphs provide more insight in the relations between concepts in a subgraph. On a global level, during pretraining, without using types, degree centrality is the only measure that correlates with graph classification performance.

Clustering Coefficients. Initially we expected a relation between degree standard deviation and clustering coefficient. However, this was not universally the case. MECHANIC-Granular and MECHANIC-Coarse exhibited very high clustering coefficients, while GENIA and the ACE schemas were on the low-side. This indicates presence of cliques was rather average in the SciERC graph. This positive influence of clustering coefficients was noted on both the global graph and local subgraph levels of the metrics.

Modularity. In terms of modularity, SciERC, GENIA, ACE05 and ACE-Event resulted in rather tightly knit communities with few edges connected to other communities; MECHANIC-Granular and MECHANIC-Coarse produced graphs with communities that were less connected internally but more connected to other communities in the graph. This modularity is more intuitively illustrated in the network graph plots found in the demo. On a local level, modularity does not always relate to average degree. Some schemas are very interconnected but do not exhibit clear subcommunities, such as MECHANIC-Granular. On a global level higher modularity appears to lead to worse performance with high or moderate modularity schemas showing limited performance. The same holds on the local (subgraph) level, with the exception of SciERC which performs well even when having a high subgraph modularity.

4.1 Graph Classification

Results are aggregated based on whether the model was pretrained (+Pretrained), whether the model captured different relation types (+Relations) and for both. Table 4 denotes the average performance over 20 runs for the different combinations of these settings. In general, we observed that pretrained models (over the global extracted graph), exhibit a strong ability in the detection of future research challenges and directions.

Table 4. F1 scores when predicting the future research suggestion label [0, 1] (generic text vs. future research suggestion). (R-)GCNs were applied to the local subgraphs per sentence. The *+Pretrained* indicates whether pretrained embeddings were used. *+Relations* indicates whether an R-GCN was used instead of a GCN for the local subgraph classification, to incorporate relation-type information. *+Pretrained + Relations* indicates the use of both. Best scores per mode are in bold and best combination of configuration (pretrained embeddings and typed relations) is underlined.

	F1	P	R
ACE-Event	0.494	0.489	0.506
+ Pretrained	0.511	0.462	0.580
+ Relations	0.737	0.723	0.757
+ Pretrained + Relations	<u>0.964</u>	0.976	0.953
ACE05	0.469	0.452	0.497
+ Pretrained	0.527	0.486	0.580
+ Relations	0.864	0.888	0.842
+ Pretrained + Relations	<u>0.978</u>	0.971	0.985
GENIA	0.403	0.423	0.402
+ Pretrained	0.369	0.354	0.394
+ Relations	0.375	0.387	0.376
+ Pretrained + Relations	<u>0.579</u>	0.510	0.682
MECHANIC-Coarse	0.544	0.481	0.629
+ Pretrained	0.602	0.536	0.690
+ Relations	0.878	0.871	0.887
+ Pretrained + Relations	<u>0.990</u>	0.990	0.991
MECHANIC-Granular	**0.594**	0.507	0.723
+ Pretrained	0.620	0.518	0.775
+ Relations	**0.906**	0.903	0.911
+ Pretrained + Relations	<u>**0.994**</u>	1.000	0.988
SciERC	0.560	0.478	0.684
+ Pretrained	**0.644**	0.556	0.771
+ Relations	0.884	0.877	0.893
+ Pretrained + Relations	<u>0.991</u>	0.991	0.991

In Table 4 we observe that the best performing run utilises both relation type information and the pretraining of node embeddings. In this setup, our model gives results comparable to the state-of-the-art in detecting future research suggestions.

5 Discussion

Overall, depending on the schema, we observed diverse graph topologies. For example, some resulted in more modular graphs compared to others. In specific cases, we observed that there were no entities or relations extracted at all. This could hint at the schema being unsuitable for the domain of choice. In parallel, the lack of extractions could imply that the sentence does not contain a research suggestion. It is noteworthy that schemas with a lower average degree generally also have much higher standard deviations in their degree. This indicates some very connected nodes and many sparsely connected nodes. In the current set-up it appears that the standard deviation and mean of the clustering coefficients on a local level are influential factors on downstream task performance. Another observation is that for most schemas/datasets the effect of the inclusion of relation types appears to have a stronger impact on performance in comparison to pretraining on the entire graph. Different schemas produce different graphs, and so differences emerge in the pretraining of embeddings, relation type usage and performance. When pretrained embeddings or relation types are not being utilised for classification, SciERC and MECHANIC-Granular consistently outperform the rest. Performing future research suggestion classification without pretraining and relation types proves difficult for any schema, resulting in poor performance on this setting, with the single exception being MECHANIC-Granular.

5.1 Limitations and Future Research

Future research can extend the present results in several directions. More schemas and input data, different joint NER and RE models and different downstream tasks (e.g. entity linking) to list just a few potential extensions. While graph classification fits well to our task and purpose for testing the influence of the schemas, other downstream tasks, such as link prediction might be less sensitive to the graph topology resulting from a choice of schema given the setting of predicting the likelihood of a subject node being connected to an object node. Additionally, we tested a single joint NER and RE model for several different graphs. While DyGIE++ provides a baseline model for joint NER and RE, other more powerful extraction models may influence the resulting graph topology (e.g. detect more entities). DyGIE++ however has long been the state of the art and provides easily accessible models. The present research characterizes the fitness of a schema for scientific modelling by how it influences our graph topology and downstream classification task. However, downstream graph ML tasks can be influenced by multiple different parameters (i.e. hyper-parameter tuning), and isolating the effect of the schema might be challenging. Additionally, a limitation is observed over the gold set of [33] since the produced dataset is focused on COVID-19 research and hence is a domain-specific dataset. An intriguing direction of future research could employ OpenIE to dynamically construct schemas, with their usefulness evaluated by their performance on several downstream tasks in parallel (LP, graph classification etc.). This would yield schemas optimised for several downstream tasks at once, hence increasing robustness (Figs. 3, 4 and Table 6).

6 Conclusion

In this work we analysed the effect of the choice of schema when extracting knowledge from text in the form of KGs, to be further used for scientific knowledge discovery and recommendation. Specifically, we experimented with extracting graphs from sentences containing a scientific research suggestion or not, by employing pretrained models of DyGIE++ with different underlying schemas. We observed that the choice of schema can have a significant influence on both graph topology and downstream graph classification performance. Moreover, we observe that there is a correlation between several topology metrics of the resulting graphs and downstream task performance. The MECHANIC-Granular schema leads to solid downstream task performance with state of the art detection of future research suggestions when combined with pretrained embeddings and typed relations.

Acknowledgments. We would like to thank Paul Groth and Thom Pijnenburg for their valuable comments and insights to this work.

A Tables and Figures

Table 5. Summary of existing schemas and datasets for joint NER and RE from research papers. Along with indications of the number of different entities and relations they have. The size is the number of annotated items.

Dataset	Domain	Size	Entities	Relations	Limitations
SciERC [40]	CS	500 abstracts	6	7	Coarse-grained, domain-specific
MECHANIC-Coarse [25]	Bio	1000 sentences	1	2	na
MECHANIC-Granular [25]	Bio	1000 sentences	N/A	N/A	na
ChemProt [58]	Bio	1,820 abstracts	2	12	Imbalanced, domain-specific
ADE [19]	Bio	6,821 sentences	2	1	Sparse, noisy, narrow
DDI [23]	Bio	1017 abstract	4	4	Imbalanced, domain-specific
CoNLL04 [51]	News	1,441 sentences	4	5	Low agreement, general
ACE05 [62]	Various	511 documents	7	6	Complex, general
ACE-Event [61]	Various	599 documents	6	18	Complex, general
GENIA [30]	Bio	2,000 abstracts	6	5	Fine-grained, domain-specific
ACE04 [45]	Various	451 documents	7	6	Complex, general
DocRED [66]	Various	5,053 documents / 101k	9	96	Document-level
TACRED [67]	Various	106k sentences	N/A	41	Sentence-level
SemEval-2010 Task 8 [22]	Various	10,717 sentences	N/A	8	Sentence-level
WebNLG [18]	Various	21,855 data/text pairs	N/A	N/A	RDF-to-text
NYT [53]	Various	2,15M annotated docs	N/A	N/A	Summarization
GDA [3]	Bio	30k sentences	3	1	Gene-disease association
BC5CDR [36]	Bio	1,500 articles	2	1	Chemical-disease relation
Retacred [57]	Various	106K sentences	N/A	41	Biomedical relation
Redocred [59]	Various	5K documents	9	96	Document-level biomedical relation
FewREL [21]	Various	70,000 sentences	N/A	100	Few-shot relation classification
KPI-EDGAR [11]	Financial	1355 sentences	12	4	Joint NER and RE
T-REx [14]	Various	6.2m sentences	N/A	642	Semantic Relation Classification
ACL RD-TEC 2.0 [48]	CL	300 abstracts	7	0	0
SemEval 2017 Task 10: ScienceIE [2]	Various	500 abstracts	3	2	0
SemEval-2018 Task 7 [20]	CL	500 abstracts	0	6	0
ARC-PDN [26]	Various	4k docs	4	1	0
STEM-ECR v1.0 [13]	ML	332 abstracts	4	0	0
NLP TDMS [46]	AI	30k docs	3	4	-
SciREX [28]	CL, Computer Vision	438 docs	4	1	-
AI-KG [10]	CS, ML	333k	5	9	-
SemEval-2021 Task 11 [12]	CL	442 docs	1	0	-

Table 6. Local graph topology metrics. For OR mode. Relatively high values are marked blue with relatively low values being marked red. Density is reported with 5 decimal precision and other factors with 3 decimals unless they are natural numbers. Some statistics for these small subgraphs are on node-level. This means that the presented statistic is an aggregation of an aggregation (for example the mean of the mean centralities of all subgraphs). This statistic could have been represented by listing the properties for every node and obtaining the statistic from that, but it is intentionally represented by aggregating the statistic on the subgraph level and obtaining the statistic over the different subgraphs to give an improved impression of the subgraph properties rather than the properties of the nodes in them. As a consequence, not all node statistics are weighed equally since some subgraphs consist of more nodes and other subgraphs consist of less nodes. Relations are after the addition of entity type relations.

	ACE-Event	ACE05	GENIA	MECHANIC-Coarse	MECHANIC-Granular	SciERC
Entities	*4.730* ± 3.240	5.131 ± 3.551	*3.386* ± 2.223	**9.118** ± 9.513	6.206 ± 3.535	**7.992** ± 6.389
Relations	*2.243* ± 1.517	2.430 ± 1.617	*1.625* ± 1.033	**4.230** ± 3.542	3.062 ± 1.727	**3.951** ± 3.143
Degrees	0.948 ± 0.729	*0.947* ± 0.784	0.960 ± 0.669	*0.928* ± 1.346	**0.987** ± 1.099	**0.989** ± 1.032
Degree centralities	**0.311** ± 0.329	0.278 ± 0.308	**0.494** ± 0.403	*0.146* ± 0.245	0.221 ± 0.286	*0.164* ± 0.222
Closeness centralities	**0.336** ± 0.324	0.307 ± 0.303	**0.522** ± 0.388	*0.162* ± 0.250	0.243 ± 0.287	*0.187* ± 0.225
Clusterings	0.012 ± 0.103	*0.011* ± 0.100	*0.000* ± 0.000	**0.204** ± 0.374	**0.275** ± 0.434	0.080 ± 0.251
Modularities	0.185 ± 0.236	**0.201** ± 0.243	0.071 ± 0.174	*0.029* ± 0.075	*0.023* ± 0.060	**0.290** ± 0.229
Densities	**0.505** ± 0.380	0.461 ± 0.368	**0.697** ± 0.371	*0.345* ± 0.362	0.379 ± 0.359	*0.321* ± 0.328

Table 7. The different entity and relation types defined by each underlying schema. We can observe that some schemas employ much more generic entity types than others e.g. "Entity" vs "Cell type"

Schema	SciERC	M-Coarse	M-Granular	ACE05	ACE-Event	GENIA
Entity Types	Task Method Metric Material Other-ScientificTerm Generic	Entity	Entity	Person Organization Geo-Political Entity Location Facility Vehicle Weapon	Person Organization Geo-Political Entity Location Facility Time Weapon	Protein DNA RNA Cell line Cell type Other
Relation Types	Compare Part-of Conjunction Evaluate-for Feature-of Used-for Hyponym-of	Mechanisms Effect	- (OpenIE)	Physical (PHYS) Part-Whole(PART-WHOLE) Artifact(ART) General Affiliation(GEN-AFF) Organization Affiliation(ORG-AFF) Personal Social(PER-SOC)	ORG-AFF.Employment PHYS.Located PART-WHOLE.Geographical ART.User-Owner-Inventor-Manufacturer GEN-AFF.Citizen-Resident-Religion-Ethnicity ORG-AFF.Membership PART-WHOLE.Subsidiary GEN-AFF.Org-Location PHYS.Near PER-SOC.Family PER-SOC.Business ORG-AFF.Founder ORG-AFF.Sports-Affiliation ORG-AFF.Investor-Shareholder ORG-AFF.Student-Alum ORG-AFF.Ownership PER-SOC.Lasting-Personal PART-WHOLE.Artifact	Component-of Subunit-of Site-of Product-of Theme-of

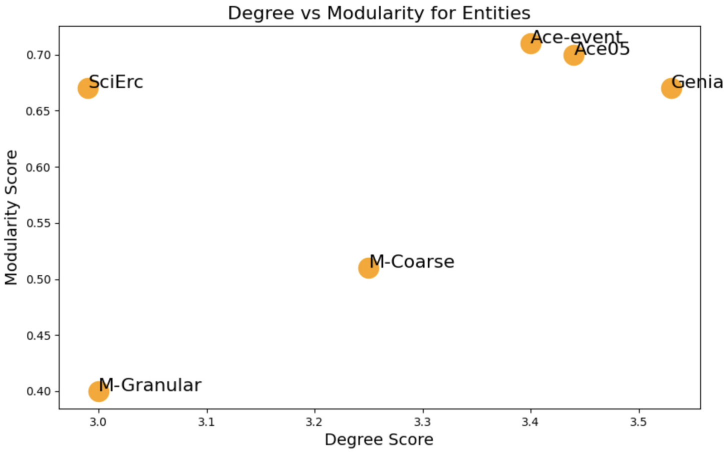

Fig. 3. Global spectrum of major graph topology metric

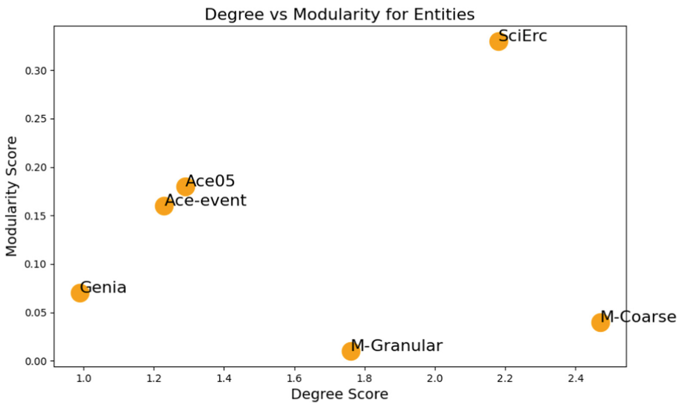

Fig. 4. Local spectrum of major graph topology metrics

References

1. Achike, F.I., Ogle, C.W.: Information overload in the teaching of pharmacology. J. Clin. Pharmacol. **40**(2), 177–183 (2000). https://doi.org/10.1177/00912700022008838, https://onlinelibrary.wiley.com/doiabs/10.1177/00912700022008838
2. Augenstein, I., Das, M., Riedel, S., Vikraman, L., McCallum, A.: SemEval 2017 task 10: ScienceIE - extracting keyphrases and relations from scientific publications. In: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), pp. 546–555. Association for Computational Linguistics, Vancouver, Canada (2017). https://doi.org/10.18653/v1/S17-2091, https://aclanthology.org/S17-2091
3. Becker, K.G., Barnes, K.C., Bright, T.J., Wang, S.A.: The genetic association database. Nat. Genet. **36**(5), 431–432 (2004). https://doi.org/10.1038/ng0504-431

4. Beltagy, I., Lo, K., Cohan, A.: Scibert: pretrained language model for scientific text. In: EMNLP (2019)
5. Binder, A., Verma, B., Hennig, L.: Full-text argumentation mining on scientific publications. None (2022). https://doi.org/10.48550/ARXIV.2210.13084, https://arxiv.org/abs/2210.13084, publisher: arXiv Version Number: 1
6. Blagec, K., Barbosa-Silva, A., Ott, S., Samwald, M.: A curated, ontology-based, large-scale knowledge graph of artificial intelligence tasks and benchmarks. Scientific Data **9**(1), 322 (2022). https://doi.org/10.1038/s41597-022-01435-x, https://www.nature.com/articles/s41597-022-01435-x
7. Boschen, M.J.: Publication trends in individual anxiety disorders: 1980–2015. J. Anxiety Disord. **22**(3), 570–575 (2008). https://doi.org/10.1016/j.janxdis.2007.04.004, https://www.sciencedirect.com/science/article/pii/S0887618507001016
8. Bui, D.D.A., Del Fiol, G., Jonnalagadda, S.: PDF text classification to leverage information extraction from publication reports. J. Biomed. Inform. **61**, 141–148 (2016). https://doi.org/10.1016/j.jbi.2016.03.026, https://www.sciencedirect.com/science/article/pii/S153204641630017X
9. Davies, J., Fensel, D., Harmelen, F.v.: Towards the Semantic Web: Ontology-driven Knowledge Management. Wiley, Hoboken (2003). google-Books-ID: kRE-OBAAAQBAJ
10. Dessì, D., Osborne, F., Reforgiato Recupero, D., Buscaldi, D., Motta, E., Sack, H.: AI-KG: an automatically generated knowledge graph of artificial intelligence. In: Pan, J.Z., et al. (eds.) ISWC 2020. LNCS, vol. 12507, pp. 127–143. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62466-8_9
11. Deußer, T., et al.: KPI-EDGAR: a novel dataset and accompanying metric for relation extraction from financial documents. In: 2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 1654–1659 (2022). https://doi.org/10.1109/ICMLA55696.2022.00254, http://arxiv.org/abs/2210.09163, arXiv:2210.09163 [cs]
12. D'Souza, J., Auer, S., Pedersen, T.: SemEval-2021 task 11: NLPContribution-Graph - structuring scholarly nlp contributions for a research knowledge graph. In: Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), pp. 364–376. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.semeval-1.44, https://aclanthology.org/2021.semeval-1.44
13. D'Souza, J., Hoppe, A., Brack, A., Jaradeh, M.Y., Auer, S., Ewerth, R.: The STEM-ECR dataset: grounding scientific entity references in stem scholarly content to authoritative encyclopedic and lexicographic sources. In: Proceedings of the Twelfth Language Resources and Evaluation Conference, pp. 2192–2203. European Language Resources Association, Marseille, France (2020). https://aclanthology.org/2020.lrec-1.268
14. Elsahar, H., et al.: T-REx: a large scale alignment of natural language with knowledge base triples. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018). European Language Resources Association (ELRA), Miyazaki, Japan (2018). https://aclanthology.org/L18-1544
15. Euler, L.: Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae, pp. 128–140 (1741). https://scholarlycommons.pacific.edu/euler-works/53
16. Feroz, H.M.B., Zulfiqar, S., Noor, S., Huo, C.: Examining multiple engagements and their impact on students' knowledge acquisition: the moderating role of information overload. J. Appl. Res. High. Educ. **14**(1), 366–393 (2021). https://doi.org/10.1108/JARHE-11-2020-0422

17. Fontelo, P., Liu, F.: A review of recent publication trends from top publishing countries. Syst. Rev. **7**(1), 147 (2018). https://doi.org/10.1186/s13643-018-0819-1
18. Gardent, C., Shimorina, A., Narayan, S., Perez-Beltrachini, L.: The WebNLG challenge: generating text from RDF data. In: Proceedings of the 10th International Conference on Natural Language Generation, pp. 124–133. Association for Computational Linguistics, Santiago de Compostela, Spain (2017). https://doi.org/10.18653/v1/W17-3518, https://aclanthology.org/W17-3518
19. Gurulingappa, H., Rajput, A.M., Roberts, A., Fluck, J., Hofmann-Apitius, M., Toldo, L.: Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports. J. Biomed. Inform. **45**(5), 885–892 (2012). https://doi.org/10.1016/j.jbi.2012.04.008
20. Gábor, K., Buscaldi, D., Schumann, A.K., QasemiZadeh, B., Zargayouna, H., Charnois, T.: SemEval-2018 Task 7: semantic relation extraction and classification in scientific papers. In: Proceedings of the 12th International Workshop on Semantic Evaluation, pp. 679–688. Association for Computational Linguistics, New Orleans, Louisiana (2018). https://doi.org/10.18653/v1/S18-1111, https://aclanthology.org/S18-1111
21. Han, X., et al.: FewRel: a large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 4803–4809. Association for Computational Linguistics, Brussels, Belgium (2018). https://doi.org/10.18653/v1/D18-1514, https://aclanthology.org/D18-1514
22. Hendrickx, I., et al.: SemEval-2010 Task 8: multi-way classification of semantic relations between pairs of nominals. In: Proceedings of the 5th International Workshop on Semantic Evaluation, pp. 33–38. Association for Computational Linguistics, Uppsala, Sweden (2010). https://aclanthology.org/S10-1006
23. Herrero-Zazo, M., Segura-Bedmar, I., Martínez, P., Declerck, T.: The DDI corpus: an annotated corpus with pharmacological substances and drug-drug interactions. J. Biomed. Inform. **46**(5), 914–920 (2013). https://doi.org/10.1016/j.jbi.2013.07.011
24. Hong, Z., Ward, L., Chard, K., Blaiszik, B., Foster, I.: Challenges and advances in information extraction from scientific literature: a review. JOM **73**(11), 3383–3400 (2021). https://doi.org/10.1007/s11837-021-04902-9
25. Hope, T., et al.: Extracting a knowledge base of mechanisms from COVID-19 papers (2021). https://doi.org/10.48550/arXiv.2010.03824, http://arxiv.org/abs/2010.03824, arXiv:2010.03824 [cs]
26. Hou, Y., Jochim, C., Gleize, M., Bonin, F., Ganguly, D.: Identification of tasks, datasets, evaluation metrics, and numeric scores for scientific leaderboards construction. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 5203–5213. Association for Computational Linguistics, Florence, Italy (2019). https://doi.org/10.18653/v1/P19-1513, https://aclanthology.org/P19-1513
27. Ifidon, E.I., Ugwuanyi, R.N.: Effective communication in academic libraries: an imperative for knowledge delivery. Int. J. Lib. Inf. Sci. **5**(7), 203–207 (2013)
28. Jain, S., van Zuylen, M., Hajishirzi, H., Beltagy, I.: SciREX: a challenge dataset for document-level information extraction. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 7506–7516. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.acl-main.670, https://aclanthology.org/2020.acl-main.670
29. Kalantari, A., et al.: A bibliometric approach to tracking big data research trends. J. Big Data **4**(1), 30 (2017). https://doi.org/10.1186/s40537-017-0088-1

30. Kim, J.D., Wang, Y., Yasunori, Y.: The Genia event extraction shared task, 2013 edition - overview. In: Proceedings of the BioNLP Shared Task 2013 Workshop, pp. 8–15. Association for Computational Linguistics, Sofia, Bulgaria (2013). https://aclanthology.org/W13-2002
31. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2017). https://doi.org/10.48550/arXiv.1609.02907, http://arxiv.org/abs/1609.02907, arXiv:1609.02907 [cs, stat]
32. Krallinger, M., Valencia, A., Hirschman, L.: Linking genes to literature: text mining, information extraction, and retrieval applications for biology. Genome Biol. **9**(2), S8 (2008). https://doi.org/10.1186/gb-2008-9-s2-s8
33. Lahav, D., et al.: A search engine for discovery of scientific challenges and directions (2022). http://arxiv.org/abs/2108.13751, arXiv:2108.13751 [cs]
34. Larsen, P.O., von Ins, M.: The rate of growth in scientific publication and the decline in coverage provided by science citation index. Scientometrics **84**(3), 575–603 (2010). https://doi.org/10.1007/s11192-010-0202-z, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2909426/
35. Li, F., Li, M., Guan, P., Ma, S., Cui, L.: Mapping Publication trends and identifying hot spots of research on internet health information seeking behavior: a quantitative and co-word biclustering analysis. J. Med. Internet Res. **17**(3), e3326 (2015). https://doi.org/10.2196/jmir.3326, https://www.jmir.org/2015/3/e81
36. Li, J., et al.: BioCreative V CDR task corpus: a resource for chemical disease relation extraction. Database **2016**, baw068 (2016). https://doi.org/10.1093/database/baw068
37. Ling, M., Chen, J.: DeepPaperComposer: a simple solution for training data preparation for parsing research papers. In: Proceedings of the First Workshop on Scholarly Document Processing, pp. 91–96. Association for Computational Linguistics, Online (2020). https://doi.org/10.18653/v1/2020.sdp-1.10, https://www.aclweb.org/anthology/2020.sdp-1.10
38. Lippi, M., Torroni, P.: Argumentation mining: state of the art and emerging trends. ACM Trans. Internet Technol. **16**(2), 10:1–10:25 (2016). https://doi.org/10.1145/2850417
39. Luan, Y.: Information extraction from scientific literature for method recommendation (2018). https://doi.org/10.48550/arXiv.1901.00401, http://arxiv.org/abs/1901.00401, arXiv:1901.00401 [cs]
40. Luan, Y., He, L., Ostendorf, M., Hajishirzi, H.: Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction (2018). http://arxiv.org/abs/1808.09602, arXiv:1808.09602 [cs] version: 1
41. Lytos, A., Lagkas, T., Sarigiannidis, P., Bontcheva, K.: The evolution of argumentation mining: From models to social media and emerging tools. Information Processing & Management **56**(6), 102055 (2019). https://doi.org/10.1016/j.ipm.2019.102055, https://www.sciencedirect.com/science/article/pii/S030645731930024X
42. Ma, Y., Dong, M., Zhou, K., Mita, C., Liu, J., Wayne, P.M.: Publication Trends in Acupuncture Research: A 20-Year Bibliometric Analysis Based on PubMed. PLoS ONE **11**(12), e0168123 (2016). https://doi.org/10.1371/journal.pone.0168123, https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0168123
43. Malmasi, S., Fang, A., Fetahu, B., Kar, S., Rokhlenko, O.: SemEval-2022 Task 11: multilingual complex named entity recognition (MultiCoNER). In: Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pp. 1412–1437. Association for Computational Linguistics, Seattle,

United States (2022). https://doi.org/10.18653/v1/2022.semeval-1.196, https://aclanthology.org/2022.semeval-1.196
44. Melgoza, P., Mennel, P.A., Gyeszly, S.D.: Information overload. Collect. Build. **21**(1), 32–43 (2002). https://doi.org/10.1108/01604950210414706
45. Mitchell, A., Strassel, S., Huang, S., Zakhary, R.: ACE 2004 multilingual training corpus (2005). https://doi.org/10.35111/8M4R-V312, https://catalog.ldc.upenn.edu/LDC2005T09, artwork Size: 366008 KB Pages: 366008 KB
46. Mondal, I., Hou, Y., Jochim, C.: End-to-end construction of NLP knowledge graph. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 1885–1895. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.findings-acl.165, https://aclanthology.org/2021.findings-acl.165
47. Nasar, Z., Jaffry, S.W., Malik, M.K.: Information extraction from scientific articles: a survey. Scientometrics **117**(3), 1931–1990 (2018). https://doi.org/10.18653/v1/2021.findings-acl.165
48. QasemiZadeh, B., Schumann, A.K.: The ACL RD-TEC 2.0: a language resource for evaluating term extraction and entity recognition methods. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), pp. 1862–1868. European Language Resources Association (ELRA), Portorož, Slovenia (2016). https://aclanthology.org/L16-1294
49. Reis, H.T., Stiller, J.: Publication trends in JPSP: a three-decade review. Pers. Soc. Psychol. Bull. **18**(4), 465–472 (1992). https://doi.org/10.1177/0146167292184011
50. Riloff, E., Wiebe, J., Phillips, W.: Exploiting subjectivity classification to improve information extraction. In: Proceedings of the 20th National Conference on Artificial Intelligence, vol. 3, pp. 1106–1111. AAAI 2005, AAAI Press, Pittsburgh, Pennsylvania (2005)
51. Roth, D., Yih, W.t.: A Linear programming formulation for global inference in natural language tasks. In: Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004) at HLT-NAACL 2004, pp. 1–8. Association for Computational Linguistics, Boston, Massachusetts, USA (2004). https://aclanthology.org/W04-2401
52. Samuel, A.L.: Some studies in machine learning using the game of checkers. IBM J. Res. Dev. **3**(3), 210–229 (1959). https://doi.org/10.1147/rd.33.0210, conference Name: IBM Journal of Research and Development
53. Sandhaus, E.: The New York Times Annotated Corpus (2008).https://doi.org/10.35111/77BA-9X74, https://catalog.ldc.upenn.edu/LDC2008T19, artwork Size: 3250585 KB Pages: 3250585 KB
54. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
55. Siegel, N., Horvitz, Z., Levin, R., Divvala, S., Farhadi, A.: FigureSeer: parsing result-figures in research papers. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 664–680. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_41
56. Spangler, S., et al.: Automated hypothesis generation based on mining scientific literature. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1877–1886. KDD 2014, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2623330.2623667

57. Stoica, G., Platanios, E.A., Póczos, B.: Re-TACRED: addressing shortcomings of the TACRED dataset (2021). https://doi.org/10.48550/arXiv.2104.08398, http://arxiv.org/abs/2104.08398, arXiv:2104.08398 [cs]
58. Taboureau, O., et al.: ChemProt: a disease chemical biology database. Nucleic Acids Res. **39**(Database issue), D367–372 (2011). https://doi.org/10.1093/nar/gkq906
59. Tan, Q., Xu, L., Bing, L., Ng, H.T., Aljunied, S.M.: Revisiting DocRED – addressing the false negative problem in relation extraction (2023). http://arxiv.org/abs/2205.12696, arXiv:2205.12696 [cs] version: 3
60. Venugopal, V., Pai, S., Olivetti, E.: MatKG: the largest knowledge graph in materials science – entities, relations, and link prediction through graph representation learning (2022). https://doi.org/10.48550/arXiv.2210.17340, http://arxiv.org/abs/2210.17340, arXiv:2210.17340 [cond-mat]
61. Wadden, D., Wennberg, U., Luan, Y., Hajishirzi, H.: Entity, relation, and event extraction with contextualized span representations. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 5784–5789. Association for Computational Linguistics, Hong Kong, China (2019).https://doi.org/10.18653/v1/D19-1585, https://aclanthology.org/D19-1585
62. Walker, C., Strassel, S., Medero, J., Maeda, K.: ACE 2005 multilingual training corpus (2006). https://doi.org/10.35111/MWXC-VH88, https://catalog.ldc.upenn.edu/LDC2006T06, artwork Size: 1572864 KB Pages: 1572864 KB
63. Ware, M., Mabe, M.: The STM report: an overview of scientific and scholarly journal publishing. Copyright, Fair Use, Scholarly Communication, etc. (2015). https://digitalcommons.unl.edu/scholcom/9
64. Wiebe, J., Riloff, E.: Finding mutual benefit between subjectivity analysis and information extraction. IEEE Trans. Affect. Comput. **2**(4), 175–191 (2011). https://doi.org/10.1109/T-AFFC.2011.19. conference Name: IEEE Transactions on Affective Computing
65. Wilson, S.J., Wilkins, A.D., Holt, M.V., Choi, B.K., Konecki, D., Lin, C.H., Koire, A., Chen, Y., Kim, S.Y., Wang, Y., Wastuwidyaningtyas, B.D., Qin, J., Donehower, L.A., Lichtarge, O.: Automated literature mining and hypothesis generation through a network of medical subject headings (2018). https://doi.org/10.1101/403667, https://www.biorxiv.org/content/10.1101/403667v1
66. Yao, Y., et al.: DocRED: a large-scale document-level relation extraction dataset. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pp. 764–777. Association for Computational Linguistics, Florence, Italy (2019). https://doi.org/10.18653/v1/P19-1074, https://aclanthology.org/P19-1074
67. Zhang, Y., Zhong, V., Chen, D., Angeli, G., Manning, C.D.: Position-aware attention and supervised data improve slot filling. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 35–45. Association for Computational Linguistics, Copenhagen, Denmark (2017). https://doi.org/10.18653/v1/D17-1004, https://aclanthology.org/D17-1004

Assessing the Overlap of Science Knowledge Graphs: A Quantitative Analysis

Jenifer Tabita Ciuciu-Kiss(✉) and Daniel Garijo

Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain
jenifer.ciuciu-kiss@alumnos.upm.es, daniel.garijo@upm.es

Abstract. Science Knowledge Graphs (SKGs) have emerged as a means to represent and capture research outputs (papers, datasets, software, etc.) and their relationships in a machine-readable manner. However, different SKGs use different taxonomies, making it challenging to understand their overlaps, gaps and differences. In this paper, we propose a quantitative bottom-up analysis to assess the overlap between two SKGs, based on the type annotations of their instances. We implement our methodology by assessing the category overlap of 100,000 publications present both in OpenAlex and OpenAIRE. As a result, our approach produces an alignment of 71 categories and discusses the level of agreement between both KGs when annotating research artefacts.

Keywords: Scientific Knowledge Graph · Knowledge Graph · Taxonomy · Alignment

1 Introduction

As the volume of scientific literature increases, the need for scalable and efficient systems to navigate this extensive information becomes crucial. Science Knowledge Graphs (SKGs) [11] have emerged as a key tool for representing research entities (publications, people, organizations, datasets, software, etc.) their relationships and metadata in a machine-readable manner.

SKGs such as OpenAIRE[1] [18,19,23,24] and OpenAlex[2] [21] contain millions of entities describing publications and research outputs. One of the main challenges when using SKGs is identifying and resolving overlaps in categorization, which is critical for querying them consistently and reliably. This challenge is complex due to the diversity and volume of data within these KGs, requiring advanced methodologies for effective detection and resolution of overlaps. Understanding these overlaps and disagreements is essential for insights into the structure of scientific knowledge, highlighting patterns that are not immediately apparent due to data scale and diversity.

1 https://www.openaire.eu/.
2 https://openalex.org/.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 171–185, 2024.
https://doi.org/10.1007/978-3-031-65794-8_11

This paper proposes a quantitative bottom-up methodology to assess the overlap of SKGs categories, based on the annotations made on their instances. More specifically we aim to explore the overlap of the taxonomies used in scientific literature [22]. Our contributions include:

1. A novel methodology designed to explore the overlap between SKGs.
2. An implementation of the methodology, based on two SKGs to validate its effectiveness, resulting in 71 new aligned categories within these graphs.
3. An initial exploration study of the intersection of two SKGs, based on 100,000 papers that are jointly described in both of them.

As a proof of concept, we have applied our methodology to a subset of OpenAlex and OpenAIRE SKGs, in the AI domain. We chose OpenAlex for its extensive global database of academic research, and OpenAIRE for its European focus and its integration from heterogeneous data sources. This combination offers a comprehensive view of academic communication, providing a comprehensive dataset for our methodology.

The remainder of the paper is structured as follows. Section 2 describes our methodology, while Sect. 3 explains how we implemented our methodology by assessing OpenAlex and OpenAIRE. Section 4 discusses the results of our categorization analysis on both SKGs, Sect. 5 introduces relevant efforts to map taxonomies and ontologies, and Sect. 6 concludes the paper.

2 A Methodology for Assessing SKG Overlap

We propose a sequential process that evaluates the degree of overlap in SKGs and aims to develop a suite of potential mappings across various KGs, informed by the insights gained from the overlap assessment.

Our methodology is divided into two phases, detailed in Fig. 1. The initial phase (on top of Fig. 1) includes data collection, alignment of the different KG instances and preprocessing. This phase may be repeated and expanded as necessary to refine the dataset to an acceptable size and quality. After completing the dataset preparation, the category alignment phase starts(bottom of Fig. 1). This phase systematically proposes, evaluates, and selects the best mappings between categories based on existing paper annotations, producing a validated set of final mappings of overlapping categories.

2.1 Data Preparation

To date, there is no available open dataset tailored for the quantitative analysis of overlaps within SKGs that considers associated papers and their categorizations. This gap requires the creation of a dataset for conducting a bottom-up quantitative analysis. The data preparation phase may be challenging due to 1) the size of SKGs and 2) the diverse structures and access methods of SKGs, which range from complete data dumps available on platforms like Zenodo [17] to those accessible only via REST APIs or SPARQL [5] queries. We detail the steps for data preparation below.

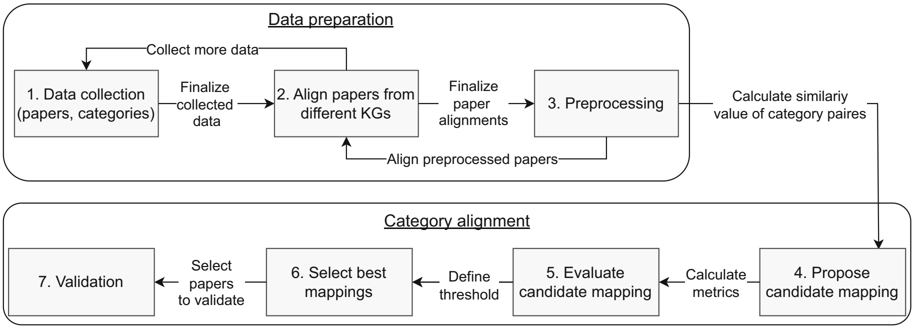

Fig. 1. Steps of the methodology for assessing the overlap of SKGs

Data Collection encloses the aggregation of data from various KGs. Data must contain references to the research papers under analysis and their associated categories. Distinct unique identifiers (typically DOIs) may be used for publications, ensuring that these identifiers are consistent across all targeted KGs for data acquisition. Preliminary examination of the collection structures is compulsory to ensure the integrity and quality of the data obtained.

Align Papers from Different KGs using the gathered data and the chosen unique identifier (e.g. paper DOI, title). Complete alignment may not be feasible due to the heterogeneous nature of data across SKGs, yet a substantial portion of the data should be possible to align, given the overlap in the data sources.

Preprocessing of the gathered data entails multiple steps. First, we eliminate noise from category data, in order to avoid potential variations in character encodings and the presence of inconsistencies in category names and titles, such as inconsistent capitalization and the use of dashes. Cleaning the textual data enhances the alignment quality between the papers and their corresponding categories.

Following text cleansing, we remove underrepresented categories to streamline the later stages of the analysis by reducing its complexity. A category is considered underrepresented when the count of associated papers falls below a predetermined threshold. The value of this threshold is flexible and should consider the size of the dataset, the overall count of categories, and how papers are distributed among these categories.

2.2 Category Alignment

This phase consists of three steps that generate an initial set of candidate mappings between categories:

Propose Candidate Mapping. To identify probable candidate mappings with significant relevance, a similarity model must be used to exclude mappings with semantic similarity below a designated threshold.

Text similarity may be computed using existing embedding techniques. For example, in our work we propose the `en_core_web_md` model in spaCy,[3] which employs GloVe word embeddings [12,20]. The similarity between category strings is determined by the cosine similarity of their vector representations:

$$\text{similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

where **A** and **B** are the vector representations of the category strings. The similarity ranges from -1 (opposite meaning) to 1 (identical meaning).

To maximize the number of candidate mappings, a similarity greater than 0 may be considered.

Evaluate Candidate Mappings using the following metrics:

- Number of papers belonging to the first category ($Support1$)
- Number of papers belonging to the second category ($Support2$)
- Number of papers belonging to both categories ($Intersection$)
- The ratio of $Intersection$ over $Union$ also called as IoU [7,15] ($Agreement$)

The $Agreement$ (i.e., $Intersection/Union$) is calculated using the $Intersection$ over $Union$ (where $Union = Support1 + Support2 - Intersection$) of papers belonging to a certain category in their respective SKG.

Select the Best Mapping based on the *Agreement*. Category mappings are classified into three types: exact, related, and unrelated. Exact matches, where the similarity score equals 1, represent identical categories across all KGs. Related matches exceed the established threshold of *Agreement*, suggesting a strong correspondence. Unrelated categories fall at or below the threshold, indicating a weaker or no relation. The threshold for *Agreement* is adjustable and upon various factors, notably the forthcoming manual validation. Although manual validation of all mappings would be ideal, resource limitations require setting a pragmatic threshold to minimize manual effort.

Validate candidate mappings by having domain experts manually review papers classified into the aligned categories.

3 Initial SKG Overlap Assessment: OpenAIRE and OpenAlex

This work uses two KGs as primary data sources: OpenAIRE (Open Access Infrastructure for Research in Europe [18,19,23,24]) and OpenAlex [21].

[3] https://spacy.io/models/en#en_core_web_md.

OpenAIRE is a European Open Science infrastructure that aims to promote open scholarship and substantially improve the discoverability and reusability of research publications and data. The OpenAIRE KG integrates data from a wide range of research outputs, including publications, datasets, projects, and research organizations, facilitating a more interconnected and comprehensive understanding of European scientific research. The OpenAIRE API[4] allows access to a vast collection of scientific publications, datasets, projects, and funding information. In this work, we used the Search API[5][6] to collect data on scientific publications and their categories, facilitating the quantitative analysis of categorization overlaps in KGs.

OpenAIRE is supported by the European Commission and various European entities. It aggregates data from a multitude of sources to build its comprehensive knowledge graph, including repositories, archives, and journals across Europe. As part of the European Open Science Cloud, OpenAIRE benefits from consistent updates and enhancements, ensuring its relevance and utility in the research community. SCINOBO[7] and other science taxonomies are used to classify the results.

OpenAlex is an open catalogue of the global research system, offering detailed information on academic papers, authors, institutions, etc. The platform indexes millions of research outputs, providing a rich dataset for analysis in various academic fields. The OpenAlex API[8][9] provides access to their extensive dataset. This API enables querying and retrieving detailed information about academic works, supporting a wide range of scholarly analyses. In this work, the OpenAlex API was used to gather information on scientific papers and their categorization.

OpenAlex offers a dynamic dataset with weekly updates, incorporating the latest data from various public and proprietary sources, including academic publishers, preprint servers, institutional repositories, and databases. It aims to index the entirety of the scholarly record, offering an open, comprehensive view of global research output. By ingesting data from various sources, OpenAlex ensures a rich and varied dataset, which includes information on publications, authors, institutions, and citation metrics. Furthermore, OpenAlex aligns its dataset with Wikidata [25] categories. As a successor to the Microsoft Academic Graph [27], OpenAlex aims to provide a comprehensive, open resource for academia.

OpenAlex employs a taxonomy with 65,000 categories, as detailed in its README[10]. Further documentation elaborates on the classification model used

4 https://graph.openaire.eu/docs/apis/home/.
5 https://graph.openaire.eu/docs/apis/search-api/.
6 https://api.openaire.eu/search/publications.
7 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10192702/.
8 https://docs.openalex.org/.
9 https://api.openalex.org/works.
10 https://github.com/ourresearch/openalex-concept-tagging.

by OpenAlex, providing comprehensive details. It is noted that the model attains a precision of 60%, an important consideration during our alignment efforts.

3.1 Paper-Category Schema Representation

Figure 2 represents the schema we used for storing the collected data, providing a structural basis for querying and retrieving information during the analysis. It outlines the associations between scientific papers and their respective categories within the database. Nodes marked as *Paper* are attributed with a *title*, while *Category* nodes encapsulate both the *category name* and the *source* attribute, which identifies whether the category is derived from the *OpenAIRE* or *OpenAlex* SKGs. The relational attribute *belongs to* connects papers to their relevant categories, and a second relational attribute *similar* binds category nodes together, equipped with a *similarity value* to express the level of similarity between category pairs. The Neo4j graph database[11] [26] was chosen store SKG data following our chosen representation.

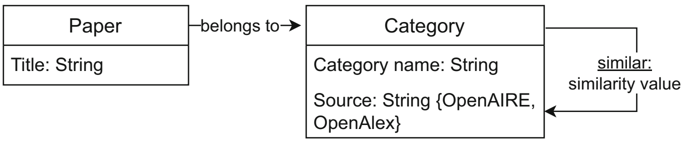

Fig. 2. Paper-category schema representation

3.2 SKG Overlap Analysis: Data Preparation

The alignment of the collected papers from the KGs was conducted using their titles. Whenever multiple papers from the same source shared the same title, leading to potential matching conflicts, such occurrences were disregarded to maintain alignment precision.

A total of 108,555 papers were aligned, that were available in both KGs. On average, OpenAIRE assigns approximately 21 categories to each paper, whereas OpenAlex assigns around 18. Consequently, the dataset from OpenAIRE encompasses a larger number of categories. OpenAlex also assigns a confidence value to each of the assigned categories (not taken into account in this analysis).

The preprocessing primarily targeted the categories to ensure the text was clean for category alignment. The steps included removing Unicode characters, removing punctuation, and converting all text to lowercase. Through experimentation, it was determined that more extensive preprocessing did not significantly improve the results.

[11] https://neo4j.com/.

A threshold was defined for the minimal number of papers a category must be represented by to be included in the analysis. This threshold was set after evaluating the initial distribution of categories in OpenAIRE and OpenAlex and testing various threshold levels. As depicted in Fig. 3, a threshold of 1 had no effect, while a threshold of 5 made a noticeable difference. A threshold of 10 was found to effectively refine the categories without excessively reducing their number. Consequently, we set the threshold at 10, thereby finalizing our dataset for further analysis stages.

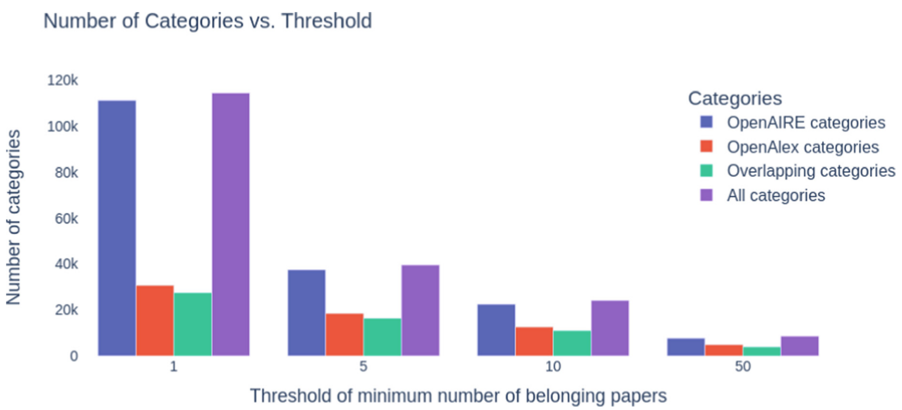

Fig. 3. The number of categories based on the threshold applied to the number of papers representing each category

3.3 SKG Overlap Analysis: Category Alignment

The category alignment phase involves proposing potential mappings between categories based on their similarity, evaluating these mappings against predefined metrics, and then refining the selection based on an agreement threshold to ensure only the most relevant mappings are considered.

While proposing the candidate mappings, a similarity threshold of 0.0 was selected, removing the mappings of opposite categories. This approach was chosen to ensure no potentially significant mappings were excluded at this early stage.

This approach resulted in 509,034 potential mappings. We then assessed these mappings using the metrics outlined in Sect. 2.2. To determine an appropriate threshold for the *Agreement* metric, we evaluated how the number of related matches varied with different threshold settings. As illustrated in Fig. 4, increasing the threshold reduces the count of mappings deemed related, with a notable decrease between 0.1 and 0.2. A plateau appears to occur between 0.4 and 0.5, beyond which the number of related matches dwindles to near zero, especially at a threshold of 0.9. Hence, we established an *Agreement* threshold of 0.5 (without imposing a limit on the 'Similarity' value), which identified 72 mappings as related.

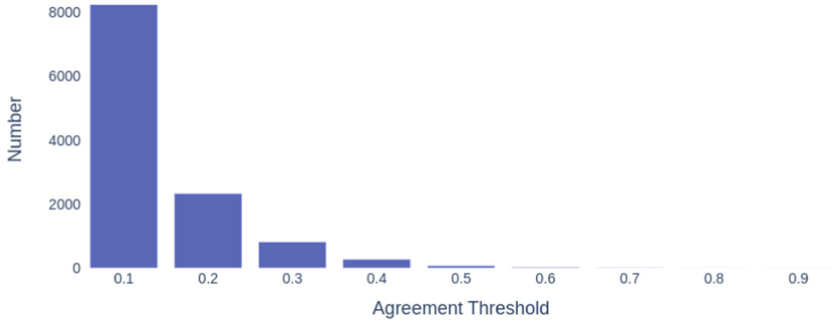

Fig. 4. Number of related matches based on the agreement threshold

We collected the papers that were associated with both of the matched categories for manual review. Following our validation process, the candidate mappings were inspected individually by two researchers, discussing the results until an agreement was reached.

4 Results

Figure 5 illustrates the evolution of the data throughout our analysis, indicating how each step of the methodology impacts data quantity and analysis detail. A corpus of 176,200 papers from OpenAlex was collected, from which 108,555 were found to correspond with the OpenAIRE database entries. Following the paper alignment, an analysis of the categories was conducted. We defined a threshold, requiring a category to contain a minimum of 10 papers for consideration in the mapping process. This criterion resulted in a total of 12,642 categories from OpenAlex and 22,462 categories from OpenAIRE. There was considerable overlap among the categories, leading to the creation of 509,034 potential category mappings. Upon calculating the metrics described in Sect. 2.2 for each mapping, we categorized the mappings into three types: exact, related, and unrelated. Detailed in the bottom right of Fig. 5, under 0.1% of these mappings were considered related (counting 72). Meanwhile, 2.34% were identified as exact matches, signifying categories with a one-to-one correspondence across both KGs. The rest, 97.65% were classified as unrelated matches, which fall outside the relevant domain of this analysis.

In summary, there are 12,642 categories in OpenAlex, with 11,920 identified as exact matches, accounting for 94.23%. These categories also exist in OpenAIRE, directing our focus to matching the remaining 722 OpenAlex categories.

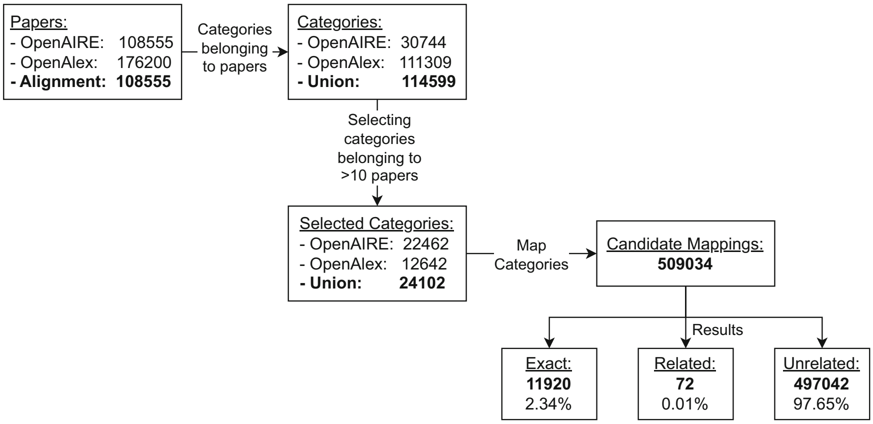

Fig. 5. Data flow from the initial collection to final category mapping analysis results

We found 72 related categories, approximately 10% of the OpenAlex categories requiring matches.

We manually examined these 72 mappings and observed that the labels do not always align (15 mappings). Upon further analysis of the overlapping papers for each SKG, we determined that the mappings remain plausible, although one of the label names may be incorrect. For instance, both 'lasso (statistics)' and 'lasso (programming language)' refer to papers related to lasso statistics. However, the label 'lasso (programming language)' in OpenAlex is used incorrectly for the reviewed papers, which all correspond to lasso statistics. We also identified 1 example of correlation, but not causation between categories: 'melanism' from OpenAlex and 'peppered moth' from OpenAIRE both refer to a collection of papers studying the melanism in a concrete species of moth. Therefore, of the initial 72 mappings we proposed, 14 were identified with misaligned labels referring to the same papers and 1 exhibited correlation without causation. This underscores the importance of our methodology in identifying candidate mappings that require expert validation.

Further, we investigated the relationship between the *Similarity* and the *Agreement* metrics, with findings illustrated in Fig. 6. Interestingly, there appears to be no significant correlation between these two metrics, indicating a high level of disagreement when annotating research publications with concepts.

Another interesting takeaway is the distribution of the *Agreement* values of the exact matches, shown in Fig. 7. Despite the presence of identical categories across both KGs, there is a strong disagreement among the papers that belong to these categories (i.e., the same papers have different category annotations). Additional work is needed to assess if the confidence values assigned in OpenAlex categories affect these findings (e.g. removing low-confidence categories).

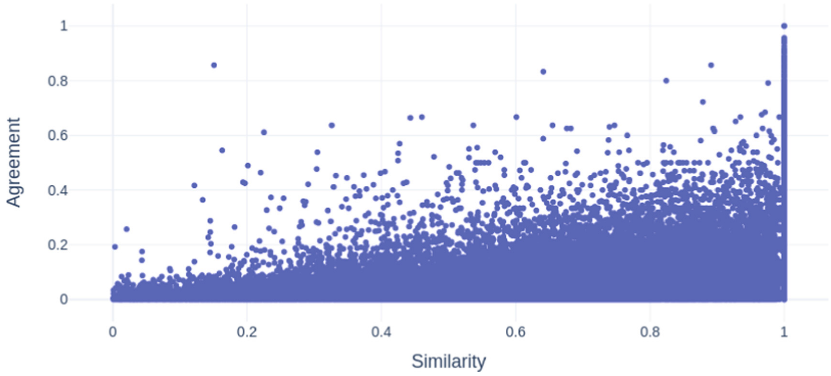

Fig. 6. Correlation of Similarity and Agreement

The scripts [2][12] used to carry out our methodology and analysis are available online under the MIT license. The results of the matches can be found in Zenodo [3].[13]

In summary, our analysis yielded 3 main findings. First, we identified 71 (72 mappings proposed - 1 mismatch: 'melanism' from OpenAlex and 'peppered moth' from OpenAIRE) newly aligned categories across two SKGs. Secondly, we observed a notable lack of correlation between the 'Similarity' and 'Agreement' metrics. Finally, our research revealed that the presence of identical categories in both SKGs does not guarantee agreement on category assignments.

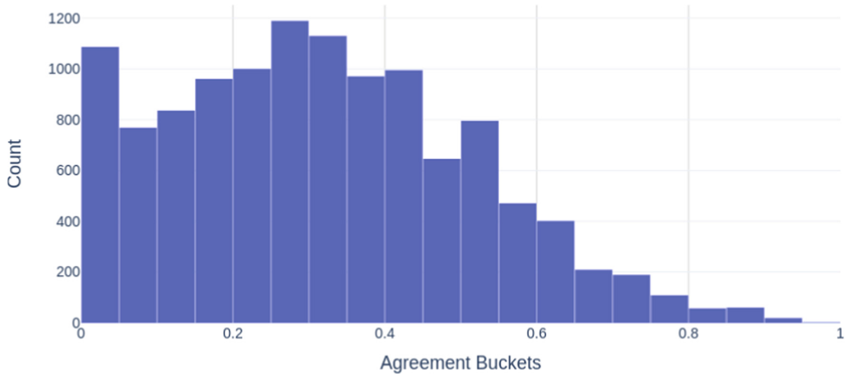

Fig. 7. Distribution of Agreement values of the exact mappings (Similarity = 1)

[12] https://github.com/kuefmz/define_taxonomy.
[13] https://zenodo.org/records/10974512.

5 Related Work

Ontology alignment [6] is a subset of KG alignment and involves matching concepts, relationships, and instances across different ontologies to enable knowledge integration, facilitating a unified view of knowledge across various domains. This section explores significant contributions to the field of KG and ontology alignment.

5.1 KG Alignment Based on Embeddings

Several methods leverage embedding techniques to enhance the interoperability and integration of heterogeneous knowledge bases.

ITransE [29] is an approach for embedding knowledge from various KGs, applicable to cross-lingual KG alignment. The method builds on TransE [1], learning embeddings for entities and relations, and then mapping these embeddings to a shared space using predefined entity alignments. ITransE updates these embeddings through an iterative process as it discovers new entity alignments, requiring uniform relations across all involved KGs for alignment execution.

JE [9] learns embeddings for multiple KGs in a single vector space to align entities. The method employs initial entity alignments to associate two KGs and modifies the TransE model to include an entity alignment loss in its loss function, allowing the alignment process.

In [8] the authors present a KG embedding method for entity alignment, a crucial task for integrating knowledge from various KGs. Their work provides a comprehensive meta-level analysis of popular embedding methods, identifying statistically significant correlations between different embedding methods and meta-features extracted from KGs. This rigorous analysis offers a unique perspective on the effectiveness and efficiency of various embedding methods in real-world KG settings, addressing critical questions about the assumptions and sensitivities of these methods to different KG characteristics.

The publications presented above focus on using embedding methods for aligning KGs, utilizing models to project entities and relations into a unified vector space. These methods often depend on pre-aligned data. In contrast, we introduce a quantitative approach that focuses on a bottom-up analysis to assess overlaps directly based on the categorization of scientific literature within the KGs. However, we base our work on these techniques to calculate similarity between categories.

5.2 KG Alignment Based on Machine Learning

SelfLinKG [16] introduces an approach for enhancing the connectivity and utility of scientific KGs. This work leverages self-supervised learning techniques to identify and establish links between disparate KGs, facilitating a more integrated and comprehensive representation of scientific knowledge. By employing self-supervision, the authors demonstrate significant improvements in the accuracy and efficiency of KG linking, offering valuable insights into the potential of machine learning in KG integration.

Cross-lingual KG alignment [28] presents method for aligning KGs across different languages using graph convolutional networks (GCNs). This work [28] focuses on the challenge of matching entities in multilingual KGs, an essential task for enhancing cross-lingual interoperability and integration of information. Their approach involves training GCNs to embed entities from different languages into a unified vector space, where alignment is determined based on the proximity of entity embeddings. This method leverages both structural and attribute information of entities, aiming to improve the accuracy and efficiency of cross-lingual KG alignment.

In [13], the authors explore the application of KGs and attention mechanisms in bag-level relation extraction, providing a quantitative analysis of their impact. This study contributes a new dataset and proposes a framework to evaluate how KGs and attention mechanisms affect the extraction process, offering insights that could inform the development of more effective relation extraction methods.

All these methods leverage the power of machine learning to identify patterns and establish connections within and across KGs, contributing to a richer, more interconnected web of knowledge. However, they often require substantial training data and can sometimes obscure the interpretability of the alignment process. These methods adapt and evolve through learning patterns in the data, which, while effective, can introduce complexities in understanding why specific alignments are suggested. Our quantitative approach sidesteps these challenges by employing a straightforward, bottom-up analysis that directly assesses the categorizations of papers in KGs. Our method offers a clear, logical pathway to understanding alignments, grounded in the inherent structure and content of the KGs themselves, rather than inferred patterns from machine learning models.

6 Conclusions and Future Work

This work proposed a quantitative bottom-up analysis to assess the overlaps among different KGs, using OpenAIRE and OpenAlex as primary data sources. The findings underscore a notable divergence in the categorization and alignment of KGs despite their reliance on similar resources and methodologies. Surprisingly, even when these KGs draw upon comparable datasets and aim to represent similar domains, the divergence in their categorization frameworks is substantial.

This study successfully proposed a set of mappings that are likely to be related, offering a new perspective on the interconnectedness of these KGs. However, it is imperative to note that the proposed mappings are preliminary and require further validation by domain experts to ensure their accuracy and relevance. This validation is crucial for ensuring the mappings' utility in enhancing the interoperability and integration of KGs in the realm of scientific research.

The future direction of this research involves expanding the scope to complete the analysis of the entire OpenAlex and OpenAIRE KGs and expand to other

SKGs (e.g. ORKG [14], AI-KG [4], Crossref [10]), thereby enriching the dataset and enhancing the robustness of the findings. Furthermore, we plan to enhance our experiments by employing various embeddings to eliminate biases and ascertain the similarity between terms. An important aspect for the future is to consider additional data available in the KGs. OpenAlex provides the confidence values for categories, which we did not incorporate in our work. By integrating more KGs and more data from the KGs, a more comprehensive understanding of the overlaps and divergences across different knowledge domains may be achieved. We intend to broaden our analysis by incorporating an inter-annotator agreement metric, which will serve not only as an additional measure but also as a tool for validation. Furthermore, we plan to enhance our experiments by employing various embeddings to ascertain the similarity between terms.

Moreover, our goal is to delve deeper into AI-related papers, extracting and analyzing their categorizations to propose a refined set of mappings. This effort will involve a systematic collection and analysis of AI research outputs across various KGs, followed by the application of advanced alignment and mapping techniques. The ultimate goal is to construct a more interconnected and semantically rich network of KGs, facilitating a more integrated and accessible repository of scientific knowledge.

Acknowledgements. This work is supported by the Madrid Government (Comunidad de Madrid - Spain) under the Multiannual Agreement with Universidad Politécnica de Madrid in the line Support for R&D projects for Beatriz Galindo researchers, in the context of the VPRICIT, and through the call Research Grants for Young Investigators from Universidad Politécnica de Madrid.

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Burges, C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. (eds.) Advances in Neural Information Processing Systems, vol. 26. Curran Associates, Inc. (2013). https://proceedings.neurips.cc/paper_files/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf
2. Ciuciu-Kiss, J.T.: Scripts used to define taxonomy in the al/ml domain. kuefmz/define_taxonomy: v0.2, April 2024. https://doi.org/10.5281/zenodo.10987999
3. Ciuciu-Kiss, J.T., Garijo, D.: Assessing the overlap of science knowledge graphs: a quantitative analysis - exact and related matches (2024). https://doi.org/10.5281/zenodo.10974512
4. Dessì, D., Osborne, F., Reforgiato Recupero, D., Buscaldi, D., Motta, E., Sack, H.: AI-KG: an automatically generated knowledge graph of artificial intelligence. In: Pan, J.Z., et al. (eds.) ISWC 2020. LNCS, vol. 12507, pp. 127–143. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-62466-8_9
5. DuCharme, B.: Learning SPARQL: Querying and Updating with SPARQL 1.1. O'Reilly Media, Inc., Sebastopol (2013)
6. Euzenat, J., Shvaiko, P.: Ontology Matching. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38721-0

7. Everingham, M.: The pascal visual object classes challenge (2007). http://wwwpascal-network.org/challenges/VOC/voc2007/workshop/index.html (2009)
8. Fanourakis, N., Efthymiou, V., Kotzinos, D., Christophides, V.: Knowledge graph embedding methods for entity alignment: experimental review. Data Min. Knowl. Disc. **37**(5), 2070–2137 (2023)
9. Hao, Y., Zhang, Y., He, S., Liu, K., Zhao, J.: A joint embedding method for entity alignment of knowledge bases. In: Chen, H., Ji, H., Sun, L., Wang, H., Qian, T., Ruan, T. (eds.) CCKS 2016. CCIS, vol. 650, pp. 3–14. Springer, Singapore (2016). https://doi.org/10.1007/978-981-10-3168-7_1
10. Hendricks, G., Tkaczyk, D., Lin, J., Feeney, P.: Crossref: the sustainable source of community-owned scholarly metadata. Quant. Sci. Stud. **1**(1), 414–427 (2020)
11. Hogan, A., et al.: Knowledge graphs. ACM Comput. Surv. (CSUR) **54**(4), 1–37 (2021)
12. Honnibal, M., Montani, I.: spacy 2: natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing (2017). https://spacy.io/
13. Hu, Z., Cao, Y., Huang, L., Chua, T.S.: How knowledge graph and attention help? a quantitative analysis into bag-level relation extraction. arXiv preprint arXiv:2107.12064 (2021)
14. Jaradeh, M.Y., Oelen, A., Prinz, M., Stocker, M., Auer, S.: Open research knowledge graph: a system walkthrough. In: Doucet, A., Isaac, A., Golub, K., Aalberg, T., Jatowt, A. (eds.) TPDL 2019. LNCS, vol. 11799, pp. 348–351. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30760-8_31
15. Li, X., Wang, W., Hu, X., Li, J., Tang, J., Yang, J.: Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11632–11641 (2021)
16. Liu, X., et al.: OAG know: self-supervised learning for linking knowledge graphs. IEEE Trans. Knowl. Data Eng. **35**(2), 1895–1908 (2021)
17. Manghi, P., et al.: OpenAIRE research graph: dumps for research communities and initiatives, June 2022. https://doi.org/10.5281/zenodo.6638478
18. Manghi, P., et al.: Openaire research graph dump (2022)
19. Manghi, P., et al.: The openaire research graph data model. Zenodo (2019)
20. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
21. Priem, J., Piwowar, H., Orr, R.: Openalex: a fully-open index of scholarly works, authors, venues, institutions, and concepts. arXiv preprint arXiv:2205.01833 (2022)
22. Rayleigh, J.W.S.B.: Scientific papers, vol. 1. University Press (1899)
23. Rettberg, N., Schmidt, B.: Openaire-building a collaborative open access infrastructure for European researchers. LIBER Q. J. Assoc. Eur. Res. Libr. **22**(3), 160–175 (2012)
24. Rettberg, N., Schmidt, B.: Openaire: supporting a European open access mandate. College Res. Libr. News **76**(6), 306–310 (2015)
25. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10), 78–85 (2014)
26. Vukotic, A., Watt, N., Abedrabbo, T., Fox, D., Partner, J.: Neo4j in action, vol. 22. Manning Shelter Island (2015)
27. Wang, K., Shen, Z., Huang, C., Wu, C.H., Dong, Y., Kanakia, A.: Microsoft academic graph: when experts are not enough. Quant. Sci. Stud. **1**(1), 396–413 (2020)

28. Wang, Z., Lv, Q., Lan, X., Zhang, Y.: Cross-lingual knowledge graph alignment via graph convolutional networks. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pp. 349–357 (2018)
29. Zhu, H., Xie, R., Liu, Z., Sun, M.: Iterative entity alignment via joint knowledge embeddings. In: IJCAI, vol. 17, pp. 4258–4264 (2017)

Shared Task: FoRC

FoRC@NSLP2024: Overview and Insights from the Field of Research Classification Shared Task

Raia Abu Ahmad(✉), Ekaterina Borisova, and Georg Rehm

Deutsches Forschungszentrum Für Künstliche Intelligenz (DFKI), Berlin, Germany
{raia.abu_ahmad,ekaterina.borisova,georg.rehm}@dfki.de

Abstract. This article provides an overview of the Field of Research Classification (FoRC) shared task conducted as part of the Natural Scientific Language Processing Workshop (NSLP) 2024. The FoRC shared task encompassed two subtasks: the first was a single-label multi-class classification of scholarly papers across a taxonomy of 123 fields, while the second focused on fine-grained multi-label classification within computational linguistics, using a taxonomy of 170 (sub-)topics. The shared task received 13 submissions for the first subtask and two for the second, with teams surpassing baseline performance metrics in both subtasks. The winning team for subtask I employed a multi-modal approach integrating metadata, full-text, and images from publications, achieving a weighted F1 score of 0.75, while the winning team for the second subtask leveraged a weakly supervised X-transformer model enriched with automatically labelled data, achieving a micro F1 score of 0.56 and a macro F1 of 0.43.

Keywords: field of research classification · shared task · scholarly information processing

1 Introduction

In recent decades, the volume of published scientific research has experienced an exponential growth rate, estimated to double approximately every 17 years [6,15]. This surge has prompted the establishment of diverse repositories, databases, knowledge graphs, and digital libraries, encompassing both general and specialised domains, aimed at capturing and organising the ever-expanding scientific knowledge landscape. Notable examples include the Open Research Knowledge Graph (ORKG) [20] and the Semantic Scholar Academic Graph (S2AG) [23], along with domain-specific repositories such as PubMed Central [8] for medical research and ACL Anthology [5] for computational linguistics (CL) and natural language processing (NLP).

Classifying scientific knowledge into Fields of Research (FoR) is a fundamental task for these resources, allowing the development of downstream applications like scientific search engines and recommender systems. However, numerous existing resources face limitations in their classification systems, which can

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 189–204, 2024.
https://doi.org/10.1007/978-3-031-65794-8_12

manifest in the form of a FoR taxonomy that lacks granularity, failing to cover fine-grained hierarchical fields, or in the utilisation of unsupervised methods in the classification model, which do not accurately capture desired labels [11].

Previous efforts of FoR classification have been conducted using machine learning [14], deep learning [12,19], and graph-based approaches [2,7,16,19]. However, a state-of-the-art system that enables the classification into a hierarchical taxonomy using human-curated labels is still lacking. Thus, we conducted the *Field of Research Classification (FoRC)* shared task as part of the Natural Scientific Language Processing Workshop (NSLP) 2024,[1] in which we offered two distinct subtasks:

- **Subtask I**: Single-label multi-class field of research classification of general scholarly articles.
- **Subtask II**: Fine-grained multi-label classification of Computational Linguistics scholarly articles.

Both subtasks aimed to classify scholarly papers in a hierarchical taxonomy of FoR, and participants chose to take part in either one or both subtasks. For subtask I, we constructed a dataset of 59,344 publications with their (meta-)data from existing open-source repositories, mainly the ORKG[2] and arXiv,[3] and used a subset of the existing ORKG research fields taxonomy [2]. On the other hand, for subtask II, we introduced a new human-annotated corpus, FoRC4CL, consisting of 1,500 publications from the ACL Anthology labelled using a novel taxonomy of CL (sub-)topics [1].

Both competitions were run using the Codalab platform [30]. For subtask I[4] we had 35 registrations, 13 of which submitted results. In contrast, for the more challenging subtask II[5] we had 20 registrations, two of which submitted results. The shared tasks had the following schedule:

- Release of training data: January 2, 2024
- Release of testing data: January 10, 2024
- Deadline for system submissions: February 29, 2024
- Paper submission deadline: March 14, 2024
- Notification of acceptance: April 4, 2024

The rest of the paper is structured as follows. Section 2 presents previous work related to FoRC in order to compare the presented systems to current research, and Sect. 3 defines both subtasks along with the used evaluation metrics. In Sect. 4, we introduce the datasets and taxonomies used for both subtasks, delving into their construction methods. Section 5 showcases the results achieved by the participating teams in both subtasks, describing the system architectures when possible. Section 6 discusses those results along with their limitations, and Sect. 7 provides concluding remarks.

1 https://nfdi4ds.github.io/nslp2024/.
2 https://orkg.org.
3 https://arxiv.org.
4 https://codalab.lisn.upsaclay.fr/competitions/16684.
5 https://codalab.lisn.upsaclay.fr/competitions/16712.

2 Related Work

Prior research on FoRC, whether in a general context or within a specific fine-grained domain, has been sporadic and isolated. Different researchers used different datasets, lacking a unified gold standard benchmark and taxonomy for training and evaluating classification systems, which makes it difficult to compare different techniques.

Generally, FoRC systems fall into supervised and unsupervised methods. The former involves systems developed with annotated data, utilising models trained on (meta-)data of scholarly articles with pre-existing, ideally human-curated, information about their respective FoR [21]. While the latter relies on clustering existing (meta-)data using various similarity measures [21].

Some argue that unsupervised classification systems are ideal as they do not rely on manually curated and expensive training data, and can be scalable solutions that handle the vast amount of publications and new FoR [35,36]. However, this approach is insufficient, requiring manual validation due to the tendency of unsupervised algorithms like topic modelling to produce noisy and error-prone results that may not accurately capture the intended labels [11]. For this reason, others prefer a supervised learning approach, working with existing datasets of research publications labelled with FoR based on established taxonomies [12,38,42]. In line with the latter, this shared task employed supervised classification because of its ability to train models on more accurate data.

In terms of supervised techniques, some efforts have proposed jointly learning (meta-)data representations in the same latent space as the FoR taxonomy either by regularising parameters and applying penalties to ensure each FoR is close to its parent nodes [42] or by utilising a contrastive learning approach that generates vector representations encompassing information about the FoR hierarchy along with the text [38]. The former used computer science publications from the Microsoft Academic Graph (MAG) and medical publications from PubMed, while the latter applied their technique to general FoR using the Web of Science (WoS) dataset.

Alternatively, other work utilised Convolutional Neural Networks (CNNs) trained on general FoR data from ScienceMetrix, considering metadata like affiliation, references, abstracts, keywords, and titles [33]. Similarly, Daradkeh et al. [12] also used CNNs by focusing on data science publications, conducting dual classification for both content (i. e., FoR) and methods employed in the publications. The authors incorporated explicit (titles, keywords, and abstracts) and implicit (authors, institutions, and journals) metadata, classifying them into a manually curated flat list of labels.

Another approach used Deep Attentive Neural Networks to classify abstract texts from WoS [22]. The authors also used Long Short-Term Memory cells and Gated Recurrent Units with an attention mechanism to embed abstract texts and classify them into 104 general FoR categories according to the WoS schema. Other work focused on hierarchical text classification, neglecting other metadata and emphasising the incorporation of hierarchical taxonomies into classification models. For instance, Deng et al. [13] developed a model maximising text-label mutual information and label prior matching, using constraints on label representation. Similarly, Chen et al. [9] argued for semantic similarity between text and label representations, introducing a joint embedding loss and a matching learning loss to project them into a shared embedding space.

Finally, addressing the research problem through a graph-based approach, Gialitsis et al. [16] viewed classification as a link prediction problem between publication and FoR nodes in a multi-layered graph. They used data from Crossref, MAG, and ScienceMetrix journal classification, and their taxonomy of labels was derived from the Organisation for Economic Cooperation and Development extended with ScienceMetrix. Other research incorporated knowledge from external knowledge graphs (KGs) to augment the representation of FoR. This was done by linking FoR to entities on DBpedia and concatenating their vector representations with (meta-)data [2,19] or by using research-specific KGs such as the AIDA KG [7].

3 Tasks Description

Both subtasks in the FoRC shared task consist of a document classification problem using data and metadata of research publications to predict the main FoR or (sub-)topic the document addresses. The tasks are described as follows:

– **Subtask I: Multi-class FoRC of general research papers:** Given each publication's available (meta-)data, predict the most probable associated FoR the publication deals with from a pre-defined taxonomy of 123 FoR.
– **Subtask II: Multi-label FoRC of CL research papers:** Given each publication's available (meta-)data, predict all possible associated (sub-)topics that describe the main contributions of the publication from a pre-defined taxonomy of 170 (sub-)topics in CL.

As a single-label multi-class classification problem, subtask I is evaluated based on the metrics of accuracy as well as weighted precision, recall, and F1 scores. On the other hand, subtask II is evaluated based on macro and micro precision, recall, and F1 scores.

4 Shared Task Datasets

4.1 Subtask I

For the first subtask, we use a dataset [2], which was developed based on various open-source resources. The ORKG (CC0 1.0 Universal) and arXiv (CC0 1.0) were the main sources for fetching publications with FoR labels, which was intentional since, for both sources, papers are uploaded manually and FoR are curated from their respective taxonomies. In contrast to other repositories, they do not employ automatic classification systems to label scholarly articles, which aligns with our goal of using only manually curated data in order to bypass duplicating a previous classifier. Additionally, Crossref API [18] (CC BY 4.0), S2AG API[6] (ODC-BY-1.0), and OpenAlex [32] (CC0) were used to fetch abstracts

[6] https://www.semanticscholar.org/product/api.

and validate (meta-)data. All publications in the dataset are categorised using a subset of the ORKG research fields taxonomy.[7]

The ORKG and arXiv datasets were combined, and articles with non-English titles and abstracts were excluded. This process resulted in a dataset comprising 59,344 scholarly articles, each labelled according to a taxonomy of 123 FoR organised into four hierarchical levels and five high-level classes: "Physical Sciences and Mathematics", "Engineering", "Life Sciences", "Social and Behavioral Sciences", and "Arts and Humanities".[8] Metadata fields for each publication consist of *title*, *abstract*, *author(s)*, *DOI*, *URL*, *publication month*, *publication year*, and *publisher*. However, it is important to note that not all instances have all metadata fields available [2]. Table 1 shows a sample of three data instances with partial metadata fields. The dataset exhibits significant imbalances in the distribution of FoR, with the high-level label "Physical Sciences and Mathematics" dominating due to the majority of articles originating from arXiv. Notably, "Physics", "Quantum Physics", and "Astrophysics and Astronomy" are the most prevalent, with 6610, 5209, and 3716 articles, respectively. Conversely, the label "Molecular, cellular, and tissue engineering" is the least frequent, comprising eight articles. The average and median number of articles per field are 482.5 and 175, respectively. Figures 1 and 2 show the distribution among the five high-level labels and the overall 123 labels [2].

To run the task, we shuffled the dataset and created a random split of 70/15/15 for training, validation, and testing. The shared task participants were first given access to the training and validation datasets, which contain labels for each publication. Then, the test dataset was shared separately with no labels attached to it. The dataset is available online.[9]

4.2 Subtask II

The dataset used for subtask II was the FoRC4CL corpus [1], which consists of 1500 CL publications extracted from the ACL Anthology[10] that are manually annotated to indicate each publication's main contribution(s). In order to construct the corpus, we randomly selected English publications from the year range of 2016 to 2022. This was done while keeping in mind the venue distribution in the original full corpus, making bigger venues, such as the main ACL Conference, represented by a proportional amount of publications in the corpus. Overall there are 255 venues represented in the corpus, with an average of six papers per venue. The following metadata is available for each publication: *ACL Anthology ID*, *title*, *abstract*, *author(s)*, *URL* to the full text in PDF, *publisher*, *publication year* and *month*, *proceedings title*, *DOI*, *venue*, and its labels in all three levels of the taxonomy. A sample of the corpus is presented in Table 2,

[7] https://orkg.org/fields.

[8] An interactive view of the taxonomy used for subtask I can be accessed at https://huggingface.co/spaces/rabuahmad/forcI-taxonomy.

[9] https://zenodo.org/records/10777735.

[10] https://github.com/shauryr/ACL-anthology-corpus.

Table 1. Partial sample of three instances from the FoRC subtask I dataset

Title	Author(s)	DOI	Label
belt losses evaluation for a push-belt cvt	['Valerian Croitorescu']	10.5194/bg-10-7035-2013	Mechanical Engineering
petroleum exploration and production: past and present environmental issues in the nigeria's niger delta	['Petters, Sunday W.', 'Ite, Margaret U.', 'Ibok, Udo J.', 'Aniefiok Ite']	10.12691/env-1-4-2	Environmental Sciences
public history and contested heritage: archival memories of the bombing of italy	['Alessandro Pesaro', 'Zeno Gaiaschi', 'Greta Fedele', 'Heather Hughes']	10.5130/phrj.v27i0.7088	Arts and Humanities

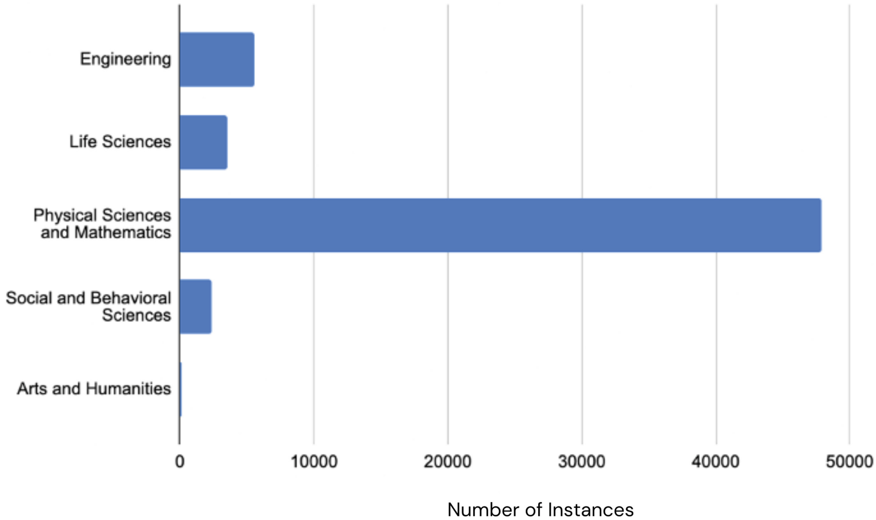

Fig. 1. High-level FoR distribution of subtask I dataset

while the complete dataset is accessible online.[11] The corpus is annotated using Taxonomy4CL [1],[12] a taxonomy developed semi-automatically using a topic modelling approach. The version of the taxonomy used for the corpus consists of 170 topics and subtopics of CL structured in three hierarchical levels.

[11] https://zenodo.org/records/10777674.
[12] https://github.com/DFKI-NLP/Taxonomy4CL.

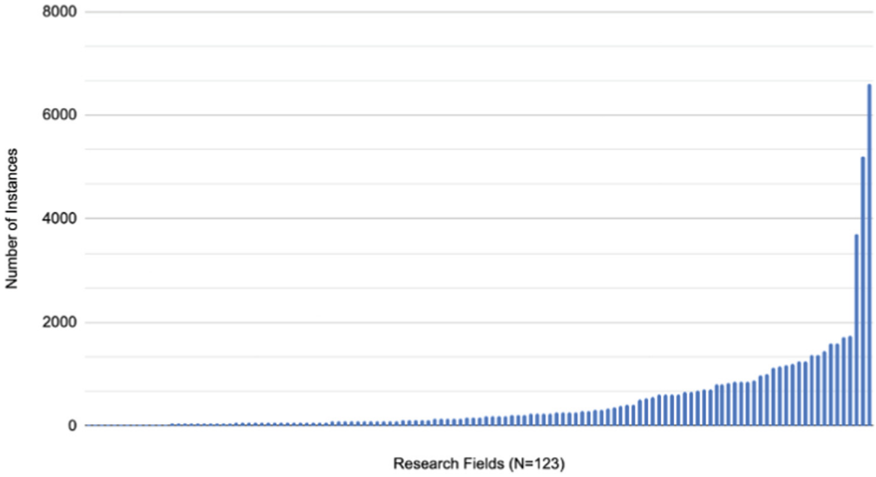

Fig. 2. Overall FoR distribution of subtask I dataset

Similar to subtask I, to run subtask II, we shuffled the corpus and split it randomly into 70/15/15 for training, validation, and testing. Notably, the randomness of the split results in some labels included in the test and/or validation sets but not in the training set. The training and testing datasets were released fully including labels of each hierarchy level, while the testing dataset was later released excluding those labels.

5 Results

5.1 Baselines

As a baseline for subtask I, we fine-tuned SciNCL [29], a model that learns scientific document representations by utilising citation embeddings, and outperforms SciBERT [4] on many tasks. The features fed into the model were the titles and abstracts, and the labels were encoded categorically using LabelEncoder[13] without taking semantic information into account. No regard was given neither to class imbalance nor to the hierarchical representation of labels. The AdamW optimizer was used during training for three epochs with a batch size of 8. We used an RTXA6000 GPU with NVIDIA Turing architecture. This resulted in 0.73 accuracy, 0.73 weighted precision, 0.73 weighted recall, and 0.72 weighted F1 scores.

Similarly, we fine-tuned SciNCL and use it as a baseline for subtask II. We utilised only titles and abstracts as representative features for each publication and combined labels from the three hierarchy levels into one flat list. All taxonomy labels were then multi-hot encoded and fed as input into the model. We

[13] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder.

Table 2. Partial sample of instances from the FoRC4CL dataset used for subtask II

ACL ID	Level 1	Level 2	Level 3
2022.udfestbr-1.5	['Parsing', 'Data Management and Generation', 'Low-resource Languages', 'Domain-specific NLP']	['Data Preparation', 'Syntactic Parsing']	['Dependency Parsing', 'Annotation Processes']
2021.konvens-1.14	['Text Preprocessing', 'Domain-specific NLP', 'Low-resource Languages', 'Classification Applications']	['Hate and Offensive Speech Detection', 'NLP for News and Media']	['NLP for Social Media']

utilised the Google Collab T4 GPU for training the model for three epochs. BCEWithLogits[14] was used as the loss function, AdamW as the optimizer, and all other hyperparameters were the default ones in the AutoModelForSequenceClassification class by Hugging Face.[15] This resulted in micro scores of 0.36 precision, 0.33 recall, and 0.34 F1, and macro scores of 0.01 precision, 0.05 recall, and 0.02 F1.

5.2 Subtask I

We received 13 systems submissions for subtask I, the evaluation results of which are shown in Table 3. The top five teams achieved accuracy, precision, and recall scores higher than the given baseline, while the top six contenders outperformed the F1 score, the last one of which only by a small margin. Although we show all evaluation metrics, we rank the submissions according to their F1 scores, and thus the winning team of the shared task is **SLAMFORC**, followed by **flo.ruo** in second place and **HALE-LAB-NITK** in third. The results of these three teams are very similar and fluctuate for the top three positions in each metric.

Since there was no obligation for each team to submit a description of their system, we provide system descriptions when available, namely for the teams of SLAMFORC [34], HALE-LAB-NITK (private communication), ZB-MED-DSS [39], and NRK [26], all of which are in the top five ranking systems, surpassing the baseline results in all metrics.

Both NRK and ZB-MED-DSS experiment with BERT-based models in a similar manner. NRK build a framework that consists of three different models: SciBERT [4], DeBERTa-V3 [17], and RoBERTa [24]. Each model is fine-tuned

[14] https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html.
[15] https://huggingface.co/docs/transformers/model_doc/auto.

using the provided training dataset of the subtask, utilising a focal loss function to account for data imbalance. The framework is then designed to take all three predictions into account and decide on the final prediction using a hard voting ensemble [27]. The team explains that the combination of all three BERT-based models outperforms the best-performing single model, which is SciBERT in this case.

Similarly, the ZB-MED-DSS team experiment with the following BERT-based models: SciBERT, SciNCL, and SPECTER2 [37]. However, instead of only fine-tuning the models using the available training data, they augment each scholarly article with data from OpenAlex, S2AG, and Crossref. They extract metadata related to (sub-)topics, concepts, keywords, fields of study, and full journal titles. These are then concatenated with the title and abstract of each publication in the available training data and used to fine-tune each of the aforementioned pre-trained BERT-based models. Their best result was achieved by using this combination of raw and augmented data to fine-tune SPECTER2.

The HALE-LAB-NITK team opted to train a support vector machine (SVM) with grid search cross-validation (CV) to find the best-performing hyperparameter combination. This resulted in using a polynomial kernel with the regularisation parameter C set to 1.5. They trained a one vs. rest classifier, meaning that the model was separated into 123 SVMs corresponding to each class in the taxonomy, learning to distinguish the specific class from all the others.

Finally, the SLAMFORC team proposed a multi-modal approach in which they combine (meta-)data from the training dataset, i. e., title, abstract, and publisher, with enriched semantic information from Crossref. The enriched data included subjects mentioned in the article as well as missing DOIs and URLs to the full text. The (meta-)data from the original training dataset was embedded using SciNCL, while the full text of each scholarly article was embedded using both SciNCL and SciBERT with a sliding window of 512 tokens and an overlap of 128 tokens in order to account for the token limitation in these models. Adopting a multi-modal approach, the SLAMFORC team also took advantage of any images found in the PDF of the full text, extracting those using PaperMage [25]. These images were converted to raster graphics and embedded using OpenCLIP [10] and DINOv2 [28]. All three embeddings for each article (i. e., data and metadata, full-text, and images) were concatenated and used to train five different models: SVM, random forest, logistic regression, extreme gradient boosting, and a multi-layer-perceptron. Additionally, SciNCL was fine-tuned using the original (meta-)data. The six predictions from the five mentioned models and SciNCL were then incorporated into a hard-voting ensemble to decide on the final prediction.

Table 3. Evaluation results of subordination for subtask I; top result in bold, runner-up underlined, third place italicised

Rank	Team	Accuracy	Precision	Recall	F1
–	Baseline	0.733	0.731	0.733	0.723
1	SLAMFORC [34]	<u>0.7558</u>	**0.7566**	<u>0.7558</u>	**0.7540**
2	flo.ruo	*0.7542*	<u>0.7545</u>	*0.7542*	<u>0.7524</u>
3	HALE-LAB-NITK	**0.7572**	*0.7536*	**0.7572**	*0.7500*
4	ZB-MED-DSS [39]	0.7476	0.7438	0.7476	0.7426
5	NRK [26]	0.7433	0.7423	0.7433	0.7391
6	Sailor Moon	0.7302	0.7247	0.7302	0.7243
7	pranjalks	0.7260	0.7194	0.7260	0.7202
8	Sallu	0.7059	0.7027	0.7059	0.6930
9	Shaad	0.7023	0.6951	0.7023	0.6915
10	CAU&ZBW	0.6815	0.6792	0.6815	0.6779
11	PhD_CV	0.6581	0.6594	0.6581	0.6528
12	Elixir	0.0584	0.0614	0.0584	0.0572
13	dingdong	0.0037	0.0015	0.0037	0.0019

5.3 Subtask II

As a more complex task, subtask II received two system submissions, both of which outperformed the given baseline in all metrics. Full evaluation results are shown in Table 4. The winning team of this subtask is **CAU&ZBW**, who outperform their runner-up, **CUFE**, on all evaluation metrics. Since we only received a system description from CAU&ZBW [3], we proceed to describe the system they developed.

The challenging aspects of this task lie in its relatively high number of labels (170), its hierarchical nature, its multi-label characteristic, and its small corpus consisting of 1500 overall instances with only 1050 articles available in the training data. For these reasons, the CAU&ZBW team treats this challenge as an extreme multi-label classification (XMLC) task. The team thus experiments with several models, specifically a tf-idf model, Parabel [31], and X-transformer [40]. To represent each scholarly article in the dataset, the CAU&ZBW team uses the title, abstract, venue, publisher, and book title (meta-)data fields from the available training dataset. In addition, they extract the full-text from the given URL of each publication.

However, since the labelled training data is not sufficient for training a model with satisfactory results, CAU&ZBW enrich the dataset with 70,000 unlabelled publications from the ACL Anthology. Then, they use their trained tf-idf model to generate weak labels for each of those publications, giving those as input to fine-tune a weakly supervised X-transformer model. Finally, the team adds the hierarchy of the taxonomy to the final stage of the model, accepting predictions in levels 2 and 3 only if their parent node is already predicted in the previous level. This model achieved their best result, which was the team's final submission.

Table 4. Evaluation results of submissions for subtask II; top result is bolded and runner-up is underlined

Rank	Team	Precision (micro)	Recall (micro)	F1 (micro)	Precision (macro)	Recall (macro)	F1 (macro)
–	Baseline	0.3556	0.3277	0.3411	0.0163	0.0459	0.0239
1	CAU&ZBW [3]	**0.4391**	**0.7591**	**0.5563**	**0.3942**	**0.5551**	**0.4344**
2	CUFE	0.4015	0.3707	0.3855	0.1043	0.0666	0.0592

6 Discussion

As the two approaches that utilise BERT-based models in subtask I, we see that ZB-MED-DSS and NRK produced similar results, with the former slightly outperforming the latter on all metrics. This can be attributed to two main reasons, the first of which is the exclusive use of science-specific BERT models by ZB-MED-DSS as opposed to NRK, which has proven to be more effective when dealing with scientific data [4]. The second reason is the enrichment process applied by the ZB-MED-DSS team, in which they added information from several open-access resources that directly relate to the FoR of each publication.

The model proposed by the HALE-LAB-NITK team is one of the top-scoring ones, yielding the top results in terms of accuracy and weighted recall scores. This means that one vs. rest SVMs with grid search CV outperform fine-tuning BERT-based models (i. e., the ZB-MED-DSS and NRK teams), despite the latter's inherent capability for language understanding. These results suggest that carefully engineered features, combined with hyperparameter tuning, effectively capture domain-specific linguistic patterns crucial for classifying FoR. Additionally, the decision boundaries created by SVMs seem to align well with the separability of different FoR in the feature space, while their computational efficiency and interpretability provide practical advantages. This highlights the importance of considering dataset characteristics, feature representation, hyperparameter tuning, and the potential for hybrid approaches when designing models for tasks requiring advanced language understanding capabilities, rather than fine-tuning pre-trained language models.

The best approach in subtask I was by SLAMFORC, using as much information from scholarly articles as possible. This includes (meta-)data such as title, abstract, publisher, and the full text of the publication along with its images. This is an interesting approach that, to the best of our knowledge, has not been applied to a FoRC task before. The results of this shared task clearly show that there is a high potential for such multi-modal models, seeing as it competes highly with the other text-based models in the task on all evaluation metrics. In the future, it would be interesting to explore the types of images and perhaps also tables used in scholarly publications and how they can help predict the FoR they pertain to.

In terms of subtask II, we see that applying methods used for XMLC tasks did indeed yield good results and thus seem to be appropriate for this task. The problem of insufficient training data was solved by the CAU&ZBW team by introducing noisy data that was automatically labelled. However, the evaluation results exhibit notable disparities across metrics, with micro metrics reflecting relatively strong classification on individual instances but macro metrics indicating variability in class prediction consistency, a problem expected when it comes to XMLC. The model's reliance on a weakly supervised dataset suggests a capacity to learn from noisy or incomplete labels, but also poses challenges in interpreting classification decisions. Future directions might involve refining weakly supervised learning techniques and exploring alternative model architectures.

Importantly, we note that none of the teams in either subtask incorporated the hierarchical relations of labels into training their models, and did not include any other semantic representation pertaining to the labels in their training processes. This can definitely be explored further in future research by incorporating techniques from work on hierarchical text classification [9,13,41,42].

Finally, as organisers of this task, we note that most teams participating in subtask I struggled with two main problems. The first is the class imbalance of the dataset that was outlined more clearly in Sect. 4, which resulted from the lack of human-annotated publications in fields such as Social and Behavioural Sciences and Arts and Humanities. Future endeavours could focus on these underrepresented fields and construct databases of human-annotated publications that can be added to the dataset. Additionally, teams were challenged by the incompleteness of the dataset in specific (meta-)data fields such as publisher and DOI, which made some of them extract additional data from external resources. In terms of subtask II, the main challenge was insufficient training data. In the future, we aim for the FoRC4CL corpus to be expanded by asking authors to annotate their own papers, which should be helpful in training more accurate classification systems [1].

7 Conclusion

In this article, we presented an overview of the *Field of Research Classification (FoRC)* shared task, which was held under the umbrella of the *Natural Scientific Language Processing Workshop (NSLP) 2024*. The FoRC shared task consisted of two subtasks, the first being a single-label multi-class classification of general scholarly papers from 123 hierarchical fields, and the second a more fine-grained multi-label classification of a specific field into a taxonomy 170 (sub-)topics, taking Computational Linguistics as a use-case. The task attracted 13 submissions for subtask I and two submissions for subtask II, both of which included teams succeeding in outperforming the given baselines. The winning team of the first subtask introduced a multi-modal approach combining (meta-)data, full-text, and images from publications, followed by training six different models and a final voting ensemble. While other top teams explored techniques of one vs. rest SVM classifier with grid search and fine-tuning different BERT-based models with data enrichment from external resources. In terms of the second subtask, the winning team utilised a weakly supervised X-transformer model while adding automatically labelled data in order to increase instances for training. Our datasets for both subtasks are publicly available and we aim for them to be used in the future by researchers developing new classification systems. Further improvements can look into incorporating the hierarchical nature of labels in both datasets in the training of the models and making use of the semantic information of the labels for classification. Future iterations of this shared task can increase the number of available training data, especially for subtask II, and incorporate an evaluation metric that takes the hierarchy of the labels into account.

Acknowledgement. This work was supported by the consortium NFDI for Data Science and Artificial Intelligence (NFDI4DS) https://www.nfdi4datascience.de as part of the non-profit association National Research Data Infrastructure (NFDI e. V.). The consortium is funded by the Federal Republic of Germany and its states through the German Research Foundation (DFG) project NFDI4DS (no. 460234259).

References

1. Abu Ahmad, R., Borisova, E., Rehm, G.: FoRC4CL: a fine-grained field of research classification and annotated dataset of NLP articles. In: Proceedings of the Joint International Conference on Computational Linguistics, Language Resources and Evaluation (2024)
2. Abu Ahmad, R., Rehm, G.: Knowledge injection for field of research classification and scholarly information processing. In: Proceedings of the 9th International Symposium on Language and Knowledge Engineering. Dublin, Ireland (2024), 4-6 June. Accepted
3. Bashyam, L.R., Krestel, R.: Advancing automatic subject indexing: combining weak supervision with extreme multi-label classification. In: Rehm, G., Schimmler, S., Dietze, S., Krüger, F. (eds.) Proceedings of the 1st International Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024). Hersonissos, Crete, Greece (2024). 27 May. Accepted
4. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 3615–3620. Association for Computational Linguistics, Hong Kong, China (2019). https://doi.org/10.18653/v1/D19-1371
5. Bird, S., et al.: The ACL anthology reference corpus: a reference dataset for bibliographic research in computational linguistics. In: LREC (2008)
6. Bornmann, L., Haunschild, R., Mutz, R.: Growth rates of modern science: a latent piecewise growth curve approach to model publication numbers from established and new literature databases. Humanit. Soc. Sci. Commun. **8**(1), 1–15 (2021)

7. Cadeddu, A., et al.: Enhancing scholarly understanding: a comparison of knowledge injection strategies in large language models. In: CEUR Deep Learning for Knowledge Graphs Workshop Proceedings (2023). https://ceur-ws.org/Vol-3559/paper-7.pdf
8. Canese, K., Weis, S.: PubMed: the bibliographic database. NCBI Handb. **2**(1), 11–19 (2013)
9. Chen, H., Ma, Q., Lin, Z., Yan, J.: Hierarchy-aware label semantics matching network for hierarchical text classification. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pp. 4370–4379. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.acl-long.337
10. Cherti, M., et al.: Reproducible scaling laws for contrastive language-image learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2818–2829 (2023)
11. Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A.: Supervised learning of universal sentence representations from natural language inference data. In: Palmer, M., Hwa, R., Riedel, S. (eds.) Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 670–680. Association for Computational Linguistics, Copenhagen, Denmark, September 2017. https://doi.org/10.18653/v1/D17-1070
12. Daradkeh, M., Abualigah, L., Atalla, S., Mansoor, W.: Scientometric analysis and classification of research using convolutional neural networks: a case study in data science and analytics. Electronics **11**(13), 2066 (2022)
13. Deng, Z., Peng, H., He, D., Li, J., Yu, P.: HTCInfoMax: a global model for hierarchical text classification via information maximization. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 3259–3265. Association for Computational Linguistics (2021). https://doi.org/10.18653/v1/2021.naacl-main.260
14. Eykens, J., Guns, R., Engels, T.C.: Fine-grained classification of social science journal articles using textual data: a comparison of supervised machine learning approaches. Quant. Sci. Stud. **2**(1), 89–110 (2021)
15. Fortunato, S., et al.: Science of science. Science **359**(6379), eaao0185 (2018)
16. Gialitsis, N., Kotitsas, S., Papageorgiou, H.: SciNoBo: a hierarchical multi-label classifier of scientific publications. In: Companion Proceedings of the Web Conference 2022, pp. 800–809 (2022)
17. He, P., Gao, J., Chen, W.: DeBERTaV3: improving DeBERTa using electra-style pre-training with gradient-disentangled embedding sharing. arXiv preprint arXiv:2111.09543 (2021)
18. Hendricks, G., Tkaczyk, D., Lin, J., Feeney, P.: Crossref: the sustainable source of community-owned scholarly metadata. Quant. Sci. Stud. **1**(1), 414–427 (2020)
19. Hoppe, F., Dessì, D., Sack, H.: Deep learning meets knowledge graphs for scholarly data classification. In: Companion Proceedings of the Web Conference 2021, pp. 417–421 (2021)
20. Jaradeh, M.Y., et al.: Open research knowledge graph: next generation infrastructure for semantic scholarly knowledge. In: Proceedings of the 10th International Conference on Knowledge Capture, pp. 243–246 (2019)
21. Jo, T.: Machine learning foundations. Supervised, Unsupervised, and Advanced Learning. Springer, Cham (2021)

22. Kandimalla, B., Rohatgi, S., Wu, J., Giles, C.L.: Large scale subject category classification of scholarly papers with deep attentive neural networks. Front. Res. Metrics Anal. **5**, 600382 (2021)
23. Kinney, R., et al.: The semantic scholar open data platform. arXiv preprint arXiv:2301.10140 (2023)
24. Liu, Y., et al.: RoBERTa: a robustly optimized BERT pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
25. Lo, K., et al.: PaperMage: a unified toolkit for processing, representing, and manipulating visually-rich scientific documents. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 495–507 (2023)
26. Nguyen, T.K., Dang, V.T.: NRK at FoRC 2024 subtask I: Exploiting BERT-based models for multi-class classification of scholarly papers. In: Rehm, G., Schimmler, S., Dietze, S., Krüger, F. (eds.) Proceedings of the 1st International Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024). Hersonissos, Crete, Greece (2024). 27 May. Accepted
27. Opitz, D., Maclin, R.: Popular ensemble methods: an empirical study. J. Artif. Intell. Res. **11**, 169–198 (1999)
28. Oquab, M., et al.: Dinov2: learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
29. Ostendorff, M., Rethmeier, N., Augenstein, I., Gipp, B., Rehm, G.: Neighborhood contrastive learning for scientific document representations with citation embeddings. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 11670–11688 (2022)
30. Pavao, A., et al.: CodaLab competitions: an open source platform to organize scientific challenges. J. Mach. Learn. Res. **24**(198), 1–6 (2023). http://jmlr.org/papers/v24/21-1436.html
31. Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., Varma, M.: Parabel: partitioned label trees for extreme classification with application to dynamic search advertising. In: Proceedings of the 2018 World Wide Web Conference, pp. 993–1002 (2018)
32. Priem, J., Piwowar, H., Orr, R.: OpenAlex: a fully-open index of scholarly works, authors, venues, institutions, and concepts. arXiv preprint arXiv:2205.01833 (2022)
33. Rivest, M., Vignola-Gagné, E., Archambault, É.: Article-level classification of scientific publications: a comparison of deep learning, direct citation and bibliographic coupling. PLoS ONE **16**(5), e0251493 (2021)
34. Ruosch, F., Vasu, R., Wang, R., Rossetto, L., Bernstein, A.: Single-label multimodal field of research classification. In: Rehm, G., Schimmler, S., Dietze, S., Krüger, F. (eds.) Proceedings of the 1st International Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024). Hersonissos, Crete, Greece (2024). 27 May. Accepted
35. Salatino, A., Osborne, F., Motta, E.: CSO classifier 3.0: a scalable unsupervised method for classifying documents in terms of research topics. Int. J. Digit. Libr. **23**(1), 91–110 (2021). https://doi.org/10.1007/s00799-021-00305-y
36. Shen, Z., Ma, H., Wang, K.: A web-scale system for scientific knowledge exploration. In: Liu, F., Solorio, T. (eds.) Proceedings of ACL 2018, System Demonstrations. pp. 87–92. Association for Computational Linguistics, Melbourne, Australia (2018). https://doi.org/10.18653/v1/P18-4015
37. Singh, A., D'Arcy, M., Cohan, A., Downey, D., Feldman, S.: SciRepEval: a multi-format benchmark for scientific document representations. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 5548–5566 (2023)

38. Wang, Z., Wang, P., Huang, L., Sun, X., Wang, H.: Incorporating hierarchy into text encoder: a contrastive learning approach for hierarchical text classification. In: Muresan, S., Nakov, P., Villavicencio, A. (eds.) Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 7109–7119. Association for Computational Linguistics, Dublin, Ireland (2022). https://doi.org/10.18653/v1/2022.acl-long.491
39. Wolff, B., Seidlmayer, E., Förstner, K.: Enriched BERT embeddings for scholarly publication classification - insights from the NSLP 2024 FoRC shared task I. In: Rehm, G., Schimmler, S., Dietze, S., Krüger, F. (eds.) Proceedings of the 1st International Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024). Hersonissos, Crete, Greece (2024). 27 May. Accepted
40. Zhang, J., Chang, W.C., Yu, H.F., Dhillon, I.: Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. Adv. Neural. Inf. Process. Syst. **34**, 7267–7280 (2021)
41. Zhang, X., Xu, J., Soh, C., Chen, L.: LA-HCN: label-based attention for hierarchical multi-label text classification neural network. Expert Syst. Appl. **187**, 115922 (2022)
42. Zhang, Y., Shen, Z., Dong, Y., Wang, K., Han, J.: MATCH: metadata-aware text classification in a large hierarchy. In: Proceedings of the Web Conference 2021, pp. 3246–3257 (2021)

NRK at FoRC 2024 Subtask I: Exploiting BERT-Based Models for Multi-class Classification of Scholarly Papers

Kiet Nguyen Tuan[1,2] and Thin Dang Van[1,2(✉)]

[1] University of Information Technology – VNUHCM, Ho Chi Minh City, Vietnam
21521042@gm.uit.edu.vn, thindv@uit.edu.vn
[2] Vietnam National University, Ho Chi Minh City, Vietnam

Abstract. This study presents the system developed by team NRK for SubTask I of the Field of Research Classification (FoRC) - NLSP 2024. The task focuses on single-label, multi-class classification of general scholarly papers. Our approach exploits the capabilities of various pre-trained BERTopology models, combined with a straightforward ensemble voting scheme to enhance classification performance. The proposed system achieved competitive results, ranking within the Top 5 on the final scoreboard.

Keywords: BERT-based models · Paper Classification · FoRC · voting ensemble · FoRC shared task

1 Introduction

The automated classification of scientific articles into their respective fields of research (FoR) plays a crucial role in various Natural Language Processing (NLP) applications. It facilitates information retrieval, knowledge organization, and facilitates scholarly search engines. While existing repositories often utilize FoR classification systems, these systems face limitations in terms of the employed taxonomy and the underlying classification model.

This paper addresses these limitations by exploring the application of BERT-based [3] models for single-label, multi-class FoR classification (FoRC) of general scholarly papers. We focus on Subtask I of the Field of Research Classification (FoRC) shared task at NLSP 2024, which aims to develop classifiers that accurately assign one of 123 predefined hierarchical classes from the ORKG taxonomy to general scholarly papers based on their available metadata (title, authors, abstract, etc.).

The contribution of this paper is presented below:

– Investigating the effectiveness of pre-trained BERT-based models for single-label, multi-class FoRC: This study examines the potential of BERT-based models in capturing the semantic relationship within scientific articles and their corresponding research fields.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 205–213, 2024.
https://doi.org/10.1007/978-3-031-65794-8_13

– Contributing to the understanding of automated FOR classification: The findings of this research can provide valuable insights into the effectiveness of pre-trained language models for automated FoRC tasks.

2 Related Work

Automated FoRC has been an active area of research within the Natural Language Processing (NLP) domain. This section explores existing work relevant to Subtask I of the FoRC shared task, focusing on single-label, multi-class classification of general scholarly papers.

Previous works often employed traditional machine learning models like Support Vector Machines (SVMs), Naïve Bayes, K-Nearest Neighbor and Decision Tree introduced in [2] and [6] were employed to classify scientific publications into three categories: Science, Business, and Social Science. These approaches relied on manually extracted features, often utilizing the TF-IDF method to represent the textual content.

To address the challenge of multi-label research paper classification in the face of growing publication volume, [9] propose a joint embedding approach. They utilize separate models for title and abstract processing: a Transformer-based model for title embedding and a combination of GloVe word vectors with a GRU network for abstract embedding. The final joint representation is obtained through a two-tower structure, and their method outperforms baseline models on the CiteULike dataset [8].

In [11], the authors propose a BERT-based graph convolutional neural network (BERT-GCN) model for scientific paper classification. This model leverages the strengths of both BERT and GCNs: BERT, fine-tuned with span masking, learning rate attenuation, and data augmentation, captures the semantic content of paper titles, while the GCN captures the relationships between words within the titles. By combining these elements, BERT-GCN aims to achieve superior classification performance compared to traditional methods.

While prior research has explored various approaches for FoRC, this study specifically focuses on the application of pre-trained BERT models for single-label, multi-class classification of general scholarly papers. We investigate the effectiveness of different BERT models and fine-tuning strategies in the context of Subtask I of the FoRC shared task, contributing to the understanding of their suitability for this specific task and domain. We also compared the performance of our BERT-based model with other techniques, including traditional machine learning approaches and deep learning models, further contributing to the understanding of effective methods for this task.

3 Approach

We employ multiple pre-trained BERT models from the Hugging Face Transformers library suitable for text classification tasks. These models are fine-tuned on the provided dataset, which allows them to adapt to the specific domain of

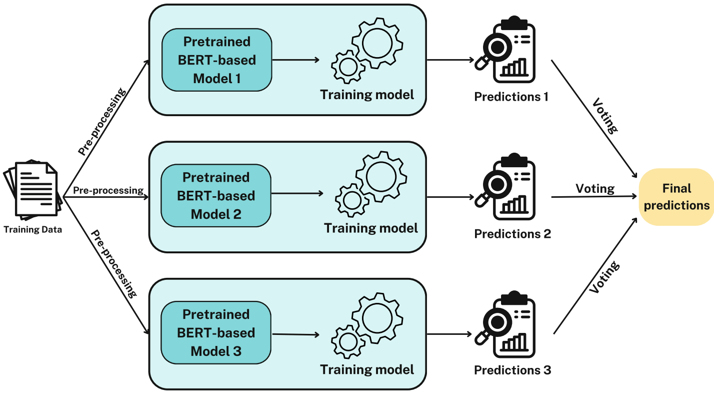

Fig. 1. The overall framework for our system.

scientific papers and the classification task at hand. As can be seen in Figure 1, we utilize the power of pre-trained contextual language models, including the SciBERT [1], DeBERTa-V3 [4] and RoBERTa [7].

- **SciBERT**: a BERT [3] model trained on a massive dataset of scientific text, including research papers, scientific articles, and other relevant materials collected from Semantic Scholar. This allows it to understand the nuances of scientific language, such as specific terminology, jargon, and sentence structures.
- **DeBERTa-V3**: a DeBERTa [5] version improved the efficiency of original DeBERTa using ELECTRA-Style pre-training with Gradient Disentangled Embedding Sharing [4].
- **RoBERTa**: a pre-trained language model builds upon BERT [3] by addressing its limitations with dynamic masking, removing the potentially harmful NSP objective, and using a full-sentence representation.

Voting Scheme: Our motivation for applying an ensemble approach is to take advantage of the performances of various models. Given predictions $\{\hat{y}_{\theta_1}, \hat{y}_{\theta_2}, .., \hat{y}_{\theta_n}\}$ the n base classifiers. We applied the hard voting technique to merge the predictions of the base models. In hard voting, the class predicted by the majority of models becomes the final predicted class for the given paper.

4 Experimental Setup

4.1 Data

Data: We utilized the official training set for training models. The development set was used to choose which model will be included in the voting scheme.

Pre-processing: We apply different pre-processing steps as below to improve the performance for our system:

- **Cleaning and normalization**: Extensive cleaning and normalization procedures common in more general text datasets (e.g., removing punctuation, converting to lowercase) is likely not required for scientific text data. Maintaining the original text format can be beneficial for preserving domain-specific terminology and nuances within the scientific vocabulary.

4.2 Configuration Settings

The system was implemented using the Trainer API from the Hugging Face library [10] for streamlined training and evaluation of the fine-tuned BERT models. This API simplifies the process by handling data loading, model training, and metric computation. We adopted a sequence classification approach, setting a maximum input length of 512 tokens to accommodate potential variations in paper titles and abstracts. The training process consisted of 5 epochs, utilizing a batch size of 16 papers per batch. To optimize training, we employed the AdamW optimizer, commonly used for deep learning models, and incorporated a linear schedule warm-up technique to gradually increase the learning rate during the initial phase. Notably, we also addressed potential class imbalance within the dataset by employing the Focal loss function with the formula as follows:

$$\textbf{Focal Loss}(p_t) = -\alpha(1 - p_t)^{\gamma} \log(p_t), \tag{1}$$

where:

p_t = Model's predicted probability for the true class.
α = Balancing factor controls the focus of the loss function towards easy or hard examples.
γ = Focusing parameter controls the degree of modulating the loss for well-classified examples.

5 Results and Discussion

5.1 Baseline Performance

Naive Bayes: The analysis of Naive Bayes reveals poor performance compared to other models on this task. Its performance falls significantly behind all models. This suggests that Naive Bayes might not be well-suited for this specific classification problem due to several limitations.

Support Vector Machine: While still falling short of the BERT-based models, SVM demonstrates a better performance compared to Naive Bayes and BiLSTM and even achieved a competitive result compared to deBERTa-v3-small - a transformer-based model. This suggests that the SVM model is able to capture well features extracted by applying TF-IDF.

We trained both a simple BiLSTM model and a CNN model, leveraging pre-trained word embeddings 'word2vec-google-news-300' provided by [1]Gensim,

[1] https://radimrehurek.com/gensim/models/word2vec.html.

to achieve moderate performance for both. This performance falls behind the traditional machine learning models (SVM) and the pre-trained BERT models. While pre-trained word embeddings provided some improvement in capturing semantic relationships, the limitations remain:

- **Limited training data:** The available dataset might not have been sufficient for **either the BiLSTM or CNN** to learn the complex relationships needed for accurate classification, even with the additional information from pre-trained embeddings. This limitation likely explains the observed performance gap between these models and the SVM classifier. In tasks with restricted training data, the ability of deep learning models to exploit their full potential compared to traditional machine learning models can be hindered.
- **Lack of pre-trained knowledge:** Compared to BERT, **both BiLSTM and CNN** still lack pre-existing knowledge of language structures and relationships beyond the information captured in the pre-trained word embeddings. Learning these intricacies from scratch remains challenging with limited data.

5.2 Results

The performance of the participant's system is reported by the metrics which are: weighted Precision, Recall, and F1 Score, Accuracy. For individual models, the SciBERT-uncased achieved the highest performance among the individual models, followed by SciBERT-cased, RoBERTa-base, and deBERTa-v3-small. This could be attributed to the specific pre-training data and objectives of each model. SciBERT, being pre-trained on scientific text, might have a better understanding of the domain-specific language used in the papers, leading to its superior performance.

The ensemble model, combining the predictions of SciBERT-uncased, SciBERT-cased, and RoBERTa-base, outperforms all individual models, demonstrating the effectiveness of combining diverse predictions with an Accuracy of 0.7433, Precision of 0.7423, Recall of 0.7433, and F1 score of 0.7391. Among the individual models, SciBERT-uncased performs best, followed by SciBERT-cased, RoBERTa-base, and deBERTa-v3-small. The deBERTa-v3-small model was excluded from the ensemble due to its poor performance, the ensemble was thus formed using only the remaining models. Consequently, we opted for the ensemble model as the final submission system over the best models based on their performance shown in Table 1 below.

Table 1. Results of base models and our ensemble on the test set.

Model	Accuracy	Precision	Recall	F1
Naive Bayes	0.2923	0.2325	0.2923	0.1741
SVM	0.6577	0.6537	0.6577	0.6285
CNN	0.6098	0.6184	0.6098	0.6062
BiLSTM	0.5430	0.5344	0.5430	0.5145
SciBERT-uncased	0.7347	0.7301	0.7346	0.7293
SciBERT-cased	0.7279	0.7280	0.7279	0.7251
deBERTa-v3-small	0.6601	0.6592	0.6601	0.6546
RoBERTa-base	0.6984	0.6943	0.6984	0.6941
Ensemble	**0.7433**	**0.7423**	**0.7433**	**0.7391**

Table 2 showcases the performance of our ensemble model alongside that of the top four teams. Our system ranks fifth in terms of all metrics and shows comparable performance to the fourth-ranked system (”benjwolff”) across all metrics. This suggests a competitive position with potential for improvement. The top three systems (”saliq7,” ”rosni,” and ”flo.ruo”) achieved slightly higher Accuracy and F1 scores compared to ours. This highlights areas where further development can enhance our system’s ability to correctly classify examples and maintain a good balance between precision and recall.

Table 2. Results of our best submission compared with four top systems.

User	Accuracy	Precision	Recall	F1
saliq7	**0.7572 (1)**	0.7536 (3)	**0.7572 (1)**	0.7500 (3)
rosni	0.7558 (2)	**0.7566 (1)**	0.7558 (2)	**0.7540 (1)**
flo.ruo	0.7542 (3)	0.7545 (2)	0.7542 (3)	0.7524 (2)
benjwolff	0.7476 (4)	0.7438 (4)	0.7476 (4)	0.7426 (4)
kietnt0603 (Ours)	0.7433 (5)	0.7423 (5)	0.7433 (5)	0.7391 (5)

5.3 Error Analysis

Based on the large number of labels and to focus on insightful cases, the error analysis section will highlight the following classes as shown in Table 3

- **Biomedical Engineering and Bioengineering**: This class has a precision, recall, and F1-score of 0.00, indicating that the model is unable to correctly classify any instances of this class. With only 1 instance in the dataset, the model may be underfitting or overfitting this class due to the

Table 3. Performance metrics and instance count on test set for selected classes.

Class	F1-score	Recall	Precision	# Instances
Biomedical Engineering and Bioengineering	0.00	0.00	0.00	1
Quantitative Finance	0.67	0.91	0.53	11
Computational Engineering	0.07	0.06	0.10	17
Neuroscience and Neurobiology	0.63	0.46	1.00	13

severe lack of data. Analyzing the misclassified instance(s) and the features used could reveal why the model struggles with this class, potentially due to non-discriminative features or similarities with instances from other classes.

– **Quantitative Finance**: This class has a high recall of 0.91 but a relatively lower precision of 0.53, resulting in an F1-score of 0.67. The high recall suggests that the model is good at identifying instances of this class, but the lower precision indicates that it also tends to incorrectly classify instances from other classes as Quantitative Finance. Investigating the instances misclassified as Quantitative Finance could provide insights into the types of features or patterns that cause the model to make these errors, potentially leading to improvements in the feature set or model architecture.
– **Computational Engineering**: This class has very low precision (0.10) and recall (0.06), leading to a poor F1-score of 0.07. With 17 instances in the dataset, the class may not be well-represented, causing the model to underfit or overfit. Analyzing the misclassified instances could reveal whether there are specific types of instances or features that the model struggles with or whether the issue is more general, guiding strategies for improving the model's performance on this class.
– **Neuroscience and Neurobiology**: This class has a perfect precision of 1.00 but a relatively low recall of 0.46, resulting in an F1-score of 0.63. The high precision indicates that when the model classifies an instance as Neuroscience and Neurobiology, it is highly likely to be correct. However, the low recall suggests that the model misses many instances of this class, classifying them as other classes. Analyzing the instances misclassified as other classes could reveal patterns or features that the model overlooks or fails to capture for this class, potentially guiding improvements in feature engineering or model architecture.

Our error analysis revealed two primary challenges contributing to the model's performance in specific classes:

– **Limited taxonomic structure:** The transformation of the original classes's hierarchical or conceptual taxonomic structure into a single-label, multi-class format (The original structure of classes can be found here) eliminates explicit relationships between classes. This flattening impedes the model's ability to distinguish between categories that share inherent similarities or belong to broader, overlapping groups within the original hierarchy. This is a common challenge in NLP tasks involving multi-class classification with complex

and potentially overlapping categories, where inherent relationships between classes might not be readily captured by traditional classification approaches.
– **Imbalanced data distribution:** The limited number of instances for specific classes in the test set hinders the model's capacity to effectively learn the distinctive patterns necessary for accurate classification. This data imbalance is a well-documented issue in various NLP domains, and it can significantly impact the generalizability and robustness of machine learning models, particularly for underrepresented classes.

6 Conclusion and Future Work

This study presents a system for single-label, multi-class classification of scholarly papers. It leverages fine-tuned BERT models, ensemble voting, and the Focal Loss function to address potential class imbalance within the dataset. While achieving a rank of 5 in the shared task, the approach demonstrates promising potential for further advancement. For future work, addressing class imbalance through techniques beyond Focal Loss and extending the approach to classify other types of scientific literature holds significant potential for broader applicability.

Acknowledgements. This research was supported by The VNUHCM-University of Information Technology's Scientific Research Support Fund. We also thank the anonymous reviewers for their valuable comments on our manuscript.

References

1. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3615–3620. Hong Kong, China, November 2019
2. Chowdhury, S., Schoen, M.P.: Research paper classification using supervised machine learning techniques. In: 2020 Intermountain Engineering, Technology and Computing (IETC), pp. 1–6 (2020)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186 (Jun 2019)
4. He, P., Gao, J., Chen, W.: Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In: The Eleventh International Conference on Learning Representations (2022)
5. He, P., Liu, X., Gao, J., Chen, W.: Deberta: decoding-enhanced bert with disentageled attention. In: International Conference on Learning Representations (2021)
6. Kim, S.W., Gil, J.M.: Research paper classification systems based on TF-IDF and LDA schemes. HCIS **9**, 1–21 (2019)

7. Liu, Y., et al.: Roberta: a robustly optimized BERT pretraining approach. CoRR (2019)
8. Wang, H., Chen, B., Li, W.J.: Collaborative topic regression with social regularization for tag recommendation. In: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, pp. 2719–2725 (2013)
9. Wei, Y., He, Y., Yang, C.: Jetam: a joint embedding method for research paper mutil-label classification. In: 2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML), pp. 390–394 (2022)
10. Wolf, T., et al.: Transformers: state-of-the-art natural language processing. In: Liu, Q., Schlangen, D. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 38–45, October 2020
11. Zhang, X., Yu, X., Liu, X., Lyu, X.: Scientific paper classification by fusing BERT and GCN. In: 2023 International Conference on Intelligent Education and Intelligent Research (IEIR), pp. 1–6 (2023)

Advancing Automatic Subject Indexing: Combining Weak Supervision with Extreme Multi-label Classification

Lakshmi Rajendram Bashyam[1(✉)] and Ralf Krestel[1,2]

[1] ZBW – Leibniz Information Centre for Economics, Hamburg, Germany
l.rajendram-bashyam@zbw.eu, rkr@informatik.uni-kiel.de
[2] Kiel University, Kiel, Germany

Abstract. The multi-label automatic classification of scientific publications based on a pre-defined taxonomy, also called automatic subject indexing is a continuing research endeavor with significant cross-domain applicability. In this paper, we assess the performance of X-transformer and its variants with other extreme multi-label classification models for the above task. Our model *Weak X-transformer* achieves a micro F1-score of 0.65 and 64% accuracy on the task outperforming all other methods. We also investigate the impact of incorporating additional unlabelled data and hierarchical structure into the models. Our findings demonstrate that the transformer-based model with weak supervision outperforms other approaches, providing insights into effective strategies for extreme multi-label classification in scholarly publications.

Keywords: Extreme multi-label classification · Automatic subject indexing · Digital libraries · Semi-supervised learning

1 Introduction

Automatic subject indexing in libraries involves the use of computational techniques to assign relevant subject headings or descriptors to library resources such as books, articles, and other materials. This process utilizes algorithms, machine learning, and natural language processing to analyze the content of the resources and determine their main topics or subjects. By automating this task, libraries can efficiently organize their collections. Automatic subject indexing not only saves time and resources for librarians but also enhances the accuracy and consistency of indexing across the library catalog. In addition, automatic subject indexing also provides a keyword-based summary of the publication to the user. Additionally, it allows libraries to keep pace with the growing volume of digital materials and ensures that their collections remain organized and easily navigable in the digital age.

The subjects/labels assigned to the publication usually are obtained from a pre-defined thesaurus maintained by the respective subject authority. Typically

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 214–223, 2024.
https://doi.org/10.1007/978-3-031-65794-8_14

each publication is assigned multiple subjects by the indexers, thus making it a multi-label classification task. However, the nature of the topics observed in the publication makes the assigned subjects highly imbalanced. Depending on the type of thesaurus used, it can also be classified as an XMLC task.

In this work, we compare XMLC models on a shared task as a part of the challenge “FoRC: Field of Research Classification of Scholarly Publications” organized by the “Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024)” workshop. For the second task “Fine-grained multi-label classification of Computational Linguistics scholarly papers”, we evaluate multi-label and extreme multi-label classification models. Additionally tweaking it to support the hierarchical nature of the task.

2 Related Work

The task of extreme multi-label classification (XMLC) is characterized by an imbalanced distribution of labels, posing significant challenges, particularly in improving the performance of less frequent labels. BONSAI [5] and PARABEL [9] represent tree-based approaches widely adopted for addressing the XMLC problem. Building upon this foundation, the X-transformer [16] introduces innovations such as X-linear, recursive linear models, and XR-transformers, a transformer-based framework that recursively fine-tunes pre-trained transformers. The pecos [14] library offers a robust implementation of these models, along with several other recent XMLC solutions including PINA [2] and FINGER [1]. More recently, the XLGEN model [4] has explored leveraging a text-to-text transformer model to tackle the challenges posed by XMLC.

Assigning labels or subjects to scientific publications constitutes a fundamental aspect of library organization. The size of the thesaurus employed for this purpose varies significantly among different organizations. The annotators responsible for assigning labels typically adhere to a predefined methodology for conducting this task. However, due to the slow annotation process, a considerable number of publications remain unlabeled. Integrating semi-supervised techniques to augment the training data becomes imperative in such scenarios. Addressing this need, recent research [17] conducts a comprehensive analysis utilizing unlabeled data through weak supervision techniques. The authors compare the efficacy of well-known weak supervision methods, including COSINE and WRENCH [15], under real-world conditions characterized by limited availability of clean labels. Widely adopted libraries such as setfit [13] and skweak [6] have significantly contributed to the adoption of semi-supervised and weak supervision techniques in resource-constrained settings. A method for generating weak labels through a noisy labeling scheme and subsequent refinement via a two-level approach [3] offers a computationally efficient solution that can prove invaluable in low-resource scenarios.

Access to full-texts represents a significant asset within library systems. Annotators are furnished with the title, abstract, and full-text of a publication to ensure accurate labeling. Without all these resources, particularly full-texts, achieving human-level performance would pose a formidable challenge for any system. Previous research [7] has examined the performance of machine learning models when provided with an abundance of abstracts versus a comparatively limited number of full-texts.

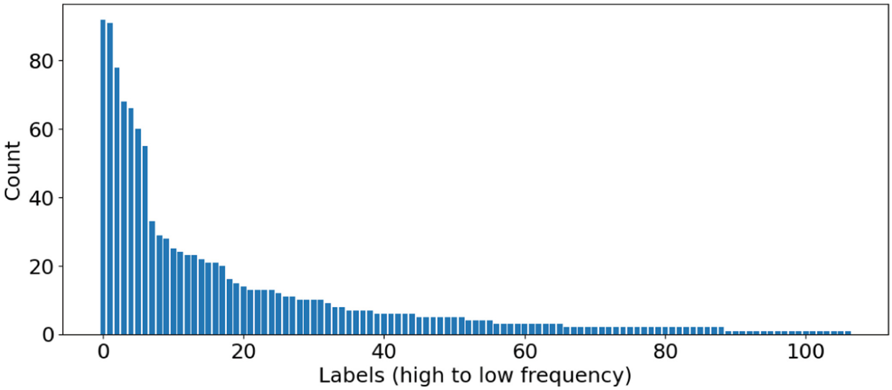

Fig. 1. Histogram of label distribution in the train set of FORC corpus

3 Data Analysis

The training set for task 2 has 1051 publications. Each publishing record contains the title, abstract, and acl-id. Other information fields, such as date, venue, and publisher, are also available for the majority of records. Each publication receives three levels of coarse-to-fine-grained labels: Level 1, Level 2, and Level 3. The annotated labels are based on a predefined taxonomy created by extracting subjects from the publication and correlating them with existing topics from multiple paper sources. As specified by the workshop guidelines, the quality of annotations is evaluated based on inter-annotator agreement scores (IAA) using Krippendorff's Alpha for multi-label annotations on each one of the three taxonomy levels. The average IAA scores for each level is given as part of the results in Table 3.

Figure 1 shows the distribution of labels across all levels. The figure clearly shows that the annotated labels are severely unbalanced. It was also discovered that several of the labels used in the validation and test splits did not appear once in the training split. This makes it more difficult to detect these labels.

In addition to the existing data, we were able to collect the full text of every paper using the acl-id [11]. The average length of full-text was approximately 16,000 words. The same source also gave abstracts for over 70,000 other publications from ACL articles and posters. The publications collected ranged from 2001 to 2021 and were published in a variety of venues. The additional publication dataset also includes the title, abstract, full-text, and all other metadata elements found in the FORC dataset. However, no labels are associated with the new dataset collected. Thus, we have approximately 70,000 unlabeled publication records from the same source as the FORC dataset.

4 Experiments

The text for training and testing the model is created by merging the fields *title*, *abstract*, *venue*, *publisher*, and *book title*, each separated by a specific token. By integrating the label sets from the three levels, we can define the Fine-grained multi-label classification task as a general multi-label classification. This allows us to use models intended for multi-label and XMLC tasks with minimal changes to the model.

The baseline model metrics were provided by the workshop committee. The metrics was reproduced by fine-tuning the scincl model [8] on FORC subtask 2 training dataset. The training text consists of combining only the title and abstract of each publication. Level wise labels for each publication were combined together as a list to form a multi-class multi-label classification task. The SciNCL is the state of the art pre-trained BERT language model to generate document-level embeddings of research papers. Since the training data for the FORC task consists of similar scientific documents, fine-tuning scincl model as baseline gives a good starting point.

We train a basic model like tf-idf on the train set in addition to the baseline that the organizers provided. Unlike the baseline, the tf-idf and all subsequent models was training on the train text created by merging *title*, *abstract*, *venue*, *publisher*, and *book title*, each separated by a specific token. The tf-idf model returns most similar subjects based on similarity in sparse tf-idf normalized bag-of-words vector space. Our XMLC models including tf-idf are trained using the Annif [12] toolset. Annif tf-idf implementation is based on the topic modelling library Gensim [10].

For this work, the label set size is 170. We model the problem as an extreme multi-label classification because this is on the higher end for a multi-label classification task. Regarding label occurrences, the label distribution in the picture likewise complies with Zipf's law. The high imbalance in the label distribution can affect the performance of the model. The model performs best on labels that appear frequently in the training set and poorly on labels that are rare. This is a typical occurrence in contexts with extreme multi-label classification. For the reasons listed above, this task could still benefit from being modeled as an XMLC problem even though most XMLC projects have label sets larger than 500 labels.

We use tree-based models like parabel [9] in the XMLC space. The label space is divided recursively by these models. Because there are an equal number of labels in each cluster, it is balanced. We tokenize the dataset using a nltk tokenizer and choose trigram tokens for training. We tune the parameters of the model such as number of clusters and maximum tree depth on the validation split of the dataset.

Furthermore, we utilize the implementation of transformer based XMC framework, the X-transformer framework [16] provided by the Pecos library [14]. This XR-transformer framework allows fine-tuning pre-trained transformers recursively on multi-resolution objectives. There are three steps involved in the fine-tuning. For starters, the label space is clustered. In the second stage, a matcher is trained to classify the publication to one of the clusters and finally, a ranker is trained to rank the labels inside each cluster. Based on this method, we train the model on the training set for 10 epochs with early stopping.

Further, we were able to obtain the full text for each publication in train, val and test split [11]. Due to hardware restrictions and because BERT-based transformer models only support up to 512 tokens. However, we train the parabel model on the full texts called parabel-ft and evaluate on the full-text corpus of the test split.

With the new dataset gathered, we now have access to more fields and around 70000 more ACL abstracts. However, they are not tagged with any labels. First, we train the X-transformer model on this unlabeled dataset to further enhance its performance. We produce weak labels from each unlabelled publication to enable supervised training. The previously trained tf-idf model is used to generate the weak labels. Since the model's performance determines the quality of these labels, they are noisy. We then combine the annotated clean labels and generate weak labels into the training dataset for the x-transformer model [3,17]. Finally, the model is then fine-tuned on the dataset containing combination of weak and clean labels. We name this model the weak X-transformer owing to the training data containing weak/noisy labels.

There are three levels of subjects, ranging from coarse to fine-grained subjects. The levels designated as Level1 > Level2 > Level3 are arranged in a rigid hierarchy. The hierarchy of the labels is not taken into account when training the model because we utilize a general XMLC model. As an alternative, we add hierarchy to the output of the model. This is accomplished by comparing the label set of each publication to the taxonomy hierarchy. The labels in Levels 2 and 3 that do not adhere to the preceding level's hierarchical structure are pruned from the collection of results. Furthermore, we only keep labels above a confidence score of 0.2 to increase the system's accuracy.

5 Results and Discussion

Table 1 provides the performance metrics for each model in our experiment. Overall, the baseline model performance is subpar. The poor macro average scores for recall, precision, and F1 indicate that the model does not work well for multi-label task. This suggests that optimizing a transformer-based model directly for our goal is not a good idea.

One of the easiest models to train and assess is the tf-idf model. It is interesting to note that the tf-idf model has very good recall scores despite having lower precision scores than other models. One reason for this could be that the lexical matching component identifies many labels in the publication text increasing

Table 1. Performance metrics of baseline and XMLC models

Model	Micro			Macro			Weighted		
	Prec	Rec	F1	Prec	Rec	F1	Prec	Rec	F1
weak x-transformer	0.64	0.65	**0.65**	**0.56**	0.52	**0.52**	0.64	0.65	**0.63**
x-transformer	**0.70**	0.60	0.64	0.50	0.37	0.40	**0.66**	0.60	0.61
parabel-ft	0.62	0.46	0.53	0.25	0.14	0.16	0.51	0.37	0.40
parabel	0.66	0.50	0.57	0.33	0.19	0.21	0.45	0.76	0.56
tfidf	0.41	**0.75**	0.53	0.44	**0.54**	0.45	0.47	**0.75**	0.55
baseline	0.35	0.32	0.34	0.01	0.04	0.02	0.16	0.33	0.17

the recall scores. However, a large number of the labels in the result set have no semantic relevance to the publication. Thus, a simple model could not adequately represent this.

Table 1 further shows the outcome of our tree-based parabel approach studies. The performance gain over the tf-idf model is not evident in the findings. The macro average scores are in fact decreasing. This suggests that the model has extremely low performance on some classes. Upon examination of the data, we discovered that the label set contains a large quantity of false negatives. In the next experiment, we again train a parabel model using the full-text dataset for the publications to refine it. The results of the parabel-ft model also demonstrate a decline in macro average scores. This suggests that training using full-text does not always result in improved performance. The reason for the degradation in performance could be that the model cannot identify the relevant parts of the text correctly.

Out of the models previously discussed, the weak X-transformer model, which is a combination of weak supervision applied to an X-transformer model designed for extreme multi-label classification, exhibits the best performance. Table 2 provides a few examples of the labels generated by our final model. The predicted label set is sorted by the confidence score from the weak X-transformer model. In terms of micro average scores, the model maintains a relatively high recall score while improving on the precision score. Both the weighted average scores and the macro scores show similar results. It obtains weighted averages, macro, and micro F1 scores of 0.64, 0.65, and 0.63 respectively.

Table 3 provides the level-wise performance of the model in the label hierarchy. We compare the multi-label outputs of our model with the actual labels at every label hierarchy. With a level1 label set, the X-transformer model obtains a high score for precision and recall metric of 0.90 each. Performance slightly declines as one moves up the tiers of the label hierarchy. The gradual decrease through the label hierarchy is also observed in the inter-annotator agreement (IAA) scores.

In the realm of XMLC, a common hurdle lies in enhancing the performance of less frequent labels. As depicted in Fig. 2, the precision scores of different models across each label class are illustrated. Labels are organized based on their

Table 2. Examples of labels obtained from the weak X-transformer model

ACL_ID:2021.sigdial-1.31		Title: Summarizing Behavioral Change Goals from SMS Exchanges to Support Health Coaches	
True Labels	Automatic Text Summarization Discourse Analysis Domain-specific NLP Data Management and Generation Model Architectures Transformer Models Extractive Text Summarization Data Preparation Medical and Clinical NLP NLP for Mental Health	Predicted Labels	Model Architectures Domain-specific NLP Data Management and Generation Classification Applications Discourse Analysis Dialogue Systems Medical and Clinical NLP Data Preparation NLP for News and Media
ACL_ID:2021.unimplicit-1.1		Title: Let's be explicit about that: Distant supervision for implicit discourse relation classification via connective prediction	
True Labels	Learning Paradigms Classification Applications Discourse Analysis Few-shot Learning	Predicted Labels	Model Architectures Learning Paradigms Discourse Analysis Classification Applications

occurrence frequency in descending order along the x-axis. Notably, the models generally exhibit superior performance for more commonly occurring labels compared to rare ones. Interestingly, the weak X-transformers model demonstrates higher precision scores for less frequent labels in comparison.

To improve on the previous model, we incorporate the unlabelled data into the training step. The result of the weak supervision/ weak x-transformer model is shown in the Table 1. While the enhancement in performance may not be substantial, the weak x-transformer still demonstrates superior performance across micro, macro, and weighted average scores in comparison. Both precision and recall scores are close to each other without any trade-offs of improvement of one score over the other. Hence, this model is comparatively better than the rest of the models.

Finally, in addition to the training of the model, label pruning is executed for each model. Pruning the labels that do not conform to hierarchical structure slightly improves the precision score by removing false labels without affecting the recall values negatively.

Table 3. Performance metrics of weak X-transformer model by each level of hierarchy and the associated inter-annotator agreement

Hierarchy	Prec	Rec	F1	IAA
Level 1	0.80	0.92	0.90	0.67
Level 2	0.80	0.73	0.76	0.58
Level 3	0.75	0.70	0.72	0.54

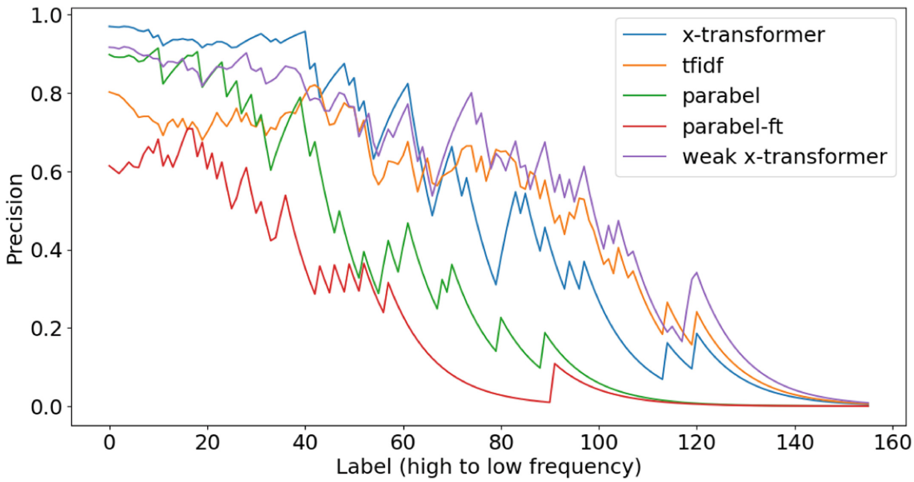

Fig. 2. Label-wise precision score for each model. The labels are arranged in descending order of count. For the presentation, the graph is smoothed.

6 Conclusion and Future Work

This study presents an approach to modeling hierarchical fine-grain multi-label classification as an XMLC problem. Furthermore, we assess how well simple models such as tf-idf to complex transformer-based models perform for the given task. We also explore other approaches, such as training with full-text and utilizing unlabeled datasets in a weak supervision setting. Our weak X-transformer model, which is our best-performing model, attains an F1 score of 0.65 across all labels.

In future work, we would like to explore the possibility of incorporating hierarchy during training instead of at the output level. Furthermore, We would like to experiment with generative AI by grounding the model to a graph RAG for label space.

References

1. Chen, P., Chang, W.C., Jiang, J.Y., Yu, H.F., Dhillon, I., Hsieh, C.J.: Finger: fast inference for graph-based approximate nearest neighbor search. In: Proceedings of the ACM Web Conference 2023, WWW 2023, pp. 3225–3235. Association for Computing Machinery, New York (2023). https://doi.org/10.1145/3543507.3583318
2. Chien, E., et al.: Pina: leveraging side information in extreme multi-label classification via predicted instance neighborhood aggregation. In: International Conference on Machine Learning, pp. 5616–5630. PMLR (2023)

3. Goh, G.B., Siegel, C., Vishnu, A., Hodas, N.: Using rule-based labels for weak supervised learning: a chemnet for transferable chemical property prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, pp. 302–310. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3219819.3219838
4. Jung, T., Kim, J., Lee, S., Kang, D.: Cluster-guided label generation in extreme multi-label classification. In: Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pp. 1670–1685. Association for Computational Linguistics (2023). https://doi.org/10.18653/v1/2023.eacl-main.122
5. Khandagale, S., Xiao, H., Babbar, R.: Bonsai: diverse and shallow trees for extreme multi-label classification. Mach. Learn. **109**(11), 2099–2119 (2020). https://doi.org/10.1007/s10994-020-05888-2
6. Lison, P., Barnes, J., Hubin, A.: skweak: weak supervision made easy for NLP. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations, pp. 337–346. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.acl-demo.40
7. Mai, F., Galke, L., Scherp, A.: Using deep learning for title-based semantic subject indexing to reach competitive performance to full-text. In: Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL 2018. ACM (2018). https://doi.org/10.1145/3197026.3197039
8. Ostendorff, M., Rethmeier, N., Augenstein, I., Gipp, B., Rehm, G.: Neighborhood contrastive learning for scientific document representations with citation embeddings. In: The 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022). Association for Computational Linguistics, Abu Dhabi (2022). https://doi.org/10.48550/arXiv.2202.06671, 7–11 December 2022. Accepted for Publication
9. Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., Varma, M.: Parabel: partitioned label trees for extreme classification with application to dynamic search advertising. In: Proceedings of the 2018 World Wide Web Conference, WWW 2018, pp. 993–1002. International World Wide Web Conferences Steering Committee (2018). https://doi.org/10.1145/3178876.3185998
10. Rehurek, R., Sojka, P.: Gensim–python framework for vector space modelling. NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic **3**(2) (2011)
11. Rohatgi, S.: ACL anthology corpus with full text. Github (2022). https://github.com/shauryr/ACL-anthology-corpus
12. Suominen, O.: Annif: Diy automated subject indexing using multiple algorithms. LIBER Quart.: J. Assoc. Eur. Res. Libr. **29**(1), 1–25 (2019). https://doi.org/10.18352/lq.10285. https://liberquarterly.eu/article/view/10732
13. Tunstall, L., et al.: Efficient few-shot learning without prompts (2022). https://doi.org/10.48550/ARXIV.2209.11055
14. Yu, H.F., Zhong, K., Zhang, J., Chang, W.C., Dhillon, I.S.: Pecos: prediction for enormous and correlated output spaces. J. Mach. Learn. Res. **23**(98), 1–32 (2022)
15. Yu, Y., Zuo, S., Jiang, H., Ren, W., Zhao, T., Zhang, C.: Fine-tuning pre-trained language model with weak supervision: a contrastive-regularized self-training approach. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1063–1077 (2021). https://doi.org/10.18653/v1/2021.naacl-main.84

16. Zhang, J., Chang, W., Yu, H., Dhillon, I.S.: Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. In: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems (NeurIPS), pp. 7267–7280 (2021)
17. Zhu, D., Shen, X., Mosbach, M., Stephan, A., Klakow, D.: Weaker than you think: a critical look at weakly supervised learning. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 14229–14253. Association for Computational Linguistics, Toronto, Canada (2023). https://doi.org/10.18653/v1/2023.acl-long.796

Single-Label Multi-modal Field of Research Classification

Florian Ruosch[1(✉)], Rosni Vasu[1], Ruijie Wang[1,2], Luca Rossetto[1], and Abraham Bernstein[1]

[1] Department of Informatics, University of Zurich, Zurich, Switzerland
{ruosch,rosni,ruijie,rossetto,bernstein}@ifi.uzh.ch

[2] University Research Priority Program "Dynamics of Healthy Aging", University of Zurich, Zurich, Switzerland

Abstract. The automated field of research classification for scientific papers is still challenging, even with modern tools such as large language models. As part of a shared task tackling this problem, this paper presents our contribution **SLAMFORC**, an approach to single-label classification using multi-modal data. We combined the metadata of papers with their full text and, where available, images into a pipeline to predict their field of research with an ensemble voting on traditional classifiers and large language models. We evaluated our approach on the shared task dataset and scored the highest values for two of the four metrics used in the evaluation of the competition, with the other two being the second highest.

Keywords: Natural Scientific Language Processing · Field of Research Classification · Multi-Modality

1 Introduction

Keywords and other classifications may help when searching or organizing scholarly publications [20]. They can be annotated by the authors or the publishers, with a corresponding manual effort, or may be machine-generated. The latter has been an application of natural language processing which, with the advent of pre-trained large language models such as BERT [14], has recently gained momentum. Still, the automated classification of research papers remains challenging [27].

This paper describes our submission to the shared task *Field of Research Classification of Scholarly Publications*[1] of the *1st Workshop on Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024)*. Its *Subtask I*, which our contribution addresses, is to develop a single-label classifier for general scholarly publications. We trained and tested it on a dataset of

[1] https://nfdi4ds.github.io/nslp2024/docs/forc_shared_task.

F. Ruosch and R. Vasu—Equal contribution.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 224–233, 2024.
https://doi.org/10.1007/978-3-031-65794-8_15

around 60,000 English scientific papers [1,2], each from one of 123 hierarchical classes of a subset of the Open Research Knowledge Graph taxonomy.[2] Our approach, dubbed **SLAMFORC** (short for *Single-Label Multi-modal Field of Research Classification*), is multi-modal in that we incorporated data from three different sources: the dataset provided by the organizers of the challenge containing metadata of the articles (e.g., title, abstract), the semantic information provided by Crossref[3], and the contents of the papers (i.e., full text and images). Using this data as features, we engineered a classifier that produces single-label predictions for a given input document. For this endeavor, we computed the embeddings with two different flavors of a pre-trained BERT [14] model and, subsequently, fed these vectors to a handful of traditional classifiers. Then, we applied a voting ensemble [21] to their output to combine them into a final classifier, incorporating all of them as well as the entirety of the available features.

The shared task was very competitive, with 13 system submissions. The margin among the top five submissions was very narrow (±0.75%), illustrating that the boundaries were pushed of what was possible with the provided data and task. In the end, our approach came in among the top results, scoring the highest values for two out of four evaluated aspects and the second-best for the others: accuracy (75.6%), precision (75.7%), recall (75.6%), and F1 (75.4%).

The remainder of this paper is structured as follows. Section 2 presents the related work, and Sect. 3 introduces our methodology. In the ensuing Sect. 4, we describe our experiments. Finally, we draw conclusions in Sect. 5.

2 Related Work

The classification of scholarly papers into research fields has found ample applications: for example, to ease organizing or searching the flood of new publications.

One such system [8] groups biomedical papers by applying non-negative matrix factorization [17] to the term relevance vectors of the documents. It uses bisecting k-means clustering [6], and, at the same time, assigns semantic meaning to each document and cluster inferred from the matrix decompositions.

The work by Taheriyan [27] describes an approach to classifying papers by using relationships such as common authors and references as well as citations in a graph. This information allows new papers to be assigned topics automatically instead of requiring manual annotations.

Nguyen and Shirai [20] focus on various text features such as the segmentation of the paper and apply three different classifiers: multi-label kNN [30], binary approach [28], and their newly proposed back-off model. While the latter performs the best, another interesting insight from their results is that only using the title, abstract, and the sections *Introduction* and *Conclusions* of papers improves over using the full text as a feature.

Another approach is presented by Kim and Gil [16]: They describe a classification system based on latent Dirichlet allocation [7] and term frequency-inverse

[2] https://orkg.org/fields.
[3] https://www.crossref.org/.

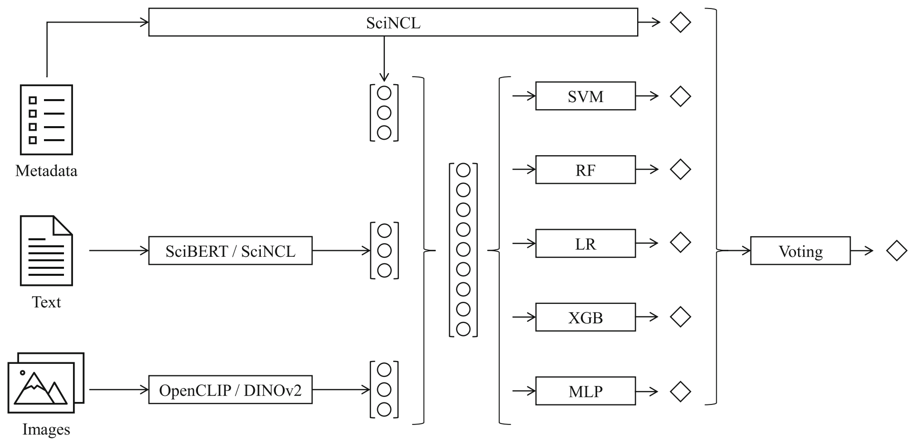

Fig. 1. Overview of the system architecture.

document frequency [25]. The former is employed to extract relevant keywords from the abstracts, the latter for k-means clustering [4] papers with similar topics.

More recently, SPECTER [12] uses pre-trained language models (e.g., SciBERT [5]) to generate document-level embeddings from the titles and abstracts. These can be used for downstream tasks, such as predicting the class of a document, which is demonstrated by applying SPECTER to a new dataset with papers in 19 classes. In this work, incorporating the entire text of papers remains an open issue due to limitations on memory and the availability of the paper contents.

3 The SLAMFORC System

This section describes our approach to solving the shared task. We first explain the multi-modal data of our system. Then, we detail the classifiers we used with this data.

Figure 1 shows an overview of the system. Its code is publicly available.[4]

3.1 Multi-modal Data

The dataset for the shared task [1,2] consisted of approximately 60,000 scholarly articles, compiled from various sources such as the *Open Research Knowledge Graph* [3], *arXiv*[5], *Crossref*[6], and the *Semantic Scholar Academic Graph* [29]. It spans 123 fields of research (FoR) across five major domains and four hierarchical

[4] https://gitlab.ifi.uzh.ch/DDIS-Public/forc24.
[5] https://arxiv.org.
[6] https://www.crossref.org/.

levels, with mapping to the *ORKG* taxonomy.[7] The challenge of imbalanced data is evident in the dataset, where the distribution of fields is uneven, varying from as low as eight articles (for *Molecular, Cellular, and Tissue Engineering*) to over $6,000$ (for *Physics*).

We utilized Crossref (see Footnote 3) to further enhance the text data of papers. Specifically, for each paper, we used its Digital Object Identifier (DOI) and the Crossref API client[8] to retrieve its annotated subjects and references from the Crossref Unified Resource API.[9] For the paper with the DOI "*10.1007/JHEP06(2012)126*," for example, we retrieved the subject "*Nuclear and High Energy Physics*" and the metadata of 37 reference papers. Despite Crossref adopting a different taxonomy, this retrieved subject remains highly useful for predicting the target label of this paper (i.e., "*Physics*"). Also, the reference papers are mostly in the Physics domain, and this information can be very useful.

We used the title, abstract, and publisher information from the provided dataset, along with the subject data, to generate the metadata embeddings for each paper. We appended all this data as input text to SciNCL [23], a pre-trained BERT model, for computing an embedding as a comprehensive representation of each paper.

In order to make use of the full text for the papers in the dataset, we first had to obtain the respective documents. This was straightforward for items that already had a download link annotated. For all other papers, we used the DOI field, where available, to find the PDFs. There were some cases where neither was available. For those, we relied on Crossref's API to resolve the paper title to its DOI, which allowed us to download the full text document, if it was available.

To extract the text from the PDFs, we employed PaperMage [19]. For each PDF, it produces a JSON file with information about its content and structure. We only relied on the extracted symbols, which we used to reconstruct the full text of the respective papers. Using this data, we computed the document-level embeddings with two pre-trained BERT models: SciBERT [5] and SciNCL [23]. Because of BERT's limitation to processing 512 tokens at a time [14] and papers exceeding this, we batched the input data accordingly. We employed a sliding window of size 512 tokens with an overlap of 128 to conserve semantics near the window borders. After computing the embeddings for each such chunk, we averaged them to obtain the final document-level embedding.

To incorporate the visual information contained in the PDFs, we extracted all their images and converted them to raster graphics. For each image, we used an OpenCLIP [11] model pre-trained on the LAION-5B dataset [26] as well as a pre-trained DINOv2 [22] model to extract image features. When PDFs contained multiple images, we used mean-pooling to aggregate the multiple feature vectors per model, resulting in two vectors per PDF; one for each applied model. For

[7] https://huggingface.co/spaces/rabuahmad/forcI-taxonomy/blob/main/taxonomy.json.
[8] https://github.com/fabiobatalha/crossrefapi.
[9] https://api.staging.crossref.org/swagger-ui/index.html.

Table 1. The first results on the *validation set* of the individual classifiers with all features (embeddings of metadata, full text, and images) as measured by accuracy, weighted precision, recall, and F1.

Classifier	Accuracy	Precision	Recall	F1
Support Vector Machine	0.755	0.754	0.755	0.752
Random Forest	0.755	0.754	0.755	0.753
Logistic Regression	0.750	0.755	0.750	0.750
XGBoost	0.731	0.735	0.731	0.731
Multilayer Perceptron	0.743	0.748	0.743	0.743

papers where the PDF did not contain any images or the PDF was not available, these vectors were set to zero.

3.2 Classifier

For the final system, we used a mixture of traditional classifiers and neural methods that we combined with an ensemble voting method [21]. Figure 1 shows an overview of the system. After computing the embeddings for the various data sources, we trained several classifiers that could handle vectors as input and predict the single-label class for each item in the dataset.

An obvious choice are Support Vector Machines [13], or SVM for short. Due to the nature of the input data, they can naturally classify them in a high-dimensional space and predict the field of research label. We employed a Random Forest (RF) [15] since they avoid overfitting to the training data, which was an overt problem to be expected because of the skew in classes in the dataset. Logistic Regression (LR) is another widely used traditional classifier to predict single labels on linearly separable data. With eXtreme Gradient Boosting (XGB) [10], we used another popular method that can achieve good performance while sacrificing interpretability. Next, we also employed a fully connected neural net that is a Multilayer Perceptron (MLP), able to deal with not linearly separable data. Furthermore, we also trained SciNCL [23] as an end-to-end solution on the metadata.

Finally, we combined the output of the classifiers described above into an ensemble method [21] with hard voting [18]. This enabled the use of all techniques and all available data at the same time while still producing a single predicted label for each item in the dataset.

4 Experiments

Table 1 shows the results of the initial experiments. We used a set of traditional classifiers as implemented by scikit-learn [24] with all of the available data for each paper consisting of the stacked embedding vectors. Since no method significantly outperformed the others, we combined all of them post-hoc using a

Table 2. Ablation study on the *validation set* by feature combination.

Meta	Text	CLIP	DINO	Accuracy	Precision	Recall	F1-Score
✓				0.774	0.776	0.774	0.773
	✓			0.437	0.709	0.437	0.463
		✓		0.254	0.552	0.254	0.241
			✓	0.259	0.505	0.259	0.250
✓	✓			0.776	0.778	0.776	0.775
✓		✓		0.772	0.775	0.772	0.772
✓			✓	0.776	0.777	0.776	0.774
	✓	✓		0.438	0.700	0.438	0.463
	✓		✓	0.437	0.700	0.437	0.461
		✓	✓	0.271	0.529	0.271	0.265
✓	✓	✓		0.776	0.777	0.776	0.775
✓	✓		✓	0.779	0.781	0.779	0.778
✓		✓	✓	0.776	0.778	0.776	0.774
	✓	✓	✓	0.433	0.701	0.433	0.456
✓	✓	✓	✓	0.777	0.779	0.777	0.775

voting ensemble method, giving us our final classifier for the results of which we submitted to the shared task.

To illustrate the impact of each data source and dissect our multi-modal approach, we performed a feature ablation study, the results of which are shown in Table 2. We used our final system architecture with all classifiers combined with voting on the powerset of possible feature combinations. It is evident that the (embeddings of the) metadata have the most positive influence on the results. Still, adding extra information to the classifier is not detrimental but rather contributes to a higher score. This holds more for the (embeddings of the) full texts of the papers which perform decent on their own. Using the embeddings of the images in the papers alone, where applicable, achieves clearly worse results than the other two data sources. Nevertheless, the combination of all features is among the highest scoring for all four employed metrics, and there was no reason not to rely on everything available.

Finally, Table 3 shows the results of the shared task evaluation.[10] Our submission (ID 683689, top row) scored the highest for precision (75.7%) and F1 (75.4%) while achieving the second-best values for accuracy (75.6%) and recall (75.6%). This goes to show that our multi-modal approach worked and performed well in this competition. Without further knowledge of the other systems, no comparisons can be made or insights gained, and are, thus, left for future work. In conclusion, the automated field of research classification of scientific papers is still challenging, but the submissions for this shared task seemed to have pushed the boundaries of what was possible with the given tools and information, seeing how close the top results were.

[10] https://codalab.lisn.upsaclay.fr/competitions/16684#results.

Table 3. The final evaluation results on the *test set* as measured by accuracy, weighted precision, recall, and F1 (best for each in **bold**, runner-up underlined). Our submission is the first line.

ID	Accuracy	Precision	Recall	F1 ↓
683689	0.756	**0.757**	0.756	**0.754**
688995	0.754	0.755	0.754	0.752
687946	**0.757**	0.754	**0.757**	0.750
689747	0.748	0.744	0.748	0.743
688510	0.743	0.742	0.743	0.739
651741	0.730	0.725	0.730	0.724
649383	0.726	0.719	0.726	0.720
686435	0.706	0.703	0.706	0.693
686384	0.702	0.695	0.702	0.692
689251	0.682	0.679	0.682	0.678
689454	0.658	0.659	0.658	0.653
647796	0.058	0.061	0.058	0.057
678150	0.004	0.002	0.004	0.002

5 Conclusions

In this paper, we presented **SLAMFORC**, a system for the *Single-Label Multimodal Field of Research Classification*. We used it to produce the results for our submission to the shared task *Field of Research Classification of Scholarly Publications*. Pursuing a multi-modal approach by incorporating not only the given dataset containing metadata of the papers but also the full text of publications as well as images in these documents, we built an ensemble classifier by combining a set of traditional classifiers using a voting ensemble. We computed the embeddings with pre-trained large language models, stacked these vectors, and trained the individual classifiers. Then, we used them jointly to obtain a single-label prediction for each item in the dataset.

As one of the conclusions of this work, we would like to raise some issues with the evaluation method. Possibly, some metric that also considers the semantics in the taxonomy might have enabled a more effective evaluation and allowed for insights into the inner workings of the systems, especially in connection with the misclassified items. One such metric was proposed by Chen et al. [9], which evaluates the performance of taxonomic assignments based on said given taxonomy.

Our system achieved the highest precision and F1 and the second-best accuracy and recall values of all submissions, demonstrating its effectiveness. While the ceiling seems to have been reached of what was possible in the shared task, judging by the range of the top submissions We hope to have contributed to the still challenging classification of research fields for scientific publications.

Acknowledgments. This work was partially funded by the University Research Priority Program "Dynamics of Healthy Aging" at the University of Zurich and the Swiss National Science Foundation through the projects CrowdAlytics (Grant Number 184994) and MediaGraph (Grant Number 202125).

References

1. Abu Ahmad, R., Borisova, E., Rehm, G.: FoRC-Subtask-I@NSLP2024 Testing Data (2024). https://doi.org/10.5281/zenodo.10469550
2. Abu Ahmad, R., Borisova, E., Rehm, G.: FoRC-Subtask-I@NSLP2024 Training and Validation Data (2024). https://doi.org/10.5281/zenodo.10438530
3. Auer, S., et al.: Improving access to scientific literature with knowledge graphs. Bibliothek Forschung und Praxis **44**(3), 516–529 (2020)
4. Balabantaray, R.C., Sarma, C., Jha, M.: Document clustering using k-means and k-medoids. CoRR abs/1502.07938 (2015). http://arxiv.org/abs/1502.07938
5. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, 3–7 November 2019, pp. 3613–3618. Association for Computational Linguistics (2019). https://doi.org/10.18653/V1/D19-1371
6. Bishop, C.M.: Pattern Recognition and Machine Learning, 5th edn. Information Science and Statistics. Springer, Cham (2007). https://www.worldcat.org/oclc/71008143
7. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003). http://jmlr.org/papers/v3/blei03a.html
8. Bravo-Alcobendas, D., Sorzano, C.: Clustering of biomedical scientific papers. In: 2009 IEEE International Symposium on Intelligent Signal Processing, pp. 205–209. IEEE (2009)
9. Chen, C.Y., Tang, S.L., Chou, S.C.T.: Taxonomy based performance metrics for evaluating taxonomic assignment methods. BMC Bioinform. **20**(1), 310 (2019). https://doi.org/10.1186/s12859-019-2896-0
10. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2016, pp. 785–794. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2939672.2939785
11. Cherti, M., et al.: Reproducible scaling laws for contrastive language-image learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, 17–24 June 2023, pp. 2818–2829. IEEE (2023). https://doi.org/10.1109/CVPR52729.2023.00276
12. Cohan, A., Feldman, S., Beltagy, I., Downey, D., Weld, D.S.: SPECTER: document-level representation learning using citation-informed transformers. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J.R. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, 5–10 July 2020, pp. 2270–2282. Association for Computational Linguistics (2020). https://doi.org/10.18653/V1/2020.ACL-MAIN.207
13. Cortes, C., Vapnik, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1007/BF00994018

14. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). https://doi.org/10.18653/v1/N19-1423. https://aclanthology.org/N19-1423
15. Ho, T.K.: Random decision forests. In: Third International Conference on Document Analysis and Recognition, ICDAR 1995, Montreal, Canada, 14–15 August 1995, vol. I, pp. 278–282. IEEE Computer Society (1995). https://doi.org/10.1109/ICDAR.1995.598994
16. Kim, S., Gil, J.: Research paper classification systems based on TF-IDF and LDA schemes. Hum. Centric Comput. Inf. Sci. **9**, 30 (2019). https://doi.org/10.1186/S13673-019-0192-7
17. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
18. Littlestone, N., Warmuth, M.K.: The weighted majority algorithm. Inf. Comput. **108**(2), 212–261 (1994)
19. Lo, K., et al.: PaperMage: a unified toolkit for processing, representing, and manipulating visually-rich scientific documents. In: Feng, Y., Lefever, E. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 495–507. Association for Computational Linguistics, Singapore (2023). https://doi.org/10.18653/v1/2023.emnlp-demo.45. https://aclanthology.org/2023.emnlp-demo.45
20. Nguyen, T.H., Shirai, K.: Text classification of technical papers based on text segmentation. In: Métais, E., Meziane, F., Saraee, M., Sugumaran, V., Vadera, S. (eds.) NLDB 2013. LNCS, vol. 7934, pp. 278–284. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38824-8_25
21. Opitz, D., Maclin, R.: Popular ensemble methods: an empirical study. J. Artif. Intell. Res. **11**, 169–198 (1999)
22. Oquab, M., et al.: DINOv2: learning robust visual features without supervision. CoRR abs/2304.07193 (2023). https://doi.org/10.48550/ARXIV.2304.07193
23. Ostendorff, M., Rethmeier, N., Augenstein, I., Gipp, B., Rehm, G.: Neighborhood contrastive learning for scientific document representations with citation embeddings. In: Goldberg, Y., Kozareva, Z., Zhang, Y. (eds.) Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pp. 11670–11688. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (2022). https://doi.org/10.18653/v1/2022.emnlp-main.802. https://aclanthology.org/2022.emnlp-main.802
24. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
25. Ramos, J., et al.: Using TF-IDF to determine word relevance in document queries. In: Proceedings of the First Instructional Conference on Machine Learning, vol. 242, pp. 29–48. Citeseer (2003)
26. Schuhmann, C., et al.: LAION-5B: an open large-scale dataset for training next generation image-text models. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, 28 November–9 December 2022 (2022). http://papers.nips.cc/paper_files/paper/2022/hash/a1859debfb3b59d094f3504d5ebb6c25-Abstract-Datasets_and_Benchmarks.html

27. Taheriyan, M.: Subject classification of research papers based on interrelationships analysis. In: Proceedings of the 2011 Workshop on Knowledge Discovery, Modeling and Simulation, pp. 39–44 (2011)
28. Tsoumakas, G., Katakis, I., Vlahavas, I.: Mining multi-label data. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook. Springer, Boston (2009). https://doi.org/10.1007/978-0-387-09823-4_34
29. Wade, A.D.: The semantic scholar academic graph (S2AG). In: Companion Proceedings of the Web Conference 2022, WWW 2022, p. 739. Association for Computing Machinery, New York (2022). https://doi.org/10.1145/3487553.3527147
30. Zhang, M., Zhou, Z.: ML-KNN: a lazy learning approach to multi-label learning. Pattern Recogn. **40**(7), 2038–2048 (2007). https://doi.org/10.1016/J.PATCOG.2006.12.019

Enriched BERT Embeddings for Scholarly Publication Classification

Benjamin Wolff(✉), Eva Seidlmayer, and Konrad U. Förstner

ZB MED – Information Centre for Life Sciences, Cologne, Germany
wolff@zbmed.de

Abstract. With the rapid expansion of academic literature and the proliferation of preprints, researchers face growing challenges in manually organizing and labeling large volumes of articles. The NSLP 2024 FoRC Shared Task I addresses this challenge organized as a competition. The goal is to develop a classifier capable of predicting one of 123 predefined classes from the Open Research Knowledge Graph (ORKG) taxonomy of research fields for a given article. This paper presents our results.

Initially, we enrich the dataset (containing English scholarly articles sourced from ORKG and arXiv), then leverage different pre-trained language Models (PLMs), specifically BERT, and explore their efficacy in transfer learning for this downstream task. Our experiments encompass feature-based and fine-tuned transfer learning approaches using diverse PLMs, optimized for scientific tasks, including SciBERT, SciNCL, and SPECTER2. We conduct hyperparameter tuning and investigate the impact of data augmentation from bibliographic databases such as OpenAlex, Semantic Scholar, and Crossref. Our results demonstrate that fine-tuning pre-trained models substantially enhances classification performance, with SPECTER2 emerging as the most accurate model. Moreover, enriching the dataset with additional metadata improves classification outcomes significantly, especially when integrating information from S2AG, OpenAlex and Crossref. Our best-performing approach achieves a weighted F1-score of 0.7415. Overall, our study contributes to the advancement of reliable automated systems for scholarly publication categorization, offering a potential solution to the laborious manual curation process, thereby facilitating researchers in efficiently locating relevant resources.

Keywords: Scholarly Publication Classification · BERT-Embeddings for scientific tasks · Enrichment of Scholarly Publications

1 Introduction and Background

In academic publishing, we currently see two trends converging: the exponential growth of publications, on average doubling every 14 years [2], as well as the increased frequency of publication, due to the growing importance of preprints, which accelerates the publishing process [6]. The vast amount of publications

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 234–243, 2024.
https://doi.org/10.1007/978-3-031-65794-8_16

presents a challenge to researchers in distinguishing relevant papers from irrelevant ones concerning a specific research question. Thus, a fine-grained categorization provides researchers with an invaluable tool for finding relevant resources. Manual curation is not a feasible approach. Besides that, categorization is not static. Instead, it has a dynamic nature: New categories emerge over time, while others evolve or disappear. Moreover, the compiled categories depend on the underlying platforms, where the publications are indexed. Thus we need reliable automated systems for the task of scholarly publication categorization. This paper stands within the scope of the NSLP2024[1] and is dedicated to the Field of Research Classification (FoRC) shared task, Subtask I. The goal of Subtask I is to develop a classifier that predicts one of 123 predefined classes from Open Research Knowledge Graph (ORKG) taxonomy of research fields for a given article. The competition hosts provide a dataset of English articles (split into test and validation sets), compiled from ORKG[2] and arXiv[3], which suggests a classifier that is trained in a supervised manner. The dataset consists of a selection of metadata fields. The shared task is run as a competition. Submitted systems will be evaluated on a test set, using accuracy and weighted scores of recall, precision, and F1.

With the advent of transformer models, the field of Natural Language Processing (NLP) underwent significant changes. Pre-trained Language Models are the current state-of-the-art across many NLP tasks and achieve top positions in General Language Understanding Tasks[4]. FoRC Shared Task I represents a task of scholarly document classification, a task where BERT [3], an encoder model for text embeddings, has demonstrated excellent results in a variety of benchmarks [12]. BERT is available in different flavors, pre-trained on specific tasks, and can be further fine-tuned to adapt it to downstream tasks.

Significant research effort is put towards automated classification into scholarly publication categorization. Automating this task could replace the time-consuming and costly manual curation process, which relies on experts in the field. Garcia et al. [4] explore how BERT and its variants effectively use specific words, semantically relevant to research fields in their final layer. In SciRepEval [12] the authors introduce a benchmark, which applies 24 realistic tasks for use cases involving scientific document embeddings. The results yield a novel model called SPECTER2. Ostendorff et al. [8] show how BERT embeddings, when combined with metadata and knowledge graph embeddings, improve results in a document classification task.

1 https://nfdi4ds.github.io/nslp2024/.
2 https://orkg.org/.
3 https://arxiv.org/.
4 https://gluebenchmark.com/.

2 Dataset

The provided dataset contains the following fields: abstract, author, DOI, URL, publication month, publication year, title, publisher and label. However, these fields are incomplete for many items. Particularly, the absence of a DOI becomes problematic later on during the enrichment process, as we can not use a unique identifier for further information retrieval from bibliometric databases. In total, the dataset comprises 50,441 items (train: 41,540, validation: 8901). For evaluation purposes, initially, a test set without labels was provided, which was distributed with labels after the end of the competition's deadline. The dataset (train and validation) shows significant class imbalance with a multitude of items assigned to the fields "Physics and Mathematics" ($\approx$ 79.5%), while other fields like "Arts and Humanities" are largely underrepresented, with only a handful of entries ($\approx$ 0.14%).

3 Enrichment

We encountered a challenge with enrichment due to the absence of DOIs in the provided article metadata collection. In the training dataset, 16,977 DOIs were missing, while the test and validation datasets had 3,712 and 3,673 gaps in the DOI column, respectively. To fill the missing DOI, we used the article titles to query the OpenAlex (OA)[5] database, a bibliographic repository containing research literature metadata. In cases where the title was not suitable for retrieval, we decided to clean it by removing special characters like line breaks, tabs, multiple spaces, as well as LaTeX tags. Subsequently, we successfully retrieved DOIs for numerous titles, filling 7,921 gaps in the training dataset and adding 1,702 and 1,710 DOIs to the test and validation datasets, respectively.

For further metadata enrichment we used three bibliographic databases: OpenAlex, Semantic Scholar Academic Graph (S2AG)[6] and Crossref (CR)[7]. These sources were employed to fill gaps in the provided original dataset and enhance it with additional categories. S2AG is a large, open, heterogeneous knowledge graph of scholarly works, authors and citations, developed by Allen Institute for AI[8], suitable for enrichment purposes. CR is an association of publishers that provides a database of metadata for scientific literature. From the mentioned providers, we collected the following metadata:

- Openalex: topics, subtopics, concepts, keywords and external identifiers
- S2AG: fields of study
- Crossref: journal title, (research) subjects

[5] https://openalex.org/.
[6] https://www.semanticscholar.org/product/api.
[7] https://www.crossref.org/.
[8] https://allenai.org/.

4 Approaches

4.1 BERT-Embeddings

In our first approach we decided to use a single BERT model to derive document embeddings, followed by a dense layer for the classification task. Since the BERTbase model uses 12 transformer blocks, and a hidden dimension size of 768, a single dense layer is sufficient to separate the documents for classification. To fully leverage the power of pre-trained models and achieve a good performance, there are two important questions to consider:

1. Which available pre-trained models suit the given task?
2. Which hyperparameter setting yields the best performance?

Furthermore, we aim to investigate the impact of transfer learning on selected pre-trained models by comparing feature-based transfer learning and fine-tuning. Additionally, we examine the effect of enriched data on the classification result.

Selecting Pre-trained Models. For selecting promising pre-trained models for our downstream task, we rely on the insights from the SciRepEval benchmark [12] and chose the top-performing models identified for scientific tasks as potential candidates: SciBERT [1], SciNCL [9] and SPECTER2 [12]. Additionally, the BERTbase model served as a baseline [3]. SciBERT, a variant of BERT, is specifically trained on a vast collection of scientific publications to optimize its performance on scientific downstream tasks. SciNCL, initialized with the SciBERTs weights, utilizes citation graph neighborhood to generate samples for Contrastive Learning, a technique used to learn representations by contrasting positive and negative examples. SPECTER2 employs a BERTbase model, which is trained from scratch using citation links, similar to the original SPECTER model. However, the training data for SPECTER2 comprises 10 times more triplets spanning 23 Fields of Study[9].

Hyperparameter Tuning. To find the best hyperparameter settings, we performed a grid search. In grid search we define a range of values for each hyperparameter. All possible combinations of these values are explored systematically and evaluated to identify the best-performing combination. The models' performance was evaluated on BERTbase with weighted F1-Score on the test set. We employed the following parameters and ranges for grid search (best parameter combination is highlighted): Batchsize: [8, 16, **32**], Learningrate: [1e−3, **1e−4**, 1e−5], Weight Decay: [1e−1, 1e−2, **1e−3**]. We experimented with a range of 3–5 training epochs. During evaluation, we observed signs of overfitting after 3 epochs. This was evident by a sharp decrease in loss on the training data while loss on the validation set simultaneously increased. We maintained hyperparameter settings constant for all experiments.

[9] https://github.com/allenai/SPECTER2.

Transfer Learning. For an assessment of the selected pre-trained models, we evaluated the performance between the feature-based and fine-tuned transfer learning: In feature-based transfer learning parameters of the pre-trained models were frozen. Only the task-specific classifier on top was trained, in the fine-tuned transfer learning the parameters of the pre-trained models are additionally adapted to the task. For the basic evaluation, we did not utilize enrichments. We relied solely on title and abstract. For a more detailed overview, we ran both approaches on the models selected in the previous step.

Table 1. Comparison of evaluation metrics for feature-based and finetuned transfer learning on a selection of different PLMs.

	Feature-Based				Finetuned			
	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
BERTbase	0.2033	0.0887	0.2033	0.1013	0.6973	0.6838	0.6973	0.6868
SciBERT	0.3313	0.2368	0.3313	0.2179	0.7259	0.7223	0.7259	0.7206
SciNCL	0.490	0.4497	0.490	0.4095	0.7285	0.7245	0.7285	0.7239
SPECTER2	0.4351	0.4035	0.4351	0.3364	0.7330	0.7285	0.7330	0.7283

Raw Data vs Enriched Data. We utilized the gathered enrichments to enhance the classification performance, assuming that these additional pieces of information could significantly influence the outcome. Since BERT has a limited input capacity (maximum of 512 tokens), we decided to use only a selection of the data enrichments. Moreover, it is crucial to determine the order and manner in which the enrichments are fed into the model. First, we selected enrichments from which we expected the most informative content, while consuming few additional input tokens. We first experimented with the data gathered from S2AG and OpenAlex and subsequently integrated enrichments from CrossRef in a second step. We decided to integrate the following enrichments:

- from S2AG: Fields of Study (FoS)
- from OpenAlex: Concepts and Topics
- from CrossRef: Journal Title, Subjects

We opted for the following order: title, fields of study, topics, abstract, concepts, categories, journal title and subject. Each of the fields was separated by a [SEP]-token. Additionally, the field name was prepended to the terms to provide context (except for title and abstract). Our final pipeline is depicted in Fig. 1.

4.2 Combined BERT-Embeddings (TwinBERT)

We explored alternative approaches to enhance the methodology described earlier, considering BERT limitations: 512 token limit and treating the entire input

Table 2. Comparison of evaluation metrics for raw data (title and abstract) and enriched data from S2AG, OpenAlex (OA) and CrossRef (CR).

	SPECTER2				SciNCL			
	Acc	Prec	Rec	F1	Acc	Prec	Rec	F1
Title+Abstract (TA)	0.7330	0.7285	0.7330	0.7283	0.7285	0.7245	0.7285	0.7239
TA+S2AG+OA	0.7419	0.7386	0.7419	0.7367	0.7353	0.7299	0.7353	0.7256
TA+S2AG+OA+CR	<u>0.7467</u>	<u>0.7438</u>	<u>0.7467</u>	<u>0.7415</u>	0.7340	0.7275	0.7340	0.7246

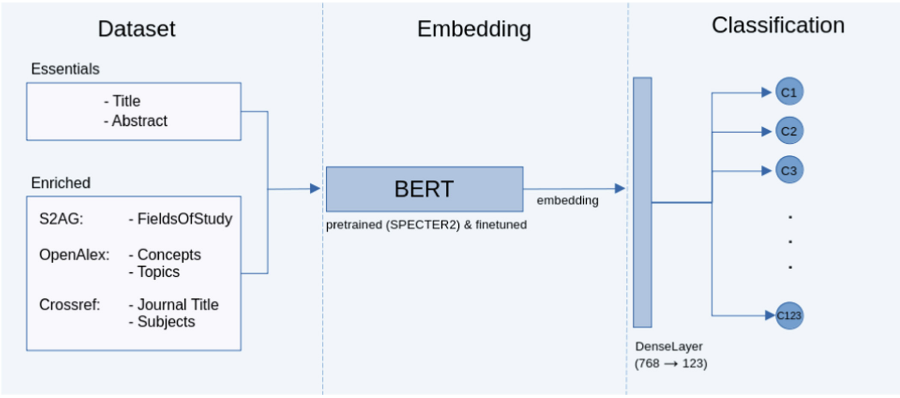

Fig. 1. Workflow using a single BERT-model with enrichments and SPECTER2 (best performing model according to Table 2)

as a single document. To overcome these issues, we experimented with a custom model that employs two BERT models in parallel. In literature, we found several similar approaches already known as TwinBERT [5,7], but in different contexts. The use of two BERT models provides separate embedding pathways: one for the basic document (title and abstract) and another for metadata. These embeddings are then concatenated before classification. By utilizing two BERT models, the number of parameters more than doubles. Thus we aimed to determine if this additional effort is justified. When experimenting with TwinBERT, we varied the number of transformer layers, to account for this consideration. We developed TwinBERT as a custom model using PyTorch [11]. Our TwinBERT model is depicted in Fig. 2. The results are presented in Table 3.

5 Results

Table 1 shows the evaluation results for transfer learning without any enrichments. In feature-based transfer learning (no fine-tuning) across different models, we achieve a weighted F1-score ranging from 0.1013 to 0.4095. While this result demonstrates a general ability to address the required task, the quality falls

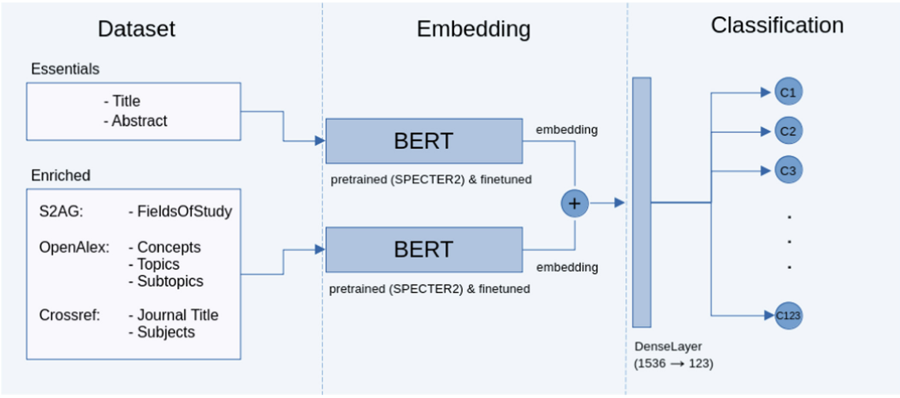

Fig. 2. Twin-Bert Model, which utilizes two BERT models, to separate embeddings from essential data (title, abstract) from enriched data.

Table 3. Comparison of evaluation metrics for TwinBERT model with varying number of transformer layers.

# of Layers	TwinBERT			
	Acc	Prec	Rec	F1
12 Layers	0.1145	0.0131	0.1145	0.0235
8 Layers	0.1280	0.0278	0.1280	0.0444
6 Layers	0.6179	0.5940	0.6179	0.5959
4 Layers	0.6688	0.6560	0.6688	0.6577
2 Layers	<u>0.6719</u>	<u>0.6597</u>	<u>0.6719</u>	<u>0.6618</u>

short of an acceptable range. Nevertheless, models optimized for scientific publications outperform the BERTbase model with a significant margin, with SciNCL leading. In fine-tuned transfer learning, the results for these models are closely clustered, with the weighted F1-score ranging from 0.6868 to 0.7283. SciBERT, SciNCL, and SPECTER2 achieve F1-scores that are nearly equivalent (between 0.7206 and 0.7283), with SPECTER2 now taking the lead position. These results suggest that for the given task, fine-tuning plays a more significant role than the actual selection of the pre-trained model. Compared to the BERTbase model, the models pre-trained for scientific tasks perform significantly better, although the results of the fine-tuned BERT Base model are quite acceptable.

Table 2 presents the evaluation results for finetuned transfer learning, incorporating enrichments. We decided to assess the two best-performing pre-trained models from the preceding task (SciNCL and SPECTER2). With enrichments, we observe an improvement in the evaluation metrics for both models. However the enrichments yield a more substantial improvement for SPECTER2. The F1-score for SPECTER2 increases by 0.013, while for SciNCL, it only rises by a slight margin of 0.0017. We find that combining enrichments from multiple sources (S2AG, OA, CR) enhances the classification process without adding noise. Our evaluation demonstrates the superior performance of the SPECTER2 model, fine-tuned for the task. The best results are achieved with enrichments from various sources (S2AG, OA, CR) with an F1-score of 0.7415.

Table 3 shows the evaluation results for our custom TwinBERT model. Despite its increased complexity, and the option to incorporate additional enrichments, this model yields unsatisfactory results. During the evaluation, we simplified the model by gradually reducing the number of transformer layers. With each layer removed, the model's performance improved. The best performance was achieved with only 2 layers in each BERT Model. However, with an F1-score of 0.6618, it still lags behind the simplest BERTbase model without any enrichments.

6 Discussion and Conclusion

We present a classification model for scholarly publications that achieves a weighted F1-Score of 0.7415 in the FoRC shared task I. When considering the evaluation results, it is noteworthy that even with a fine-tuned BERTbase model without any enrichments, a remarkable weighted F1 of 0.6868 is achieved. This demonstrates the robustness and versatility of the BERTbase model. However, the models optimized for scholarly publications perform noticeably better, yet these models are almost equivalent without enrichments. By enriching from various sources, we were able to further boost the classification performance, with SPECTER2 showing a better response to these enrichments compared to other models. To appropriately contextualize and evaluate our results, a comparison with human curators would be valuable. The provided dataset represents reality in a highly simplified manner. In our dataset, there is only one target label per item. Thus, in some cases, there are numerous alternative categorizations that deviate from the provided solution but are nonetheless not incorrect due to thematically overlapping categories. Instead of providing a single correct label, training the model with multiple possible classes per item could lead to a more nuanced prediction. Additionally the dataset is highly unbalanced, with a strong emphasis on "Physical Sciences and Mathematics", leading to a bias in the model. Training with hundreds or thousands of samples greatly enhances class efficacy compared to having only a few (e.g., "Art and Humanities"), which limits the transferability of our trained model. The approach of a two-stage TwinBERT, based on theoretical considerations, failed to deliver significant advantages in practical application. Although we applied the same methodology to compute the evaluation results, we were unable to reproduce the competition's outcomes, leading to a weighted F1-score approximately 0.008 lower.

Our current approach might potentially achieve even better results by incorporating a dedicated SPECTER2 adapter specifically designed for classification tasks. We also intend to explore a sliding window approach, addressing the input limitations of BERT [10,13], to enable the model to leverage the full scope of enriched metadata and full-text. Our source code, as well as the trained models are available for reuse[10] under the MIT-License.

[10] https://github.com/foerstner-lab/NSLP2024-FoRC.

References

1. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. arXiv preprint arXiv:1903.10676 (2019)
2. Bornmann, L., Haunschild, R., Mutz, R.: Growth rates of modern science: a latent piecewise growth curve approach to model publication numbers from established and new literature databases. Humanit. Soc. Sci. Commun. **8**(1), 1–15 (2021)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Garcia-Silva, A., Gomez-Perez, J.M.: Classifying scientific publications with BERT - is self-attention a feature selection method? In: Hiemstra, D., Moens, M.-F., Mothe, J., Perego, R., Potthast, M., Sebastiani, F. (eds.) ECIR 2021. LNCS, vol. 12656, pp. 161–175. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72113-8_11
5. Gombert, S.: Twin BERT contextualized sentence embedding space learning and gradient-boosted decision tree ensembles for scene segmentation in German literature. In: STSS@ KONVENS, pp. 42–48 (2021)
6. Hoy, M.B.: Rise of the Rxivs: how preprint servers are changing the publishing process. Med. Ref. Serv. Q. **39**(1), 84–89 (2020)
7. Lu, W., Jiao, J., Zhang, R.: TwinBERT: distilling knowledge to twin-structured compressed BERT models for large-scale retrieval. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2645–2652 (2020)
8. Ostendorff, M., Bourgonje, P., Berger, M., Moreno-Schneider, J., Rehm, G., Gipp, B.: Enriching BERT with knowledge graph embeddings for document classification. arXiv preprint arXiv:1909.08402 (2019)
9. Ostendorff, M., Rethmeier, N., Augenstein, I., Gipp, B., Rehm, G.: Neighborhood contrastive learning for scientific document representations with citation embeddings. arXiv preprint arXiv:2202.06671 (2022)
10. Pappagari, R., Żelasko, P., Villalba, J., Carmiel, Y., Dehak, N.: Hierarchical transformers for long document classification. Technical report arXiv:1910.10781 (2019)
11. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
12. Singh, A., D'Arcy, M., Cohan, A., Downey, D., Feldman, S.: SciRepEval: a multi-format benchmark for scientific document representations. arXiv preprint arXiv:2211.13308 (2022)
13. Wang, W., Yan, M., Wu, C.: Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In: Gurevych, I., Miyao, Y. (eds.) Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1705–1714. Association for Computational Linguistics, Melbourne (2018). https://doi.org/10.18653/v1/P18-1158. https://aclanthology.org/P18-1158

Shared Task: SOMD

SOMD@NSLP2024: Overview and Insights from the Software Mention Detection Shared Task

Frank Krüger[1(✉)], Saurav Karmakar[2], and Stefan Dietze[2]

[1] Wismar University of Applied Sciences, Wismar, Germany
frank.krueger@hs-wismar.de

[2] GESIS – Leibniz-Institut für Sozialwissenschaften, Köln, Germany
{saurav.karmakar,stefan.dietze}@gesis.org

Abstract. Software is a central part of the scientific process and involved in obtaining, analysing, visualising and processing research data. Understanding the provenance of research requires an understanding of the involved software. However, software citations in scientific publications often are informal, what creates challenges when aiming at understanding software adoption. This paper provides an overview of the Software Mention Detection (SOMD) shared task conducted as part of the 2024 Natural Scientific Language Processing Workshop, aiming at advancing the state-of-the-art with respect to NLP methods for detecting software mentions and additional information in scholarly publications. The SOMD shared task encompasses three subtasks, concerned with software mention recognition (subtask I), recognition of additional information (subtask II) and classification of involved relations (subtask III). We present an overview of the tasks, received submissions and used techniques. The best submissions achieved F1 scores of 0.74 (subtask I), 0.838 (subtask II) and 0.911 (subtask III) indicating both task feasibility but also potential for further performance gains.

Keywords: scholarly information processing · software mention extraction · software metadata identification · information extraction · relation classification

1 Introduction

Science across all disciplines has become increasingly data-driven, leading to additional needs with respect to software for collecting, processing and analysing data. Hence, transparency about software used as part of the scientific process is crucial to ensure reproducibility and to understand provenance of individual research data and insights. Knowledge about the particular version or software development state is a prerequisite for reproducibility of scientific results as even minor changes to the software might impact them significantly.

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 247–256, 2024.
https://doi.org/10.1007/978-3-031-65794-8_17

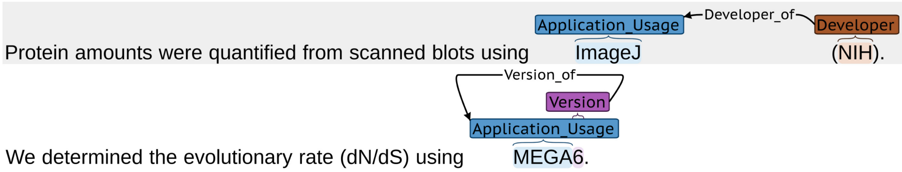

Fig. 1. Annotated sentences from SoMeSci. (Image taken from [18])

Furthermore, from a macro-perspective, understanding software usage, varying citation habits and their evolution over time within and across distinct disciplines can shape the understanding of the evolution of scientific disciplines, the varying influence of software on scientific impact and the emerging needs for computational support within particular disciplines and fields. Initial efforts are made to provide publicly accessible datasets that link open access articles to respective software that is used and cited, for instance, the OpenAIRE Knowledge Graph [10] or SoftwareKG [20]. Given the scale and heterogeneity of software citations, robust methods are required, able to detect and disambiguate mentions of software and related metadata. Despite the existence of software citation principles [6,21], software mentions in scientific articles are usually informal and often incomplete [19]—information about the developer or the version are often missing entirely, see Fig. 1. Spelling variations and mistakes for software names, even common ones [20], increase the complexity of automatic detection and disambiguation. Training and evaluation of information extraction approaches require reliable ground truth data of sufficient size, raising the need for manually annotated gold standard corpora of software mentions.

With this shared task, we would like to advance the field of software mention detection, seeking novel methods that outperform the state-of-the-art on the provided three subtasks.

We use Codalab [14] as a platform to run all three competitions. Subtask I[1] received 22 registrations from participants, from which 10 results were submitted. In contrast, the more challenging subtasks II[2] and III[3] received 12 registrations and 11 registrations respectively, but for each, only 3 actual submissions were received.

The rest of the paper is structured as follows. Section 2 presents previous work related to SOMD in order to compare the presented systems to current research, Sect. 3 defines both subtasks along with the used evaluation metrics. In Sect. 4, we introduce the datasets and taxonomies used for both subtasks, delving into their construction methods. Section 5 showcases the results received from submissions of both subtasks, describing the system architectures when possible. Finally, Sect. 6 discusses those results along with their limitations, and provides concluding remarks.

[1] https://codalab.lisn.upsaclay.fr/competitions/16935.
[2] https://codalab.lisn.upsaclay.fr/competitions/16936.
[3] https://codalab.lisn.upsaclay.fr/competitions/16937.

2 Related Work

Most works concerned with recognition of software mentions in scientific articles apply manual analysis on small corpora in order to answer specific questions [4,12] or are limited to specific software [8,9]. Automatic methods, enabling large scale analysis, have been implemented by iterative bootstrapping [13] as well as machine learning on manually engineered rules [2]. However, both achieve only moderate performance. Extraction through deep learning with a Bi-LSTM-CRF [20] shows promise, but requires sufficient and reliable ground truth data which only recently became available. More recently, Schindler et al. [17] provided robust information extraction models based on SciBERT and trained on the SoMeSci corpus [18] for NER and classification outperforms state-of-the-art methods for software extraction by 5% points on average. A similar approach was taken by [5] to recognize software mentions across several million articles achieving Named Entity Recognition performances at a similar level. Given that performance of related works still widely varies and is far from robust, this shared task aims at advancing the field of software mention detection and disambiguation across various subtasks.

3 Tasks Description

Software is an important part of the scientific process and should therefore be recognized as first class citizen of research. Research Knowledge Graphs have recently been adopted to provide bibliographic data at scale that could be populated by automatic extraction of software mentions. Given the scale and heterogeneity of software citations, robust methods are required to detect and disambiguate mentions of software and related metadata. The SOftware Mention Detection in Scholarly Publications (SOMD) task utilises the SoMeSci - Software mentions in Science - corpus. Participants had the option to sign up for one or more subtasks. Automated evaluations of submitted systems are done through the Codalab platform.

- **Subtask I: Software Mention Recognition**: Software mentions are recognized from individual sentences. At the same time, software mentions had to be classified according to their mention type, e.g., mention, usage, or creation and their software type, e.g., application, programming environment, or package. Participants developed classifiers that take individual sentences from the different subsets of SoMeSci and output mentions of software further classified into their type of software and mention. Submissions were evaluated using the F1 score computed based on exact matches. Please note that subtask I deviates from the from the original publication in that it combines the identification of the software and the classification of mention and software type.
- **Subtask II: Additional Information**: For each software mention, additional information according to the SoMeSci schema shall be recognized from

the sentence. This includes information such as version, URL, and developer. Participants had to develop classifiers that take sentences with software mentions as input and identify all additional information within the sentence. As in Subtask I, submissions were evaluated through the F1 score based on exact matches.

– **Subtask III: Relation Classification**: For each software mention, relations to other recognized entities had to be classified. This includes versions and developers, but also URLs or host applications for plugins. The evaluation was based on exact matches rather than partial matches. F1 score had been used as an evaluation performance metric for all the subtasks.

4 Dataset

The shared tasks utilise SoMeSci-Software Mentions in Science - a gold standard knowledge graph of software mentions in scientific articles [18]. It contains high quality annotations (Inter-Rater Reliability, IRR: $\kappa = .82$) of 3756 software mentions in 1367 PubMed Central articles. Besides the plain mention of the software, it also provides relation labels for additional information, such as the version, the developer, a URL or citations and distinguishes between different types, such as application, plugin or programming environment, as well as different types of mentions, such as usage or creation. SoMeSci is the most comprehensive corpus about software mentions in scientific articles, providing training samples for Named Entity Recognition, Relation Extraction, Entity Disambiguation, and Entity Linking.

SoMeSci is created by manually annotating 3756 software mentions with additional information, resulting in 7237 labelled entities in 47,524 sentences from 1367 PubMed Central articles. Data is lifted into a knowledge graph (excerpt in Fig. 2) by using established vocabularies such as NLP Interchange Format (NIF) [3] and schema.org [16], disambiguated and linked to external resources, and shared as a publicly available 5-star Linked Open Data resource [1] that can be explored interactively.

In preparation of the three subtasks for the SOMD challenge, we released a new dataset implementing predefined splits for training and testing.

For each subtask, the same split was created, which also follows the original train-test split, as reported for SoMeSci [18], resulting in 39.768 sentences for train and 8180 sentences for testing for the first subtask.

The new dataset was released via Zenodo [7] and consists of three parts, one for each subtask. Each of the individual parts contains a list of sentences and labels for training and a list of sentences for testing. As subtask II and III require additional information for the test set, such as the already identified software mentions and their respective meta data, for both tasks, we provided an additional file with this information for train and test set respectively. As subtask II and III require already recognized software mentions, the number of sentences provided for these tasks reduced to 2353 (1091) for the train set and 374 (131) for the test set for subtask I (subtask II).

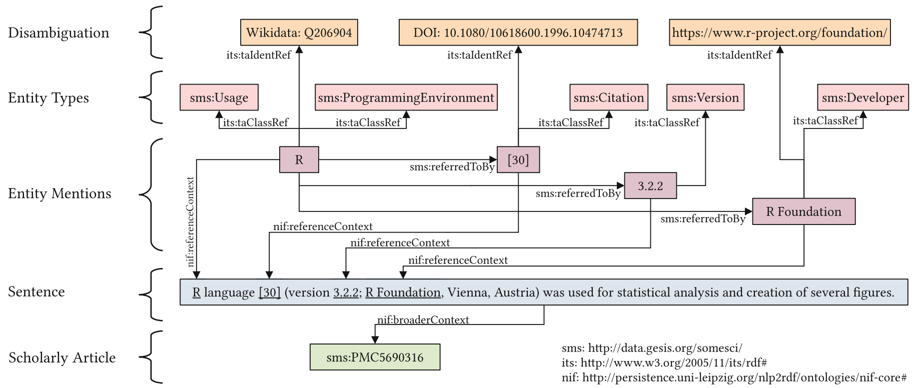

Fig. 2. Excerpt of the SoMeSci knowledge graph illustrating the textual references of software mentions and their version, developer and citation. The different levels of representation separate the main concerns of interest, natural language sentences, mentions of entities, their types and disambiguation to knowledge entities. For clarity, some information is omitted. (Image taken from [18])

Participants of the SOMD shared task were required to retrieve the dataset and use all available training data to establish a classifier to be tested on the provided test set. The prediction as created from the test set of the respective tasks was subject to submission at the Codalab platform. Finally, evaluation scripts implemented in Python by using the packages scikit-learn [15] and seqeval [11] were used to determined weighted Precision, Recall and F1 score, where the F1 score was used to rank the submissions.

5 Results

In this section we describe results reported as well the techniques and strategies adopted by different participants in the subtasks.

5.1 Subtask I

Overall, 23 participants registered for subtask I, from which we received nine submissions in total. Two of the submissions obtained F1 scores close to 0, resulting in seven valid submissions. An overview of the valid submissions including the achieved scores is provided in Table 1. As not all teams submitted system descriptions, in to following we focus on submissions that provided such descriptions. `phinx` achieved the highest F1 score, followed by `david-s477`, `ThuyNT03` and `ottowg`.

Team `phinx` experimented with different LLMs namely BloomZ, Mistral, Llama-2 and Jaskier-7b-dpo, where Jaskier-7b-dpo provided the best performance (F1 score of 74%). They further finetuned pre-trained models using the LoRA (Low-Rank Adaptation) [22] technique.

Table 1. Evaluation results of submissions for subtask I

User	F1 (weighted)	precision (weighted)	recall (weighted)
phinx	**0.740**	**0.761**	**0.750**
david-s477	0.692	0.739	0.711
ThuyNT03	0.678	0.729	0.649
ottowg	0.652	0.679	0.664
vampire	0.648	0.682	0.637
fddaFIT	0.504	0.548	0.499
dainb	0.483	0.709	0.392

Team ThuyNT03 experimented with BERT based models specifically XLM-Roberta, BERT and SciBERT. Utilizing each model, they experimented with three approaches: in their first approach they chose direct classification, whereas in the second approach, classification was split into two stages, where the first stage produced BIO tags and the second stage produced actual entity labels. The third approach conducted a three stage classification that included a preliminary step to detect if a sentence contains any entities before continuing with the two stage classification.

The team fddaFIT investigated the effectiveness of the decoder-only Falcon-7b model, which is known for its performance across a wide range of NLP tasks. They experimented with different sampling schemes like selective sampling and adaptive sampling to compose finetuning data. They also experimented with different strategies but that did not yield enhancements in outcome. To address the class imbalance they used a weighted loss mechanism (where class weights are inversely proportional to class frequencies) and adaptive sampling, i.e. oversampling the underrepresented data by a factor of 2 and undersampling the over-represented data to sizes equal to multiples (1, 1.5, 3) of the oversampled data volume.

Team ottowg employed SciBERT pre-trained model and they also experiment with generative large language models. Various prompting strategies were used by the team for the subtask. Retrieval-Augmented Generation (RAG) with LLM has been applied using Generative Language Models (GLMs), specifically GPT 3.5 and GPT 4 for the task. They used a pipeline strategy that prioritizes selecting relevant text passages for GLM analysis, improving efficiency by filtering out unrelated content. Their performance optimization employs a hybrid method, combining a fine-tuned NER model for sentence selection with GLMs for information extraction. Their best configuration achieved an F1 score of 0.679 for subtask I using a generative LLM.

5.2 Subtask II

Out of 12 registrations for subtask II, we received submissions by two teams, namely phinx and ottowg. Performance metrics for this subtask are reported in Table 2. The team `ottowg` achieves the top performance with an F1 score of 0.838, whereas the `phinx` team achieves 0.743.

For this subtask, the `ottowg` team adopted an approach similar to the approach of subtask I, but tuned for the extraction of associated software attributes. They utilised a retrieval mechanism to augment the task description in a few shot setup. For each sample, including those derived from few-shot learning, the process entailed presenting the sentence containing the software entity(ies) and then predicting a JSON list of identified entities.

The `phinx` team followed the same experimentation model as in subtask I with LLMs with modifications to the prompt engineering to accommodate the additional information for the software such as as version, URL, and developer etc. With multiple experimentation with various LLMs, they achieved 0.743 in F1 score as their best performing approach for this subtask using the Jaskier-7b-dpo model.

Table 2. Evaluation results of submissions for subtask II

User	F1 (weighted)	precision (weighted)	recall (weighted)
ottowg	**0.838**	**0.835**	**0.847**
phinx	0.743	0.745	0.748

5.3 Subtask III

Like the previous subtask, we received two submissions for subtask III by the same two same teams `phinx` and `ottowg`. Table 3 depicts the results of their submission with `ottowg` scored the best followed by `phinx` team.

For this subtask, the study for `ottowg` proposed a novel approach by conceptualizing the task of relation extraction as a single-choice question-answering (QA) activity. This method entailed generating a comprehensive list of all possible entities within a sentence, drawing from the existing entities and their relationships as delineated in the training dataset. Each potential pair of entities was then evaluated to ascertain if a specific relation attribute types. These questions were then presented to a Large Language Model for answering.

The `phinx` team followed the same experimentation model as they applied for their earlier subtasks here with LLMs for modifications to the prompt engineering to accommodate the relations to other recognized entities which includes versions and developers, but also URLs or host applications for plugins. With multiple experimentation with various LLMs; again the Jaskier-7b-dpo model proved best for them by bringing 0.897 in F1 score as their best for this subtask.

Table 3. Evaluation results of submissions for subtask III

User	F1 (weighted)	precision (weighted)	recall (weighted)
ottowg	**0.911**	**0.924**	**0.916**
phinx	0.897	0.904	0.897

6 Conclusion

In this paper, we presented an overview of the *Software Mention detection (SOMD)* shared task, that was run as part of the *2024 Natural Scientific Language Processing Workshop*, in conjunction with the Extended Semantic Web Conference 2024 (ESWC2024). The task is the first of its kind, proposing a set of three subtasks concerned with the detection of software mentions and related attributes in scholarly publications together with benchmark datasets and baselines. Given the important role of used software in the scientific process, understanding software citations is a crucial factor towards reproducibility of scientific works. This shared task provides the basis for advancing research into detecting and disambiguating software mentions. Unsurprisingly, the submissions to a large extent adopted various kinds of pre-trained language models as starting point for their pipelines. However, the diversity of submissions documented the range of techniques that can facilitate performance gains, starting from different base model choices, retrieval augmented approaches, sampling techniques or the use of prompt engineering as part of intermediate steps.

References

1. Berners-Lee, T.: Is your linked open data 5 star? (2010). http://www.w3.org/DesignIssues/LinkedData#fivestar
2. Duck, G., Nenadic, G., Filannino, M., Brass, A., Robertson, D.L., Stevens, R.: A survey of bioinformatics database and software usage through mining the literature. PloS One **11**(6), 1–25 (2016). https://doi.org/10.1371/journal.pone.0157989
3. Hellmann, S., Lehmann, J., Auer, S., Brümmer, M.: Integrating NLP using linked data. In: Alani, H., et al. (eds.) ISWC 2013. LNCS, vol. 8219, pp. 98–113. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41338-4_7
4. Howison, J., Bullard, J.: Software in the scientific literature: problems with seeing, finding, and using software mentioned in the biology literature. J. Am. Soc. Inf. Sci. **67**(9), 2137–2155 (2016)
5. Istrate, A.M., Li, D., Taraborelli, D., Torkar, M., Veytsman, B., Williams, I.: A large dataset of software mentions in the biomedical literature (2022).https://doi.org/10.48550/ARXIV.2209.00693
6. Katz, D., et al.: Recognizing the value of software: a software citation guide. F1000Research **9**, 1257 (2021).https://doi.org/10.12688/f1000research.26932.2
7. Krüger, F.: SOMD - SOftware Mention Detection (2024). https://doi.org/10.5281/zenodo.10472161
8. Li, K., Lin, X., Greenberg, J.: Software citation, reuse and metadata considerations: an exploratory study examining LAMMPS. Proc. Assoc. Inf. Sci. Technol. **53**(1), 1–10 (2016)

9. Li, K., Yan, E., Feng, Y.: How is R cited in research outputs? Structure, impacts, and citation standard. J. Informet. **11**(4), 989–1002 (2017)
10. Manghi, P., et al.: The OpenAIRE research graph data model (2019). https://doi.org/10.5281/ZENODO.2643199
11. Nakayama, H.: seqeval: a python framework for sequence labeling evaluation (2018). https://github.com/chakki-works/seqeval
12. Nangia, U., Katz, D.S.: Understanding software in research: initial results from examining nature and a call for collaboration. In: 2017 IEEE 13th International Conference on e-Science (e-Science), pp. 486–487. IEEE (2017)
13. Pan, X., Yan, E., Wang, Q., Hua, W.: Assessing the impact of software on science: a bootstrapped learning of software entities in full-text papers. J. Informet. **9**(4), 860–871 (2015)
14. Pavao, A., et al.: CodaLab competitions: an open source platform to organize scientific challenges. J. Mach. Learn. Res. **24**(198), 1–6 (2023). http://jmlr.org/papers/v24/21-1436.html
15. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
16. Ronallo, J.: Html5 microdata and schema. org. Code4Lib J. (16) (2012)
17. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: The role of software in science: a knowledge graph-based analysis of software mentions in PubMed Central. PeerJ Comput. Sci. **8**, e835 (2022)
18. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: SoMeSci-A 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21). Association for Computing Machinery, Virtual Event, QLD, Australia (2021). https://doi.org/10.1145/3459637.3482017
19. Schindler, D., Hossain, T., Spors, S., Krüger, F.: A multi-level analysis of data quality for formal software citation. Quant. Sci. Stud., 1–31 (June 2024). https://doi.org/10.48550/arXiv.2306.17535
20. Schindler, D., Zapilko, B., Krüger, F.: Investigating software usage in the social sciences: a knowledge graph approach. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 271–286. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_16
21. Smith, A.M., Katz, D.S., Niemeyer, K.E.: Software citation principles. PeerJ Comput. Sci. **2**, e86 (2016). https://doi.org/10.7717/peerj-cs.86
22. Yu, Y., et al.: Low-rank adaptation of large language model rescoring for parameter-efficient speech recognition. In: 2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), pp. 1–8. IEEE (2023)

Software Mention Recognition with a Three-Stage Framework Based on BERTology Models at SOMD 2024

Thuy Nguyen Thi[1,2], Anh Nguyen Viet[1,2], Thin Dang Van[1,2(✉)], and Ngan Luu-Thuy Nguyen[1,2]

[1] University of Information Technology - VNUHCM, Ho Chi Minh City, Vietnam
{21521514,19521204}@gm.uit.edu.vn, ngannlt@uit.edu.vn
[2] Vietnam National University, Ho Chi Minh City, Vietnam
thindv@uit.edu.vn

Abstract. This paper describes our systems for the sub-task I in the Software Mention Detection in Scholarly Publications shared-task. We propose three approaches leveraging different pre-trained language models (BERT, SciBERT, and XLM-R) to tackle this challenge. Our best-performing system addresses the named entity recognition (NER) problem through a three-stage framework. (1) Entity Sentence Classification - classifies sentences containing potential software mentions; (2) Entity Extraction - detects mentions within classified sentences; (3) Entity Type Classification - categorizes detected mentions into specific software types. Experiments on the official dataset demonstrate that our three-stage framework achieves competitive performance, surpassing both other participating teams and our alternative approaches. As a result, our framework based on the XLM-R-based model achieves a weighted F1-score of 67.80%, delivering our team the 3rd rank in Sub-task I for the Software Mention Recognition task. We release our source code at this repository (https://github.com/thuynguyen2003/NER-Three-Stage-Framework-for-Software-Mention-Recognition).

Keywords: Software mention recognition · Named entity recognition · Transformer · Three-stage framework

1 Introduction

Named Entity Recognition (NER) is an important task in NLP that involves identifying and classifying named entities in text. That will transform them into structured data, making it easier to categorize and perform search processing or carry out other NLP tasks [5] on that data such as text classification, sentiment analysis, and contextual analysis, particularly in the domain of Biomedical Named Entity Recognition (Bio-NER), which is challenged by a range of entities like genes, proteins, medications, and diseases [9].

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 257–266, 2024.
https://doi.org/10.1007/978-3-031-65794-8_18

The SOMD 2024 shared-task, hosted within Natural Scientific Language Processing and Research Knowledge Graphs (NSLP 2024) workshop [8], is designed to extract mentioned software and metadata from documents. In this context, both the software and the metadata are identified as specific intervals in the original documents. Understand and identify the software mentioned in documents, which is especially important to support information extraction in scientific documents.

In this paper, we present three different approaches to address the challenge of sub-task I, including:

- **Approach 1**: Fine-tuning pre-trained language models as a token classification problem.
- **Approach 2**: Two-stage framework for entity extraction and classification.
- **Apporach 3**: Three-stage framework for entity sentence classification, entity extraction, and entity type classification.

2 Related Work

In recent years, pre-training language models (PLMs) have made significant advancements in Named Entity Recognition (NER) tasks [16]. Among these, the most popular model is BERT [7] and its variations like SciBERT [2], RoBERT [3], and BiLSTM [11]. These models are often paired with machine learning techniques, particularly Conditional Random Fields (CRF) [10]. Additionally, some approaches involve breaking down the NER task into two simpler tasks using question-answering methods [1], achieving notable results on various datasets like BioNLP13CG, CTIReports, OntoNotes5.0 [12], and WNUT17 [6] based on the F1 measure.

With the emergence of ChatGPT, researchers have been exploring the use of Large Language Models (LLMs) for NER tasks [15,17], with some studies demonstrating that ChatGPT can be distilled into smaller UniversalNER models for open NER [18]. These UniversalNER models have shown exceptional accuracy across 43 datasets spanning diverse fields such as biomedicine, programming, social media, law, and finance, without requiring direct supervision. UniversalNER surpasses traditional guideline-tuned models like Alpaca and Vicuna by an average of over 30 F1 points and achieves a high F1 score of 0.8 on SoMeSci. In this paper, BERT, SciBERT, and XML-R models are still utilized to address the first task of the shared SOMD 2024 challenge.

3 Approach

To address the Software Mention Recognition task, we utilize the power of different pre-trained transformer-based language model in different approaches. Figure 1 illustrates three approaches to participate in the competition. Because shared-task is related to each token in the sentence and whether words are in capital letters or not also greatly affects the recognition of entities. Therefore,

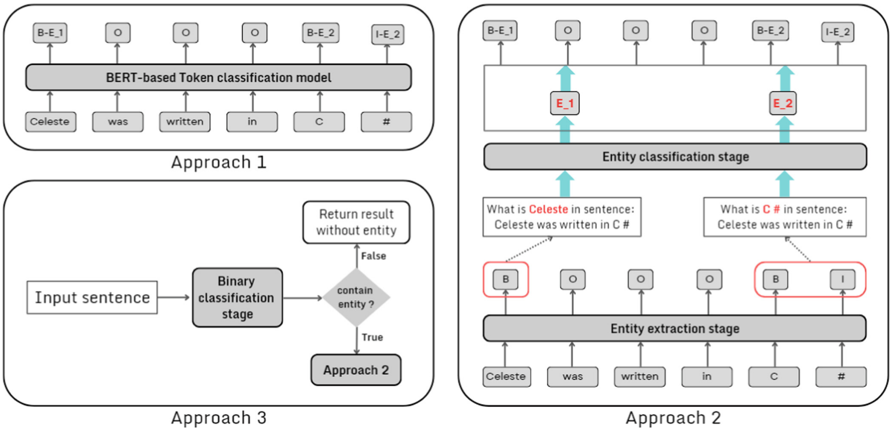

Fig. 1. Overview system of three approaches: Sample input is "Celeste was written in C #" with two entities are E_1 and E_2. E_1 and E_2 play the role of two entity types in this example

we do not apply any preprocessing techniques but use data directly from the organizers. Also the tokenize method will depend on the default tokenier of the models. In our work, we employ various pre-trained language models, including the XLM-Roberta (XLM-R) [4], BERT [7], and SciBERT [2] as our main backbones. The detail of our three approaches are present as follow.

3.1 Approach 1: Token Classification with BERTs

For the first approach, we address the task by fine-tuning different transformer BERT-base models for the token classification task. We adapted different pre-trained language models to the training dataset. After tokenizing the input, we feed the token sequence to backbones models to extract the fixed vector in the last layer as the final representation of the input sentence. Then, we apply a fully connected layer to process the vectors and predict labels for each input token using a softmax function. There are a total of 27 labels (in Table 1), where 26 correspond to 13 different entity types, and one label represents non-entities. Figure 1 illustrates the overview of our first approach.

3.2 Approach 2: Two-Stage Framework for Entity Extraction and Classification

Motivated by recent work by [1], we address Task 1 - Software Mention Recognition with a two-stage framework composed of entity extraction and entity classification components. However, our components are re-designed to improve the overall performance than original framework proposed by [1]. Figure 1 illustrates the overview of this approach, the detail of each component is presented below:

Table 1. List of labels for token classification task in Approach 1

Index	Label	Index	Label
1	B-Application_ Creation	15	B-PlugIn_Deposition
2	I-Application_ Creation	16	I-PlugIn_Deposition
3	B-Application_Deposition	17	B-PlugIn_Mention
4	I-Application_Deposition	18	I-PlugIn_Mention
5	B-Application_Mention	19	B-PlugIn_Usage
6	I-Application_Mention	20	I-PlugIn_Usage
7	B-Application_Usage	21	B-ProgrammingEnvironment_Mention
8	I-Application_Usage	22	I-ProgrammingEnvironment_Mention
9	B-OperatingSystem_Mention	23	B-ProgrammingEnvironment_Usage
10	I-OperatingSystem_Mention	24	I-ProgrammingEnvironment_Usage
11	B-OperatingSystem_Usage	25	B-SoftwareCoreference_Deposition
12	I-OperatingSystem_Usage	26	I-SoftwareCoreference_Deposition
13	B-PlugIn_Creation	27	O
14	I-PlugIn_Creation		

- **Stage 1 - Entity extraction**: This stage aims to identify whether each token in a given input sentence belongs to an entity or not. We achieve this through token classification, similar to Approach 1. However, instead of using 27 labels for different token types, we only use 3 labels as:
  - **O**: Non-entity token
  - **B-X**: Beginning token of an entity of type X (where X represents one of the 13 entity types)
  - **I-X**: Token within an entity of type X

  Using separate labels for the beginning (B) and inside (I) positions of tokens within an entity allows us to efficiently extract all words belonging to the same entity in stage 2.
- **Stage 2 - Entity classification:** In this stage, we classify the detected entities from stage 1. We use a classifier with 13 labels corresponding to the 13 entity types, discarding the B-I prefix distinction used for token position. This classifier is built by fine-tuning a transformer-based model like BERT. [14] During fine-tuning for classification tasks, it's common practice to use the hidden state associated with the [CLS] token as input for a classifier. However, in this approach, we fine-tune the entire transformer model end-to-end. This means the hidden states are not treated as fixed features, but are trained alongside the classification head (a component added on top of the pre-trained model) for optimal performance. Additionally, to leverage the knowledge of transformer models, we format this classifier as a question-and-answering model by constructing the input as the following prompt:
  - **Input**: What is <entity> in the sentence: <input sentence>
  - **Output**: Type of entity

Table 2. General statistics in the training set and private test set

Information	Training set	Private test set
#Sentence	39768	8180
#Sentence with entity	2353	374
Total entity	3241	515
Max length	568	347
Avg length	28.32	28.82

3.3 Approach 3: Three-Stage Framework

Our analysis in Table 2 revealed a limited number of sentences containing entities within the training set. This disparity raised concerns about potential biases in the label information during the training process for the previously mentioned approaches. To address this, we introduce a new three-stage framework, which integrate a binary classification with Approach 2. We simply built a binary classification model to detect the sentences which contain the entity. As shown in Fig. 1, if a sentence is classified as class 0, assign all tokens in the sentence as O, otherwise, this sentence will be passed to Approach 2 to extract the entity and its type.

4 Experimental Setup

4.1 Data and Evaluation Metrics

This shared-task uses the SoMeSci dataset [13] which included 39768 sentences and 3756 software mentions divided into a training set and a private test set. We train our systems only on the training set and evaluate the performance of our model on the private test set using weighted precision, recall, and F1-score. In Table 2, we summarize some general information about the two data sets. Where #Sentence denotes the number of sentences, #Sentence with entity denotes the number of sentences containing the entity, and Total entity is the total of entities in all sentences. Max length and Avg length are the maximum length and average length of the sentences in each set, respectively. This dataset contains six groups of entity Application, OperatingSystem, PlugIn, ProgrammingEnvironment, and SoftwareConference. Each group can have the entity belong to four types [Creation, Deposition, Mention, Usage]. In Table 3 we indicate the distribution of each entity in the dataset

4.2 System Settings

We conduct all experiments on three approaches, using three base-version backbones: XLM-R[1], BERT[2], and SciBERT[3]. We loaded the weights of the backbones from the HuggingFace library and carried out training on an NVIDIA

[1] https://huggingface.co/FacebookAI/xlm-roberta-base.
[2] https://huggingface.co/google-bert/bert-base-uncased.
[3] https://huggingface.co/allenai/scibert-scivocab-uncased.

Table 3. Statistics the number of entities in each entity type entity in each entity group in the training set and private test set

Entity group	Entity	Training set		Testing set	
		Quantity	Total	Quantity	Total
Application	Application_Creation	150	2353	47	348
	Application_Deposition	80		22	
	Application_Mention	162		31	
	Application_Usage	1958		248	
OperatingSystem	OperatingSystem_Mention	13	140	17	33
	OperatingSystem_Usage	127		16	
PlugIn	PlugIn_Creation	53	344	17	81
	PlugIn_Deposition	21		8	
	PlugIn_Mention	40		11	
	PlugIn_Usage	230		45	
ProgrammingEnvironment	ProgrammingEnvironment_Mention	41	372	6	49
	ProgrammingEnvironment_Usage	331		43	
SoftwareCoreference	SoftwareCoreference_Deposition	35	35	4	4

T4(x2) GPU provided by Kaggle. The corresponding hyper-parameters for each approach are presented below:

- **Approach 1:** batch size = 32, learning rate = 5e−05, and the number of epoch = 25 with XLM-R model and the number of epoch = 20 both remain backbones.
- **Approach 2:**
  - Stage 1: batch size = 32, learning rate = 5e−05 and the number of epoch = 20 for all three backbones.
  - Stage 2: batch size = 16, learning rate = 2e−05 and the number of epoch = 25 with XLM-R model and epoch = 20 two remainder models.
- **Approach 3:**
  - Stage 1: batch size = 32, learning rate = 2e−5 and the number of epoch = 10 for all three backbones.
  - Stage 2 and Stage 3: Using the configuration and architecture as the Approach 2.

5 Main Results

According to the organizing committee, this sub-task will be evaluated by F1-Score based on exact matches. As shown in Table 4, we provide a tabulated summary of 9 experiments, each representing one of the 9 final systems generated from three different approaches and using three distinct backbones.

The experimental results in Table 4 indicate that Approach 3, a three-stage system, demonstrates the best performance across all backbones, with the XLM-RoBERTa backbone exhibiting the highest efficacy among all approaches. However, this result is for reference only and is only true in all of my experiments.

Table 4. Comparative performance of our three Approaches with different pre-trained language models on the test set.

Models	Approach 1			Approach 2			Approach 3		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
BERT	0.675	0.594	0.625	0.682	0.643	0.653	0.690	0.629	0.650
SciBERT	0.658	0.621	0.623	0.719	0.645	0.670	0.736	0.631	0.670
XLM-R	0.716	0.614	**0.649**	0.707	0.654	**0.671**	0.729	0.649	**0.678**

Table 5. Official scoreboard (https://codalab.lisn.upsaclay.fr/competitions/16935#results) for the sub-task I: Software mention recognition.

Participant	Ranking	Evaluation metrics		
		Precision	Recall	F1-score
phinx	Top 1	**0.761**	**0.750**	**0.740**
david-s477	Top 2	0.739	0.711	0.692
ottowg	Top 4	0.679	0.664	0.652
vampire	Top 5	0.682	0.637	0.648
Our best system	Top 3	0.729	0.649	0.678

It's important to acknowledge that different contexts, set up or datasets might yield different outcomes, and we are not sure this is the best result that each backbone could give in other cases. Finally, the best system was built according to approach 3 with XLM-R backbone and our best submission was ranked 3rd. Table 5 show the final score of the top 5 participants.

With the test dataset labels provided by the organizing committee, we evaluated the performance of our best system for each entity class in Table 6. We observed that the SoftwareCoreference_Deposition entity achieved the highest Precision score, while the ProgrammingEnvironment_Usage entity attained the highest Recall and F1 score, top 5 F1-score classes are ProgrammingEnvironment_Usage, SoftwareCoreference_Deposition, and OperatingSystem_Mention. It is evident that entities belonging to the PlugIn group typically scored lower than those in other groups shows that it has difficulty in the regconization process. Although, the number of PlugIn_Usage entities in the training set is pretty large the result on the test set is not positive. Besides that, PlugIn_Creation and PlugIn_Deposition entities have the sample in the training set are pretty low and their score moves forward to zero. The number of OperatingSystem_Mention entities in the training set is low and the score on the test set is high so we predict the mention entity type in this group is featured and easier to recognize than other groups.

Additionally, in Table 7, we evaluated each individual stage in our final three-stage system by assuming that the accuracy of the stages before it is 100%. The first stage works well with an F1-score of 0.992 in classifying whether a sentence contains an entity or not. Moving to stage 2, tasked with detecting entities in sentences, achieved an F1 score at a relatively good level, but a significant

Table 6. Performance of the final system on the test dataset across entity classes evaluated by Precision, Recall, and F1-score.

Entity class	Precision	Recall	F1-score
Application_Creation	0.692	0.766	0.727
Application_Deposition	0.615	0.727	0.667
Application_Mention	0.560	0.452	0.500
Application_Usage	0.812	0.730	0.769
OperatingSystem_Mention	0.867	0.765	0.812
OperatingSystem_Usage	0.579	0.688	0.629
PlugIn_Creation	0.200	0.059	0.091
PlugIn_Deposition	0.000	0.000	0.000
PlugIn_Mention	0.667	0.364	0.471
PlugIn_Usage	0.682	0.333	0.448
ProgrammingEnvironment_Mention	0.500	0.167	0.250
ProgrammingEnvironment_Usage	0.886	**0.907**	**0.897**
SoftwareCoreference_Deposition	**1.000**	0.750	0.857

Table 7. Performance of components in our final three-stage framework.

Stage	Precision	Recall	F1-score
Stage 1	0.992	0.992	0.992
Stage 2	0.912	0.786	0.845
Stage 3	0.786	0.806	0.784

difference between Precision and Recall (12.6% difference) is evident, which also affects the overall system performance. In the final stage, the scores between the three metrics are relatively balanced, but it appears that the task of classifying 13 entity classes had some impact on this stage with relatively lower overall performance. The propagation of errors between the three stages has a significant impact on the entire system, with the final F1-score of the entire system being 0.678.

6 Conclusion and Future Work

In this paper, we present and evaluate three approaches for tackling sub-task I in the Software Mention Detection in Scholarly Publications shared task. While we explored the use of suitable transformer models like BERT, our three-stage system leveraging the XLM-R model achieved the highest performance in the competition. As a result, our best system achieved the Top 3 in the private test. In future work, our intention is to analyze the error propagation between the three stages to enhance the performance of the entire three-stage system. Additionally, with access to more substantial computational resources, we aim to experiment with fine-tuning sub-tasks using larger batch sizes and epochs for each backbone in order to investigate the effects of these hyper-parameters on the model's performance.

Acknowledgements. This research was supported by The VNUHCM-University of Information Technology's Scientific Research Support Fund. We also thank the anonymous reviewers for their valuable comments on our manuscript.

References

1. Arora, J., Park, Y.: Split-NER: named entity recognition via two question-answering-based classifications. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Toronto, Canada, pp. 416–426. Association for Computational Linguistics (2023). https://doi.org/10.18653/v1/2023.acl-short.36. https://aclanthology.org/2023.acl-short.36
2. Beltagy, I., Lo, K., Cohan, A.: SciBERT: a pretrained language model for scientific text. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, pp. 3615–3620. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/D19-1371. https://aclanthology.org/D19-1371
3. Chen, T., et al.: RoBERT-Agr: an entity relationship extraction model of massive agricultural text based on the RoBERTa and CRF algorithm. In: 2023 IEEE 8th International Conference on Big Data Analytics (ICBDA), pp. 113–120. IEEE (2023)
4. Conneau, A., et al.: Unsupervised cross-lingual representation learning at scale. In: Jurafsky, D., Chai, J., Schluter, N., Tetreault, J. (eds.) Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 8440–8451. Association for Computational Linguistics (2020). https://doi.org/10.18653/v1/2020.acl-main.747. https://aclanthology.org/2020.acl-main.747
5. Dash, A., Darshana, S., Yadav, D.K., Gupta, V.: A clinical named entity recognition model using pretrained word embedding and deep neural networks. Decis. Anal. J. **10**, 100426 (2024)
6. Derczynski, L., Nichols, E., van Erp, M., Limsopatham, N.: Results of the WNUT2017 shared task on novel and emerging entity recognition. In: Proceedings of the 3rd Workshop on Noisy User-Generated Text, Copenhagen, Denmark, pp. 140–147. Association for Computational Linguistics (2017). https://doi.org/10.18653/v1/W17-4418. https://www.aclweb.org/anthology/W17-4418
7. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics (2019). https://doi.org/10.18653/v1/N19-1423. https://aclanthology.org/N19-1423
8. Simperl, E., Peter Clark, K.K.: Natural scientific language processing and research knowledge graphs. In: Lecture Notes in Artificial Intelligence (2024)

9. Li, L., Zhou, R., Huang, D.: Two-phase biomedical named entity recognition using CRFs. Comput. Biol. Chem. **33**(4), 334–338 (2009)
10. Lopez, P., Du, C., Cohoon, J., Ram, K., Howison, J.: Mining software entities in scientific literature: document-level NER for an extremely imbalance and large-scale task. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 3986–3995 (2021)
11. Luo, L., et al.: An attention-based BiLSTM-CRF approach to document-level chemical named entity recognition. Bioinformatics **34**(8), 1381–1388 (2018)
12. Pradhan, S., et al.: Towards robust linguistic analysis using OntoNotes. In: Proceedings of the Seventeenth Conference on Computational Natural Language Learning, Sofia, Bulgaria, pp. 143–152. Association for Computational Linguistics (2013). https://aclanthology.org/W13-3516
13. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: SoMeSci-A 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4574–4583 (2021)
14. Tunstall, L., Von Werra, L., Wolf, T.: Natural Language Processing with Transformers: Building Language Applications With Hugging Face. O'Reilly (2022)
15. Wang, S., et al.: GPT-NER: named entity recognition via large language models. arXiv preprint arXiv:2304.10428 (2023)
16. Zhang, H., et al.: Samsung research China-Beijing at SemEval-2023 Task 2: an AL-R model for multilingual complex named entity recognition. In: Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023), pp. 114–120 (2023)
17. Zhang, Z., Zhao, Y., Gao, H., Hu, M.: LinkNER: linking local named entity recognition models to large language models using uncertainty. arXiv preprint arXiv:2402.10573 (2024)
18. Zhou, W., Zhang, S., Gu, Y., Chen, M., Poon, H.: UniversalNER: targeted distillation from large language models for open named entity recognition. In: The Twelfth International Conference on Learning Representations (2023)

ABCD Team at SOMD 2024: Software Mention Detection in Scholarly Publications with Large Language Models

Phi Nguyen Xuan[1,2], Quang Tran Minh[2], and Thin Dang Van[3,4](✉)

[1] HCMC University of Technology and Education, Ho Chi Minh City, Vietnam
20134004@student.hcmute.edu.vn
[2] Bosch Global Software Technologies Vietnam, Ho Chi Minh City, Vietnam
quang.tranminh@vn.bosch.com
[3] University of Information Technology - VNUHCM, Ho Chi Minh City, Vietnam
thindv@uit.edu.vn
[4] Vietnam National University, Ho Chi Minh City, Vietnam

Abstract. This paper outlines the ABCD team's approach to employing LLMs for three subtasks at SOMD 2024, specifically focusing on Software Mention Detection in Scholarly Publications. The task revolves around scientific articles and information, comprising three subtasks: (1) extracting software entities from a given sentence, (2) extracting relevant entities related to software entities from the task I, and (3) determining the relationship between entities extracted from the previous two tasks. Our objective is to gain valuable insights into fine-tuning LLMs using LoRA. The experimental results showcase that our approach has demonstrated competitive performance across all three tasks, securing Top 1, Top 2, and Top 2 rankings for Subtask I, Subtask II, and Subtask III, respectively. We release our source code in this Github repository(https://github.com/Xphi310302/ABCD-team-NLSP.git).

Keywords: SOMD 2024 · Software Mention Detection in Scholarly Publications · Large Language Model · LoRA · Fine-tuning LLM · Prompting Engineering

1 Introduction

Named Entity Recognition (NER) is the task of identifying and classifying entities such as people, locations, and organizations within a piece of text. While most NER datasets, like CoNLL2003 [14], have focused on high-resource languages like English and specific domains like news, the recently shared task on Software Mention Detection in Scholarly Publications (SOMD) [15] indicated the growing importance of software in research. With the rise of Research Knowledge Graphs, there is a need for efficient extraction of bibliographic data, particularly through automated software mention detection. The Software Mention Detection (SOMD) shared task aims to tackle this problem by offering three different sub-tasks as below:

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 267–277, 2024.
https://doi.org/10.1007/978-3-031-65794-8_19

- **Subtask I: Software NER**: This task involves identifying four types of software-related entities: Application, Plugin, Operating System, and Programming Environment. Accurately classifying software mentions is necessary to understand their role in academic discourse.
- **Subtask II: Attribute NER**: This task aims to extract ten different attributive information associated with software entities, such as alternative names, abbreviations, authorship (developers), software release maintenance (versions, extensions, release dates, licenses), and elements of the referencing system (in-text citations, URLs, and software coreferences for cross-sentence linkage), to a specific software mention.
- **Subtask III: Relation Extraction**: This task involves establishing relationships between software entities and their attributes using specific relation types and mapping each attribute to these relations. This task aids in comprehending the interrelations of software entities within scholarly texts.

In this paper, we present our approach for three subtasks at the SOMD shared task. Instead of using traditional classification methods, in this study, we use a generative approach for shared task problems. We leverage the power of large language models (LLMs) combined with prompt engineering, further fine-tuning them with the LoRA technique.

2 Related Work

Named Entity Recognition (NER) is the task of identifying and classifying key information in text into predefined categories. A common approach for NER is to formulate it as a sequence labelling task. For example, Hammerton et al. (2003) [3] used unidirectional LSTMs to obtain token-level representations, which were then fed to a softmax classifier for predictions. Collobert et al. (2011) [1] employed CNNs to embed each input word and used CRF to decode each embedding into a specific entity type. Devlin et al. (2019) [2] utilized BERT to obtain token-level representations for classifications. Methods leveraging large pre-trained Transformers have further pushed the state-of-the-art, such as the UnifiedSG model [10]. Recent advancements include DeepStruct by [20], which modifies language modelling to enhance the model's awareness of the corpus's logical structure, and then applies this trained model to NER tasks. Other approaches involve introducing specialized architectures designed specifically for the characteristics of NER, as seen in the work of [7].

Regarding LLMs for NER, Wang et al. (2023) [21] examined the use of text-generation models for sequence labeling in low-resource and few-shot scenarios. Ji et al. (2023) [5] proposed a zero-shot and few-shot NER framework based on the Vicuna model.

For LLMs in relation extraction, Wadhwa et al. (2023) [18] utilized few-shot prompting and fine-tuning with large language models to achieve state-of-the-art performance. Wan et al. (2023) [19] introduced GPT-RE to enhance relation extraction accuracy through task-specific entity representations.

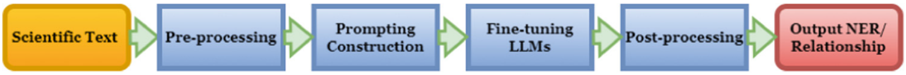

Fig. 1. Our workflow for three sub-tasks at the SOMD shared task.

Fine-tuning large language models (LLMs) for downstream tasks is a recent approach that leverages their capabilities to solve various NLP tasks. Stavropoulos et al. (2023) [16] introduced a method for extracting knowledge from scientific literature, focusing on identifying datasets and code/software mentions. Using a meticulously curated dataset, generated with ChatGPT, the authors employed Low-Rank Adaptation (LoRA) [4] to fine-tune a Large Language Model (LLM), turning Research Artifact Analysis (RAA) into an instruction-based Question Answering (QA) task. This innovative approach significantly enhances the LLM's performance, facilitating accurate extraction of research artifacts and addressing reproducibility and reusability challenges in scientific research.

3 Approach

3.1 Overview

The diagram in Fig. 1 illustrates our approach for all three subtasks. The framework consists of four steps: a pre-processing layer, a layer for Prompting construction, a layer of fine-tuning Large Language Models, and a post-processing layer. Firstly, the input text is subject to several processing steps in the pre-processing layer. Following this, we fine-tune LLMs in order to obtain probability outputs of labels. Finally, the probability outputs are processed by the post-processing layer to convert the model output to the format that can be submitted to the competition. The detailed structure of the pipeline is described in the following:

Pre-processing: Pre-processing before fine-tuning a Large Language Model is essential for cleaning, standardizing, and structuring the data to optimize the model's learning process and improve its performance on specific downstream tasks:

- **Subtask I**: In the dataset sentence, certain links or websites should be eliminated or substituted to assist the language model in disregarding extraneous information. Next, we convert the original label of subtask I into natural language to input it into LLMs. For example, we have the input sentence, original label and converted label as follows:
  - **Input sentence:** In this work, we described the Delphi package and associated resources.
  - **Original label:** O O O O O O O B-Application_Creation O O O O O.
  - **Converted label:** DelPhi:Application_Creation.
- **Subtask II**: we convert the original label of subtask I into natural language for inputting into LLMs. The process is similar to subtask I.

– **Subtask III**: we convert the original label of subtask III into natural language for inputting into LLMs. For example, we have the input sentence, original label and the covered label as follows:
  • **Input sentence:**: Users are welcome to use an instance of PhyloBot available at http://www.phylobot.com, or launch their own instance of PhyloBot using its open-source code.
  • **Original label:** O O O O O O O O B-Application_Deposition O O B-URL O O O O O O O B-Application_Mention O O O O O O O.
  • **Converted label:** http://www.phylobot.com is URL_of PhyloBot.

Prompting Construction: Through experimentation, we conducted comparative analyses involving the utilization and absence of Prompting Construction. Our findings suggest that transforming labels from the BIO Tags into a prompt-based format aids Large Language Models (LLMs) in grasping the task's intent. Moreover, the linguistic structure embedded in the prompts closely corresponds to the knowledge encapsulated within LLMs.

– **Subtask I**: The prompt outlines four key skills: (1) Sentence Scrutiny: Analyzing sentences and flagging software-related keywords. (2) Entity Identification: Pinpointing entities related to identified keywords and ensuring contextual alignment. (3) Categorize Software: Evaluating input by mapping it to predefined software categories. (4) Ascertain Mention Type: Determining the type of software mention, such as Deposition, Usage, Creation, or Mention. An example is provided in Fig. 2.
– **Subtask II**: The prompt outlines four key skills: (1) Comprehensive Understanding: Thoroughly analyze the given text and software terminologies for a holistic view of the context. (2) Details Recognition: Identify elements associated with specific categories, including Abbreviation, Developer, Extension, AlternativeName, Citation, Release, URL, and Version. (3) Focused Extraction: Diligently sift through the text to extract data fitting the identified categories. (4) Data Sorting: Allocate extracted information to its appropriate category, including Abbreviation, Developer, Extension, AlternativeName, Citation, Release, URL, and Version. An example is provided in Fig. 2.
– **Subtask III**: The prompt outlines two key skills: (1) Rigorous Examination: Thoroughly examine given phrases and related software terminologies to comprehend the wider context. (2) Relationship Categorization: For each software mentioned, categorize its relationships with other recognized pieces of information, including versions, developers, URLs, or host applications for plugins. An example is provided in Fig. 3.

Fine-Tuning LLMs: We implement the fine-tuning with different large language models as below:

– **Phi** [8]: Phi 1.5 and Phi-2, developed by Microsoft Research, are small language models (SLMs). Phi-1.5, with 1.3 billion parameters, excels in Python

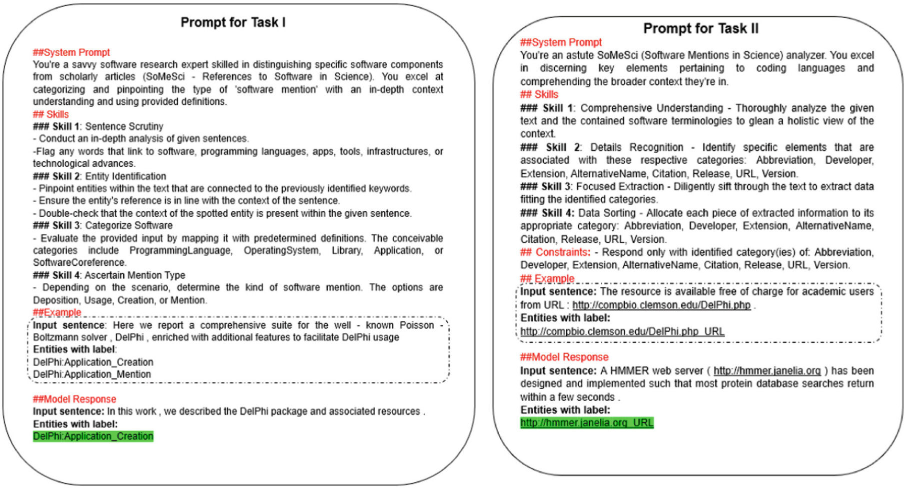

Fig. 2. Prompting Construction for Subtask I and Subtask II

coding tasks and common sense reasoning. Despite its smaller scale, it performs comparably to models five times larger. Phi-2, boasting 2.7 billion parameters, demonstrates outstanding reasoning and language understanding capabilities. It matches or outperforms models up to 25 times larger. The model showcases that smaller models can achieve remarkable feats when strategically trained and curated with high-quality data.

- **BloomZ** [11]: The BLOOM family of LLMs, including Bloomz, leverages the 176-billion parameter BLOOM model for multitask fine-tuning. This approach improves performance on unseen tasks compared to purely pre-trained models.
- **Mistral** [6]: Mistral-7B, developed by Mistral AI, is a 7-billion parameter large language model (LLM) designed for efficiency and ease of use. It boasts competitive performance on various benchmarks compared to larger models, highlighting its potential for applications where computational resources are limited. Additionally, Mistral-7B is available through open-source channels and offers fine-tuning capabilities for specific tasks, promoting further expLoRAtion and customization by researchers.
- **Llama-2** [17]: pushes the boundaries of large language models, offering a range of pre-trained and fine-tuned models with varying parameter sizes (7B to 70B). This family of models, built upon advancements in training data, context length, and efficient inference techniques, demonstrates significant progress compared to the LLaMA 1.
- **Jaskier-7b-dpo**https://huggingface.co/bardsai/jaskier-7b-dpo-v6.1: Jaskier-7b-dpo is a 7-billion parameter large language model (LLM), is based on the PAULML/OGNO-7B model and further fine-tuned using Direct Preference Optimization (DPO) [12]. This fine-tuning approach aims to enhance

the LLM's capabilities beyond general language understanding by incorporating user preferences into its training process.

Post-processing: In this layer, the task involves converting the output of the LLMs back to the original label format for submission to the competition. Essentially, this layer represents the reverse process of the pre-processing layer for all three tasks.

3.2 Low-Rank Adaptation

Low-rank adaptation (LoRA) stands out as a widely adopted method for the fine-tuning of pre-trained models, such as large language models and diffusion models. This approach is particularly known for its efficiency in leveraging low-rank adaptation of weight matrices. In the context of our paper, we draw upon the effectiveness of LoRA to enhance and optimize our proposed method. As [4] mentioned, LoRA allows us to train some dense layers in a neural network indirectly by optimizing rank decomposition matrices of the dense layers' change during adaptation instead while keeping the pre-trained weights frozen. Due to the aforementioned benefits, we applied LoRA as the fine-tuning technique to adapt LLMs to the domain of three subtasks.

4 Experimental Setup

4.1 Dataset and Evaluation Metrics

Dataset: Each subtask in the SOMD Shared Task is assessed independently. Subtask II's attributive NER builds upon software entities identified in Subtask I, while Subtask III's Relation Extraction combines annotations from both previous subtasks. The dataset follows a hierarchical structure, leading to some data leakage, particularly in Subtask I. Despite this, the design ensures interconnectedness and cumulative challenge complexity. Initially presented as the SoMeSci knowledge graph [15], the Corpus consists of 1,367 documents containing 399,942 triples representing 47,524 sentences. It includes 2,728 software entities and 7,237 labeled entities, both positive and negative examples, enhancing model accuracy. Duplicate sentences, headings, and varied sentence lengths add complexity to extracting information from scientific texts.

Evaluation Metrics: The evaluation metrics for the three sub-tasks are reported using Precision, Recall, and F1-score. These scores are calculated based on exact matches and weighted accordingly.

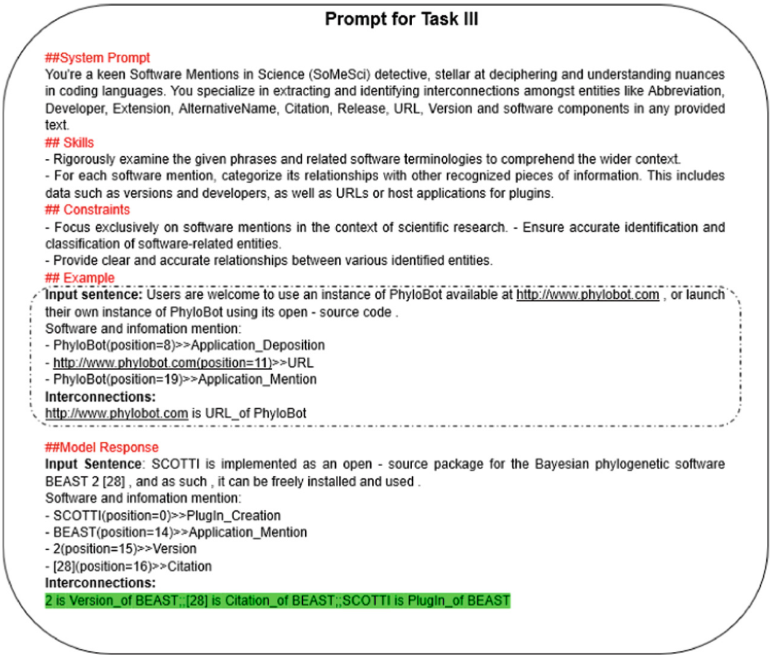

Fig. 3. Prompting Construction for Subtask III

4.2 System Settings

Training Setting: We use the PyTorch framework and HuggingFace's Transformers library [22] for our system. We used the various LLM architectures and fine-tuned them on each task. We train the model in a batch size of 4 on the training dataset. We used a learning rate of 2.10^{-4}. For the optimizer, we used AdamW optimizer [9]. AdamW is a stochastic optimization method, and it modifies the implementation of weight decay in Adam by separating weight decay from the gradient update.

LoRA Setting: In the specified LoRA setting for Causal Language Modeling (CAUSAL_LM), key parameters are defined. The attention dimension "r" is set to 8, "LoRA _alpha" to 16, and a dropout of 0.05 in LoRA layers. Target modules for attention, like "q _proj" and "o _proj," are identified, with exclusions like "embed _tokens" and "lm _head." We trained the model on 12 epochs for subtask I and 15 epochs for subtask II and subtask III and gaining the experiments results as in Table 1.

5 Main Results

In this section, we present the results of our final submission model within the framework of the SOMD shared task competition's main tasks. When evaluating

Table 1. The performances of various LLMs for three Subtasks.

Models	Subtask I			Subtask II			Subtask III		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Phi - 1.5	70.10	66.80	66.99	69.02	71.36	69.74	75.58	76.97	76.02
Phi - 2	72.13	71.46	70.85	71.64	74.03	72.36	80.00	81.21	80.38
BloomZ - 1B1	69.99	69.13	68.53	71.15	71.60	70.80	83.79	82.73	83.01
BloomZ - 1B7	69.91	68.35	67.59	74.05	70.63	72.02	87.65	86.06	86.65
BloomZ - 3B	73.35	70.87	71.47	71.69	71.60	71.17	86.80	74.55	80.03
BloomZ - 7B1	73.84	72.43	72.07	71.94	73.06	71.91	85.84	85.76	85.56
Mistral - 7B	66.99	56.89	59.74	58.21	59.47	56.14	79.12	80.30	79.60
Llama - Chat - 7B	73.83	71.65	72.50	70.47	**75.73**	72.65	89.33	87.58	88.33
Jaskier-7b-dpo - 7B	**76.14**	**74.95**	**73.96**	**74.46**	74.76	**74.28**	**90.04**	**89.70**	**89.74**

Table 2. The results and corresponding ranking of our best submission with other team participants for three Subtasks in the official scoreboard.

User	Subtask I			Subtask II			Subtask III		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
david-s477	73.90	71.10	69.20 (Top 2)	–	–	–	–	–	–
ThuyNT03	72.90	64.90	67.80 (Top 3)	–	–	–	–	–	–
ottowg	67.90	66.40	65.20 (Top 4)	83.50	84.70	83.80 (Top 1)	91.10	92.40	91.16 (Top 1)
vampire	63.70	68.20	64.80 (Top 5)	–	–	–	–	–	–
Ours results	**76.14**	**74.95**	**73.96 (Top 1)**	**74.46**	**74.76**	**74.28 (Top 2)**	**90.04**	**89.70**	**89.74 (Top 2)**

Subtask II and Subtask III, we specifically focus on comparing our performance metrics with those of the leading team. On the other hand, we offer a thorough summary by presenting the results of the top five performing systems.

The performance on Subtask I revealed notable differences among the LLM models. Jaskier-7b-dpo emerged as the clear leader, achieving a precision of 76.14 and a robust F1-score of 73.96. This indicates that Jaskier-7b-dpo effectively identified relevant information and minimized errors in its responses for Subtask I. In contrast, Mistral-7B displayed the poorest performance, with a significantly lower precision of 66.99 and an F1-score of 59.74. This suggests that Mistral-7B struggled with this specific task, potentially due to limitations in its training data or architecture.

Jaskier-7b-dpo consistently achieved the best performance across all metrics for Subtask II and Subtask III. This finding suggests that the fine-tuning process using Direct Preference Optimization (DPO) [13] has significantly improved the model's performance compared to other baseline large language models (LLMs). It is noteworthy that Llama Chat exhibited the highest recall on Subtask II (Table 1).

6 Discussion

6.1 Challenges in Applying LLMs to NER Tasks

Incorporating Large Language Models (LLMs) into Named Entity Recognition (NER) tasks presents distinct challenges that can greatly affect extraction results' effectiveness and dependability. A common challenge in generative NER methods is the occurrence of hallucination, as documented by Wang et al. (2023) [21], wherein the model generates entities that aren't actually present in the test data. This can arise from the model mistakenly interpreting given examples as text from which entities should be extracted, resulting in inaccuracies and inconsistencies in the outcomes.

Moreover, locating mention positions poses significant challenges, especially within span-based evaluation systems. These systems assess entity extraction accuracy based on the exact text span identified as an entity. Discrepancies in the extracted span, such as corrected spellings or variations in representation, can complicate matching.

Additionally, texts with multiple mentions of the same entity add complexity. The task involves accurately classifying each mention as an entity while addressing ambiguity arising from entities with identical representations. Overlapping or nested mentions that deviate from the ground truth data add further complexity, demanding nuanced approaches to entity recognition and classification.

6.2 Conclusion and Future Work

This paper introduced an effective strategy for NER tasks within the SOMD shared tasks. Our system involves fine-tuning LLMs applying the LoRA technique with parameter counts ranging from 1 billion to 7 billion to serve as the NER extractor. Our experiments demonstrated that harnessing the capabilities of LLMs leads to competitive results in downstream tasks like NER.

References

1. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**(ARTICLE), 2493–2537 (2011)
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: pre-training of deep bidirectional transformers for language understanding. google AI language. arXiv preprint arXiv:1810.04805 (2019)
3. Hammerton, J.: Named entity recognition with long short-term memory. In: Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003, pp. 172–175 (2003)
4. Hu, E.J., et al.: Lora: low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
5. Ji, B.: Vicunaner: zero/few-shot named entity recognition using vicuna. arXiv preprint arXiv:2305.03253 (2023)
6. Jiang, A.Q., et al.: Mistral 7b (2023)

7. Li, J., et al.: Unified named entity recognition as word-word relation classification. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 10965–10973 (2022)
8. Li, Y., Bubeck, S., Eldan, R., Giorno, A.D., Gunasekar, S., Lee, Y.T.: Textbooks are all you need ii: phi-1.5 technical report, September 2023
9. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
10. Lu, Y., et al.: Unified structure generation for universal information extraction. arXiv preprint arXiv:2203.12277 (2022)
11. Muennighoff, N., et al.: Crosslingual generalization through multitask finetuning. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of ACL, pp. 15991–16111, July 2023
12. Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C.D., Finn, C.: Direct preference optimization: your language model is secretly a reward model (2023)
13. Rafailov, R., Sharma, A., Mitchell, E., Manning, C.D., Ermon, S., Finn, C.: Direct preference optimization: your language model is secretly a reward model. In: Advances in Neural Information Processing Systems, vol. **36** (2024)
14. Sang, E.F., De Meulder, F.: Introduction to the conll-2003 shared task: language-independent named entity recognition. arXiv preprint cs/0306050 (2003)
15. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: Somesci-a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4574–4583 (2021)
16. Stavropoulos, P., Lyris, I., Manola, N., Grypari, I., Papageorgiou, H.: Empowering knowledge discovery from scientific literature: a novel approach to research artifact analysis. In: Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023), pp. 37–53 (2023)
17. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models (2023)
18. Wadhwa, S., Amir, S., Wallace, B.C.: Revisiting relation extraction in the era of large language models. In: Proceedings of the conference. Association for Computational Linguistics. Meeting, vol. 2023, p. 15566. NIH Public Access (2023)
19. Wan, Z., Cheng, F., Mao, Z., Liu, Q., Song, H., Li, J., Kurohashi, S.: Gpt-re: in-context learning for relation extraction using large language models. arXiv preprint arXiv:2305.02105 (2023)
20. Wang, C., Liu, X., Chen, Z., Hong, H., Tang, J., Song, D.: Deepstruct: pretraining of language models for structure prediction. arXiv preprint arXiv:2205.10475 (2022)
21. Wang, S., et al.: GPT-NER: named entity recognition via large language models. arXiv preprint arXiv:2304.10428 (2023)
22. Wolf, T., et al.: Huggingface's transformers: state-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)

Falcon 7b for Software Mention Detection in Scholarly Documents

AmeerAli Khan[1], Qusai Ramadan[3], Cong Yang[4], and Zeyd Boukhers[1,2](✉)

[1] Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany
{ameerali.khan,zeyd.boukhers}@fit.fraunhofer.de
[2] University Hospital of Cologne, Cologne, Germany
[3] University of Koblenz, Koblenz, Germany
qramadan@uni-koblenz.de
[4] School of Future Science and Engineering, Soochow University, Suzhou, China
cong.yang@suda.edu.cn

Abstract. This paper aims to tackle the challenge posed by the increasing integration of software tools in research across various disciplines by investigating the application of Falcon-7b for the detection and classification of software mentions within scholarly texts. Specifically, the study focuses on solving Subtask I of the Software Mention Detection in Scholarly Publications (SOMD), which entails identifying and categorizing software mentions from academic literature. Through comprehensive experimentation, the paper explores different training strategies, including a dual-classifier approach, adaptive sampling, and weighted loss scaling, to enhance detection accuracy while overcoming the complexities of class imbalance and the nuanced syntax of scholarly writing. The findings highlight the benefits of selective labelling and adaptive sampling in improving the model's performance. However, they also indicate that integrating multiple strategies does not necessarily result in cumulative improvements. This research offers insights into the effective application of large language models for specific tasks such as SOMD, underlining the importance of tailored approaches to address the unique challenges presented by academic text analysis.

Keywords: Falcon · Entity Recognition · Mention Detection · Scholarly Data Analysis

1 Introduction

The integration of software tools into scientific research is no longer confined to engineering disciplines but has extended to all disciplines, including the humanities and social sciences. This is largely driven by the escalating need to process and analyze data across various domains. Consequently, an extensive mention of these tools in the scientific articles. Furthermore, the volume of published

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 278–288, 2024.
https://doi.org/10.1007/978-3-031-65794-8_20

scholarly articles continues to grow year after year, which makes overcoming the challenge of managing this vast repository of knowledge more important than ever before. In response, numerous projects and initiatives such as EOSC[1] and NFDI[2] have been launched to organize research outputs, including articles, software mentions, and datasets, in a manner that adheres to the FAIR principles [27] (i. e., Findable, Accessible, Interoperable, and Reusable) to enhance the overall integrity and reproducibility of scientific research.

The absence of a standardized mechanism for authors to accurately cite and reference tools and datasets in their scholarly articles necessitates the extraction of such mentions post-publication. However, this task is challenging due to the unstructured nature of these articles. This underscores the importance of developing sophisticated and reliable methodologies capable of automatically and accurately recognizing mentions of these tools. This paper addresses this challenge by concentrating on a key aspect of it: "*Subtask I of the Software Mention Detection in Scholarly Publications (SOMD)*."[3] The aim is to explore new methodologies and models of software mention detection processes in academic literature.

Although the challenge of detecting software mentions has been directly addressed using the KG approach [20,22], a lot of approaches that solve the general problem have been proposed to solve the general problem of extracting information from scholarly data, including Named Entity Recognition [23], Metadata Extraction [1,5,10] and Reference Extraction and Segmentation [4,8,19]. The proven capabilities of large language models (LLMs) to understand and generate human-like text offer new possibilities. Among the advanced models, the LLM such as the Falcon-7b model stands out due to its decoder-only architecture and the breadth of its training data. Although it was initially designed for generative tasks, the scale of architecture and the breadth of its training data present an exciting opportunity for exploring its potential for application in specialized areas of NLP, such as the detection and classification of software mentioned in academic publications.

This paper investigates the effectiveness of the Falcon-7b model on the task of software mention detection in the scientific literature by adapting it to the task. The choice of model is arbitrary without assuming it has a specific advantage over similar models like LLAMA 2 [24] or GPT3.5 [6]. This choice allows us to explore how well such LLM models perform on a specialized task, specifically, we leverage the model's extensive training on a wide array of texts and its advanced understanding of linguistic nuances. This effectiveness is critically assessed through comprehensive experiments, encompassing a variety of evaluation metrics to ensure a balanced analysis of the model's applicability to the task at hand.

The structure of this paper is laid out as follows: Sect. 2 provides a review of the relevant literature. In Sect. 3, we delve into the methodology of this study,

[1] https://eosc-portal.eu/.
[2] https://www.nfdi.de/.
[3] https://nfdi4ds.github.io/nslp2024/docs/somd_shared_task.html.

outlining the approaches and techniques utilized. Section 4 presents the experimental setup, alongside the results achieved from our investigations. Finally, Sect. 5 concludes the paper, summarizing our findings and offering insights into potential avenues for future research.

2 Related Work

As there are not so many efforts to detect software mention using LLM, we review in this section approaches on named-entity recognition (NER), where we consider that software mention detection is a use case of this problem. This review is structured in three categories:

2.1 Rule-Based and Classical Machine Learning Approaches

Named entity recognition has been at the core of machine learning research for decades due to its importance in a variety of applications. However, the task is challenging and thus many approaches have been proposed to tackle the problem in specific scenarios. [2] introduces a rule-based NER framework designed for the Malay language, to improve the retrieval process of articles. It addresses the challenges faced by NER processes in languages with morphological differences and the lack of existing systems for the Malay language. By analyzing the domain of studies and the specific linguistic features of Malay, this framework accurately classifies named entities such as people, organizations, and locations. [16] proposes to address the limitations of traditional rule-based approaches by using Hidden Markov Models (HMM) for NER tasks in Indian Language beyond domain-specific applications. [25] proposes an approach using the Conditional Random Field (CRF) and Active Learning (AL) algorithm, where the training process of the CRF classifier is repeated until the model stabilizes. This resulted in an efficient and cost-effective outcome. [9] uses the Beam search algorithm to detect named entities in the Persian language by segmenting text into suitable and unsuitable expressions for the named entities and then applying dynamic external knowledge to recognize the emerging named entity.

2.2 Deep Learning-Based Approaches

Deep Neural Networks have proven significant performance over classical machine learning approaches on different tasks, including NLP, computer vision and others. Consequently, they have been used to address NER tasks as well [12,14]. [29] proposes a NER framework called E-NER that uses evidential deep learning (EDL) to explicitly model predictive uncertainty for named entity recognition (NER) tasks. It addresses the challenges of sparse entities and out-of-vocabulary (OOV) entities in NER by introducing uncertainty-guided loss terms and training strategies. E-NER achieves accurate uncertainty estimation, better OOV/OOD detection performance, and improved generalization ability on OOV entities compared to state-of-the-art baselines. [15] introduces a novel approach to Clinical NER for de-identifying sensitive health information in clinical texts by developing a Capsule-LSTM network that leverages the strengths of capsule networks for capturing complex data relationships and LSTM networks for understanding sequential data. [13] the All CNN (ACNN) model for Chinese clinical NER that employs CNN enhanced by an attention mechanism, sidestepping the inefficiencies of traditional LSTM models. By leveraging multi-level CNN layers with various kernel sizes and a residual structure, ACNN adeptly captures context information across different scales, addressing the challenges posed by the complex grammar and terminology of Chinese clinical texts. [7] proposes to combine CNN and bi-LSTM architectures for biomedical NER (bioNER), aiming to efficiently handle the complexities of biomedical texts, such as variant spellings and inconsistent use of prefixes and suffixes. The proposed combinatorial feature embedding and attention mechanism for enhanced entity recognition demonstrate superior performance on benchmark datasets JNLPBA and NCBI-Disease when compared with existing methods.

2.3 Large Language Model-Based Approaches

Due to their power to understand and generate natural text, LLMs have been employed for many Natural Language Processing (NLP) tasks, including Named Entity recognition. [26] proposes GPT-NER, a method that transforms the sequence labelling task of Named Entity Recognition (NER) into a text generation task, allowing large language models (LLMs) to be easily adapted for NER. GPT-NER achieves comparable performances to fully supervised baselines on five widely adopted NER datasets and exhibits a greater ability in low-resource and few-shot setups, making it suitable for real-world NER applications with limited labelled examples. [17] employs Generative Pre-trained Transformer 3 (GPT-3) by OpenAI together with a weak supervisor to address the NER challenge within the legal domain, exemplified by documents from the Official Gazette of the Federal District (DODF). [11] proposes an architecture that refines the adversarial example generation process for LLM using disentanglement and word attribution techniques to efficiently generate adversarial examples while maintaining semantic similarity. The experiments conducted on benchmark datasets-CoNLL-2003, Ontonotes 5.0, and MultiCoNER-demonstrated that the approach improves the F1 scores by 8%.

3 Method

This section elaborates on the methodologies applied for the Software Mention Detection in Scholarly Publications (SOMD) [21] Subtask I, part of the Natural Scientific Language Processing and Research Knowledge Graphs (NSLP)[4] 2024 workshop. Our approach is centred on token classification, addressing the unique challenges software mention recognition poses in scientific texts.

[4] https://nfdi4ds.github.io/nslp2024/.

The primary objective of the subtask is to recognize software mentions within individual sentences, further classifying them by mention type (e.g., mention, usage, creation) and software type (e.g., application, programming environment, package). The task is approached with a methodology that innovatively applies a large language model (LLM), such as the Falcon-7b [3] model, as the foundation for a token classification system.

The core of our methodology is The Falcon-7b model, which is a decoder-only architecture known for its performance across a wide range of NLP tasks. Despite its primary design as a generative model, we adapt Falcon-7b for token classification by appending a classification layer atop its structure. This adaptation is driven by the hypothesis that the extensive pre-training on diverse corpora, coupled with its vast parameter space, provides the Falcon-7b model with a nuanced understanding of textual context. Such capabilities can significantly enhance the model's proficiency in identifying and classifying software mentions within the complex syntax and semantics of scholarly writing.

The methodology implemented in this study is structured around several strategic training approaches, tailored to address the intrinsic challenges of the SOMD task. Initially, the task is framed as a token classification problem, where labels are assigned to each word or subtoken within a sentence to denote the presence and category of software mentions. A notable challenge arises from the tendency of transformer-based models like Falcon-7b to segment words into multiple subtokens, which complicates the direct application of labels.

To address the challenge of labelling consistency in the presence of subtoken segmentation by transformer models, our methodology employs two distinct strategies. The first, referred to as "**Unified Labeling**" assigns the same label to all subtokens derived from a single word's segmentation, ensuring consistency across the subtoken sequence. This maintains label continuity across divisions, which facilitates coherent entity recognition despite the segmentation process. In contrast, the second strategy "**Selective Labeling**" assigns a label only to the first subtoken of a segmented word, disregarding subsequent subtokens. This method aims to minimize label redundancy and computational complexity associated with processing multiple labels for a single entity. Each approach is independently explored to determine its efficacy in addressing the challenges of subtoken label alignment in the context of the SOMD task.

A critical obstacle encountered in the SOMD task is the substantial class imbalance, where data is skewed towards non-mention ('O') labels. To address this imbalance, we implemented two distinct and independently tested strategies to recalibrate the dataset for training. In the first strategy "**Weighted Loss**", weighted loss is applied, where class weights are inversely proportional to class frequencies. however, given the substantial imbalance, original weight values became impractical for underrepresented classes. To resolve this, we scaled the weights within a sensible range, setting minimums and maximums (1, and thresholds of 25, 50, 100, and 200), thereby enabling the nuanced training of models across various weight configurations to explore their efficacy in balancing classification performance. The second strategy "**Adaptive Sampling**" strategically segments the dataset into over-represented (all 'O' tokens) and under-represented (at least one non-'O' tokens) categories. This involves oversampling the under-represented data by a factor of 2 and undersampling the over-represented data to sizes equal to multiples (1, 1.5, 3) of the oversampled data volume. Adaptive Sampling aims to achieve a more balanced class distribution, enhancing the model's capacity to learn from a representative spectrum of the dataset. Each strategy's independent evaluation provides insights into its effectiveness in addressing dataset imbalance challenges in the SOMD task.

Additionally, recognizing the multifaceted nature of software mentions, separate token classifiers are developed for identifying software types and the mentioned types. We call this strategy "**Dual-Classifier**". It enables more precise label application by separating the task into two distinct classification problems, by distinguishing between software and mention types. This setup is designed to explore whether such a nuanced approach can effectively capture the diverse nature of software mentions, offering a potentially more sophisticated mechanism for their identification and classification.

To ensure the effectiveness and generalizability of the model, the pre-defined training dataset was utilized to develop and refine the model's capabilities. For evaluating the model's performance in accurately identifying and classifying software mentions, the test dataset provided for Subtask II was employed. With this, we aim to conduct a comprehensive assessment of the model's ability to generalize across various scenarios and text variations found in scholarly publications, thereby validating the model's applicability and effectiveness in real-world tasks.

Following the comprehensive methodologies delineated for the Software Mention Detection (SOMD) task, our evaluation process is designed to rigorously assess the effectiveness of our approaches, namely the "Unified Labeling," "Selective Labeling," "Weighted Loss," "Adaptive Sampling," and the "Dual-Classifier" Approach. Central to our evaluation is the F1-Score, focusing on exact matches, which serves as a critical metric to quantify the precision and recall of our models in accurately identifying and classifying software mentions. Adherence to the IOB2 format for our submission files ensures our alignment with the standardized training labels, facilitating direct comparison of our model's performance against established benchmarks.

4 Experimental Results

4.1 Results

To evaluate our method on Software Mention Detection in Scholarly Publications (SOMD) Subtask I, we explored various settings centred around the Falcon-7b model. The experiments are conducted using the Hugging Face Transformers library [28], with PyTorch [18] as the computational backend. Model training and evaluation were performed on a high-performance computing cluster equipped with NVIDIA A100 GPUs. To maintain consistency across all experiments conducted in this study, we standardized our training hyperparameters. This ensures that any observed differences in model performance are attributable to the variations in model architecture, data preprocessing, or other experimental conditions, rather than inconsistencies in training configurations.

The focal point of our evaluation is the F1 score, which balances precision and recall and offers a comprehensive measure of model performance. The evaluation result on the test dataset is illustrated in Table 1. These results are submitted to the shared task platform under the username: "**fddaFIT**".

Table 1. Evaluation metrics for each experimental approach.

Method	Precision	Recall	F1 Score
Unified Labeling	0.6769	0.4718	0.5561
Selective Labeling	0.7563	0.5243	0.6193
Adaptive Sampling multiples@1	0.6496	0.4932	0.5607
Adaptive Sampling multiples@1.5	0.7480	**0.5417**	**0.6284**
Adaptive Sampling multiples@3	0.7213	0.4874	0.5817
Weighted loss scaled@25	0.7559	0.4990	0.6012
Weighted loss scaled@50	0.7211	0.4971	0.5885
Weighted loss scaled@100	0.7195	0.4932	0.5853
Weighted loss scaled@200	0.7110	0.4874	0.5783
Dual-Classifier	**0.7602**	0.5048	0.6068
Dual-Classifier + Adaptive Sampling multiples@1.5	**0.7602**	0.5048	0.6068

In comparison between the two distinct labelling strategies:"Unified Labeling" and "Selective Labeling." The outcome favoured the "Selective Labeling" strategy, which demonstrated superior precision and recall and achieved a higher F1 score of 0.6193 compared to the 0.5561 of "Unified Labeling." This demonstrates the effectiveness of selectively assigning labels in enhancing the model's precision and recall. This finding guided the direction of subsequent experiments, by embedding the "Selective Labeling" strategy in our methodology.

Further investigations focus on adaptive sampling and weighted loss scaling to address the notable challenge of dataset imbalance. Adaptive sampling experiments, applying various undersampling multipliers to over-represented data, demonstrated the precise impact of data distribution on model efficacy. The employment of a multiplier of 1.5 showed an enhancement of the F1 score to 0.6284, the highest among adaptive sampling variations. This suggests an optimal balance in dataset composition, significantly contributing to model performance.

Experiments with weighted loss, adjusted across a range of maximum weights (25, 50, 100, and 200), aimed to refine the model's sensitivity to class frequencies. While experimenting with weighted loss scaling offered insights into handling class imbalance, the adjustments did not surpass the adaptive sampling's peak F1 score, with the highest recorded F1 at 0.6012 for a scaling factor of 25.

The "Dual-Classifier" approach introduced a bifurcated strategy for identifying software types and mention types, aiming to enrich the model's understanding and classification accuracy. Interestingly, this approach alone achieved an F1 score of 0.6068, comparable to some weighted loss scaling strategies but slightly below the best-performing adaptive sampling method. Combining the "Dual-Classifier" with "Adaptive Sampling" at a multiplier of 1.5 did not further enhance the F1 score, indicating that while each method independently contributes to addressing specific challenges in SOMD, their combined effect does not necessarily result in cumulative improvements.

The experiments demonstrate the critical role of selective labelling and adaptive sampling in enhancing F1 scores for SOMD. While weighted loss scaling and the "Dual-Classifier" approach contribute to performance improvements, the combination of strategies does not yield further enhancements. This indicates the necessity of strategic selection and implementation in model development for SOMD tasks.

In these experiments, we observed that the model effectively identifies software mentions within the complex academic text and accurately classifies entities like specific software tools or programming languages.

For example, it might correctly identify "Python" as a programming language used within a research context. However, challenges arise in distinguishing between mentions of software and instances where the software is being actively used or discussed in depth. This differentiation is crucial for understanding the role of software in research, as only mentions might not signify importance or relevance to the study's outcomes. Examples of the model's output are provided in the full version[5] of this paper.

5 Conclusion

In this paper, we presented the efficacy of the Falcon-7b model, a prominent Large Language Model (LLM), in tackling the nuances of Software Mention Detection (SOMD) within scholarly publications. Guided by the hypothesis that advanced LLMs could significantly improve the precision and recall for SOMD tasks due to their extensive training on diverse datasets, our study systematically explored various strategies centred around the Falcon-7b model.

The comparative analysis of "Unified Labeling" and "Selective Labeling" strategies revealed a preference for "Selective Labeling," which yielded a higher F1 score. This result underscores the incremental nature of advancements achievable with the Falcon-7b model in the context of SOMD tasks. Additionally, our experiments with adaptive sampling and weighted loss scaling aimed at addressing dataset imbalances highlighted that, despite certain improvements, the enhancements were not as substantial as anticipated. The adaptive sampling strategy, especially with a multiplier of 1.5, did indeed enhance the F1 score, to the highest obtained score. However, this observation was not sufficient to

[5] https://zenodo.org/doi/10.5281/zenodo.10993039.

categorize the performance as extraordinary. This finding suggests that while the Falcon-7b model and similar LLMs hold promise for NLP tasks, their application in specialized areas such as SOMD may not always yield groundbreaking results.

In conclusion, the outcomes of our study indicate that, despite the advanced capabilities of LLMs like Falcon-7b, the performance improvements in specific NLP tasks such as SOMD are modest. While LLMs offer advantages in processing and understanding complex language patterns, their effectiveness in specialized domains like software mention detection within scholarly texts does not markedly outperform more traditional approaches. This emphasizes the importance of combining LLMs with other approaches and the need to explore a broad spectrum of models and methodologies to identify the most effective solutions for specialized NLP challenges.

Acknowledgements. This work was funded by the FAIR Data Spaces project of the German Federal Ministry of Education and Research (BMBF) under grant numbers FAIRDS05 and FAIRDS15 and by NFDI4DS project with grant number 460234259.

References

1. Ahmed, R.M.W.: Metadata Extraction using Geometric and Layout Features from Research Publications. Ph.D. thesis, Capital University (2022)
2. Alfred, R., Leong, L.C., On, C.K., Anthony, P.: Malay named entity recognition based on rule-based approach (2014)
3. Almazrouei, E., et al.: The Falcon Series of Open Language Models (Nov 2023). https://doi.org/10.48550/arXiv.2311.16867, http://arxiv.org/abs/2311.16867, arXiv:2311.16867 [cs]
4. Boukhers, Z., Ambhore, S., Staab, S.: An end-to-end approach for extracting and segmenting high-variance references from pdf documents. In: 2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 186–195. IEEE (2019)
5. Boukhers, Z., Beili, N., Hartmann, T., Goswami, P., Zafar, M.A.: Mexpub: deep transfer learning for metadata extraction from German publications. In: 2021 ACM/IEEE Joint Conference on Digital Libraries (JCDL), pp. 250–253. IEEE (2021)
6. Brown, T.B., et al.: Language models are few-shot learners (2020)
7. Cho, M., Ha, J., Park, C., Park, S.: Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition. J. Biomed. Inform. **103**, 103381 (2020)
8. Cuéllar-Hidalgo, R., Reyes-Salgado, G., Torres-Moreno, J.M.: Automatic reference mining: review and perspectives. In: TextMine (2022)
9. Gholami-Dastgerdi, P., Feizi-Derakhshi, M., Forouzandeh, A.: Named entities detection by beam search algorithm. Concurrency Comput. Pract. Exp.**34**(27), e7325 (2022). https://doi.org/10.1002/cpe.7325, https://onlinelibrary.wiley.com/doi/10.1002/cpe.7325
10. Guo, M., et al.: Investigations on scientific literature meta information extraction using large language models. In: 2023 IEEE International Conference on Knowledge Graph (ICKG), pp. 249–254. IEEE (2023)

11. Jin, X., Vinzamuri, B., Venkatapathy, S., Ji, H., Natarajan, P.: Adversarial robustness for large language NER models using disentanglement and word attributions. In: The 2023 Conference on Empirical Methods in Natural Language Processing (2023)
12. Kesim, E., Deliahmetoglu, A.: Named entity recognition in resumes, June 2023. https://doi.org/10.48550/arXiv.2306.13062, http://arxiv.org/abs/2306.13062, arXiv:2306.13062 [cs]
13. Kong, J., Zhang, L., Jiang, M., Liu, T.: Incorporating multi-level CNN and attention mechanism for chinese clinical named entity recognition. J. Biomed. Inform. **116**, 103737 (2021)
14. Li, W.: The Advance of deep learning based named entity recognition. Highlights in Sci. Eng. Technol. **12**, 68–73 (2022). https://doi.org/10.54097/hset.v12i.1368, https://drpress.org/ojs/index.php/HSET/article/view/1368
15. Liu, C., Li, J., Liu, Y., Du, J., Tang, B., Xu, R.: Named entity recognition in clinical text based on capsule-LSTM for privacy protection. In: Wang, D., Zhang, L.-J. (eds.) AIMS 2019. LNCS, vol. 11516, pp. 166–178. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-23367-9_12
16. Morwal, S., Jahan, N., Chopra, D.: Named entity recognition using hidden markov model (HMM). Int. J. Nat. Lang. Comput. (IJNLC) **1** (2012)
17. Oliveira, V., Nogueira, G., Faleiros, T., Marcacini, R.: Combining prompt-based language models and weak supervision for labeling named entity recognition on legal documents. Artif. Intell. Law 1–21 (2024)
18. Paszke, A., et al.: Pytorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
19. Rizvi, S.T.R., Dengel, A., Ahmed, S.: A hybrid approach and unified framework for bibliographic reference extraction. IEEE Access **8**, 217231–217245 (2020)
20. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: Somesci- a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, CIKM 2021, pp. 4574-4583. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3459637.3482017
21. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: SoMeSci- a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4574–4583. ACM, Virtual Event Queensland Australia, October 2021. https://doi.org/10.1145/3459637.3482017, https://dl.acm.org/doi/10.1145/3459637.3482017
22. Schindler, D., Zapilko, B., Krüger, F.: Investigating software usage in the social sciences: a knowledge graph approach. In: Harth, A., et al. (eds.) ESWC 2020. LNCS, vol. 12123, pp. 271–286. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-49461-2_16
23. Singh, A., Garg, A.: Named entity recognition (NER) and relation extraction in scientific publications. Int. J. Recent Technol. Eng. (IJRTE) **12**(2), 110–113 (2023). https://doi.org/10.35940/ijrte.B7846.0712223, https://www.ijrte.org/portfolio-item/B78460712223/
24. Touvron, H., et al.: Llama 2: open foundation and fine-tuned chat models (2023)

25. VeeraSekharReddy, B., Rao, K.S., Koppula, N.: named entity recognition using CRF with active learning algorithm in English texts. In: 2022 6th International Conference on Electronics, Communication and Aerospace Technology, pp. 1041–1044, December 2022. https://doi.org/10.1109/ICECA55336.2022.10009592, https://ieeexplore.ieee.org/document/10009592
26. Wang, S., et al.: GPT-NER: named entity recognition via large language models. arXiv preprint arXiv:2304.10428 (2023)
27. Wilkinson, M.D., et al.: The fair guiding principles for scientific data management and stewardship. Sci. Data **3**(1) (2016). https://doi.org/10.1038/sdata.2016.18
28. Wolf, T., et al.: HuggingFace's transformers: state-of-the-art natural language processing, July 2020. https://doi.org/10.48550/arXiv.1910.03771, http://arxiv.org/abs/1910.03771, arXiv:1910.03771 [cs]
29. Zhang, Z., et al.: E-ner: evidential deep learning for trustworthy named entity recognition. In: Findings of the Association for Computational Linguistics: ACL 2023, pp. 1619–1634 (2023)

Enhancing Software-Related Information Extraction via Single-Choice Question Answering with Large Language Models

Wolfgang Otto[1(✉)], Sharmila Upadhyaya[1], and Stefan Dietze[1,2]

[1] GESIS - Leibniz Institute for the Social Sciences, Cologne, Germany
{wolfgang.otto,sharmila.upadhyaya,stefan.dietze}@gesis.org,
stefan.dietze@hhu.de
[2] Heinrich Heine University Düsseldorf, Düsseldorf, Germany

Abstract. This paper describes our participation in the Shared Task on Software Mentions Disambiguation (SOMD), with a focus on improving relation extraction in scholarly texts through generative Large Language Models (LLMs) using single-choice question-answering. The methodology prioritises the use of in-context learning capabilities of LLMs to extract software-related entities and their descriptive attributes, such as distributive information. Our approach uses Retrieval-Augmented Generation (RAG) techniques and LLMs for Named Entity Recognition (NER) and Attributive NER to identify relationships between extracted software entities, providing a structured solution for analysing software citations in academic literature. The paper provides a detailed description of our approach, demonstrating how using LLMs in a single-choice QA paradigm can greatly enhance IE methodologies. Our participation in the SOMD shared task highlights the importance of precise software citation practices and showcases our system's ability to overcome the challenges of disambiguating and extracting relationships between software mentions. This sets the groundwork for future research and development in this field.

Keywords: Generative Large Language Models · Information Extraction · Named Entity Recognition · Relation Extraction · Software Citation · Retrieval-Augmented Generation · Single-Choice Question Answering · Software Mentions Disambiguation Task

1 Introduction

The evolution of information extraction (IE) in scholarly communications necessitates the development of innovative methodologies to accurately identify and categorise software mentions. This is a critical component for ensuring research transparency and reproducibility. The Shared Task on Software Mentions Disambiguation (SOMD) highlights the importance of refined citation practices amidst inconsistent referencing of software artifacts. This research employs generative

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 289–306, 2024.
https://doi.org/10.1007/978-3-031-65794-8_21

Large Language Models (LLMs) to address the challenges in software mention extraction and relation identification, marking a significant step towards the sophisticated extraction of software-related information. Our approach integrates Retrieval-Augmented Generation (RAG) techniques with LLMs to dissect and comprehend the intricate web of software citations and their attributive details (e.g. the developer or the version) within scientific texts. By examining the potential of LLMs in performing IE tasks by transforming them into single-choice question-answering, we present a comprehensive analysis that addresses the nuances of software mention extraction and could be applied on a broader scope of scholarly artifacts NER and relation extraction. This paper explores the complexities of applying LLMs to NER tasks, providing insights into the challenges and proposing a new methodology for relation extraction that could pave the way for future innovations in the field.

2 Related Work

This section explores works in Named Entity Recognition (NER) and Information Extraction (IE), focusing on software mention extraction, scholarly artifact NER, and the use of generative LLMs in these areas.

In software mention extraction, the Softcite dataset [2] and the SoMeSci knowledge graph [7] are significant. Softcite offers a gold-standard dataset for extracting software mentions from biomedical and economic research, while SoMeSci Knowledge Graph includes software mentions in scientific articles with relation labels such as version, developer and citations, highlighting the need for accurate software mention extraction.

In scholarly artifact NER, Saji and Matsubara [6] introduced a method using academic knowledge graphs to extract research resource metadata from scholarly papers, enhancing metadata quality and repository size. Otto et al. [5] developed the GSAP-NER corpus for extracting machine learning-related entities from scientific publications, filling the gap in general-purpose NER models. These words demostrate the importance of domain-specific NER tasks and leverageing knowledge graphs for enhancing research resource repositories.

Regarding LLMs for NER, Wang [10] explores the use of text-generation models for sequence labeling and underlying challenges, particularly in low-resource and few-shot setups. Furthermore, Xie [11] investigates LLMs' self-improving capabilities for zero-shot NER.

For LLMs in relation extraction, Wadhwa [8] uses few-shot prompting and fine-tuning with large language models achieving state-of-the-art performances in relation extraction using LLMs, while Wan [9] introduce GPT-RE to enhancing relation extraction accuracy through task-specific entity representations.

Lastly, Xu [12] conducts a comprehensive survey providing an overview of generative information extraction using LLMs, categorizing works by IE subtasks and learning paradigms, and highlighting the transformative potential of LLMs in IE.

Overall, these studies underscore the evolving methods and significant contributions in extracting information from scholarly texts using specialized datasets, domain-specific approaches, and advanced generative models.

3 SOMD Shared Task

The Shared Task on Software Mentions Disambiguation (SOMD) aims to enhance transparency and reproducibility in scientific research by improving software citation practices. Participants are required to develop models that can identify and disambiguate software mentions in scholarly texts using the expanded Software Mentions in Science (SoMeSci) knowledge graph, with a focus on AI and Computer Science.

Subtask 1, Software NER, involves identifying four types of software-related entities: Application, Plugin, Operating System, and Programming Environment. Accurately classifying software mentions is necessary to understand their role in academic discourse.

Subtask 2, Attribute NER, aims to extract ten different attributive information associated with software entities, such as alternative names, abbreviations, authorship (developers), software release maintenance (versions, extensions, release dates, licenses), and elements of the referencing system (in-text citations, URLs, and software coreferences for cross-sentence linkage), to a specific software mention.

Subtask 3, Relation Extraction, involves establishing relationships between software entities and their attributes using specific relation types and mapping each attribute to these relations. This task aids in comprehending the interrelations of software entities within scholarly texts.

The tasks **evaluation metric**, is the Weighted Average Macro F1 score. This metric adjusts the influence of each label on the final result based on the number of test instances assigned to it, ensuring that labels with fewer instances have a proportionally smaller impact on the overall performance evaluation.

4 Using LLMs for Software Related IE-Tasks

The exploration of generative LLMs such as GPT-4 for Information Extraction (IE) tasks related to software underscores the transformative potential these models hold as general-purpose task solvers. The allure of leveraging LLMs in this capacity is significant, given their ability to process and generate human-like text across a wide range of topics and formats. However, the challenge arises in the specificity and nuanced requirements of domain-specific tasks, such as NER within specialized fields. It has been observed that, despite their vast knowledge and versatility, current models like GPT-4 often fall short when tasked with domain-specific NER, primarily due to their generalized training and lack of domain-specific tuning.

To mitigate these shortcomings and enhance the performance of LLMs in specialized IE tasks, various in-domain learning strategies are employed. These strategies are designed to equip the LLM with a deeper understanding of the task at hand, essentially guiding the model towards more accurate identification and classification of relevant text spans. Among these strategies, optimizing the task description plays a crucial role. A well-crafted, precise task description can significantly improve the model's focus and comprehension of the task's objectives, leading to more relevant and accurate outcomes.

Furthermore, the provision of speaking, prototypical examples serves as another effective strategy. By presenting the model with clear, illustrative examples that encapsulate the essence of the task, we can anchor its understanding and enhance its ability to generalize from these examples to new, unseen instances. This approach leverages the model's inherent learning capabilities, allowing it to draw parallels and apply learned concepts to the task at hand.

Additionally, augmenting the model's capabilities with Retrieval Augmented Generation (RAG) introduces a powerful dimension to the information extraction process. RAG combines the generative prowess of LLMs with the specificity and relevance of retrieved documents, enabling the model to access a broader context and detailed examples that are pertinent to the task. This strategy is particularly advantageous in domain-specific applications, where the relevance and accuracy of the information extracted are paramount.

In our approach, we capitalize on the last strategy, RAG, to maximize the utility of training sets for each specific task. The retrieval component of this strategy entails identifying instances within the training set that are similar to the test instance and can provide valuable insights for the identification and classification of relevant information. This method not only enhances the model's performance by providing it with task-relevant data but also ensures that the information extracted is of high relevance and quality, tailored to the specific demands of the domain-specific NER task. Through these tailored strategies, we aim to bridge the gap between the broad capabilities of LLMs and the precise requirements of domain-specific information extraction tasks, paving the way for more effective and efficient utilization of generative language models in specialized domains.

4.1 Challenges in Applying LLMs to NER Tasks

The integration of generative LLMs into NER tasks introduces a set of unique challenges that can significantly impact the performance and reliability of extraction outcomes. One of the most prevalent issues in generative NER approaches is the phenomenon of hallucination [10], where the model generates entities not present in the test instance. This can result from the model misinterpreting the provided examples as part of the text from which entities should be extracted, leading to inaccuracies and inconsistencies in the results. n Furthermore, matching problems during the location of mention positions present considerable challenges, particularly in the context of span-based evaluation systems. These systems evaluate the accuracy of entity extraction based on the precise span of text identified as an entity. Discrepancies in the extracted span-whether through corrected spellings or variations in representation-can complicate the matching process. For example, the extraction of "jquery" as "jQuery" illustrates a common scenario where the prevalent spelling of a software library may differ from its mention in the text, yet both are correct. This variability necessitates sophisticated matching strategies to ensure accuracy in evaluation.

The situation is further complicated by texts containing multiple mentions of a single entity. The challenge lies in determining whether each mention can be accurately classified as an entity, alongside dealing with the ambiguity of different entities that are written in the same way. Overlapping or nested entity mentions that do not align with the ground truth data introduce additional layers of complexity, requiring nuanced approaches to entity recognition and classification. Our proposed baseline solution, filtering out non-matching entities and employing rule-based decisions for handling multiple matches have been adopted. However, the disadvantage of this method is that it is based on simplistic heuristics.

Future work may explore advanced solutions like using LLMs for precise entity matching, offering potential improvements for NER challenges. While this paper does not delve into these complex methods, it highlights the importance of ongoing research to further refine and improve LLMs for more accurate and efficient entity extraction.

4.2 Sample Retrieval for RAG on Various IE-Tasks

Retrieval-Augmented Generation (RAG) significantly bolsters the capabilities of LLMs in IE tasks by effectively utilizing both unstructured and structured contexts [3,4]. This dual approach is essential in IE for achieving precise extractions, yet the selection of optimal samples for the generative process poses a substantial challenge. Addressing this involves two primary strategies: utilizing sentence embeddings to find contextually similar textual content for the LLM, and identifying analogous entities to uncover beneficial training sentences. These methods enable the LLM to discern structural and semantic patterns for more accurate text extractions. A crucial obstacle is accurately identifying target entities within test instances, for which we employ a pre-trained Language Model (PLM) tailored to our NER task. This PLM is instrumental in both spotting potential entity candidates and facilitating entity similarity searches, leveraging last hidden state embeddings from training examples to locate matching entities within the dataset.

Our methodology extends to evaluating various retrieval techniques and their impact on the LLM's efficiency, particularly within a Few-Shot learning framework. We explore different methods, including random illustrative samples, text similarity-based RAG, and entity-based sentence retrieval, to provide the LLM with contextually relevant examples, thereby optimizing the software entity extraction process from scientific texts. This exploration aims to identify the most effective strategies for utilizing LLMs in domain-specific software entity extraction tasks. By overcoming challenges related to resource demand, execution time, and precision in entity identification, our approach aims to enhance the accuracy and efficiency of NER processes, contributing valuable insights and methodologies to the field of computational linguistics and information extraction.

A special retrieval method is used for the RE Task. Because for this task, the entities for each test instance are given, we easily could list all possible relations (compare Table 3). We could use them to find similar relations in the train set. If we find more then one, we decide for the one, with highest sentence similarity.

For the RE Task, a special retrieval method is employed. As entities for each test instance are provided, all possible relations can be listed based on Table 3. These relations can then be used to find relations of the same type, and with the same domain and range in the train set. If multiple similar relations are found, the one with the highest sentence similarity is selected.

4.3 Extraction of Software Entities

The extraction of software entities from scientific texts represents a specialized challenge within the realm of NER, targeting the identification of software entities across four distinct categories ranging from applications to operating systems. Furthermore, this task extends beyond mere identification, seeking to understand the intent behind each mention of software entities-whether it pertains to creation, usage, deposition, or mere citation.

We address this challenging task using LLMs, despite their high demands on resources and time, especially when processing extensive publications with sparse relevant text. Our approach includes a pipeline strategy that prioritizes selecting relevant text passages for LLM analysis, improving efficiency by filtering out unrelated content. This method's success depends on the selection accuracy, directly impacting recall. However, the trade-off for reduced computational costs justifies the potential minor decrease in recall. Our performance optimization employs a hybrid method, combining a fine-tuned NER model for sentence selection with LLMs for information extraction. This approach faces limits, notably when sentences crucial for analysis are missed in the selection phase, capping the LLM extraction phase's accuracy as indicated by an initial sentence classification task recall of 0.882 (0.884 F1). This establishes a theoretical limit on extraction accuracy due to potential false negatives, illustrating a balance between efficiency and the precision constraints of LLMs in detailed text analysis.

4.4 Extraction of Software Attributes

Following the identification of software entities within scientific texts, a further nuanced aspect of NER and IE tasks emerges in the extraction of associated software attributes. These attributes encompass a wide array of specific details, the version, developer, citations, URLs, release dates, abbreviations, licenses, extensions, software co-references, and alternative names. We used a similar approach as for subtask 1, and utilised train sample retrieval to augment the task description in a few shot setup. For each sample, including those derived from few-shot learning, the process entails presenting the sentence containing the software entity(ies) and then predicting a JSON list of identified entities along with their respective attribute types.

4.5 Relation Extraction as Single-Choice Question Answering Task

In the field of Natural Language Processing (NLP), the extraction of relations between entities within a text corpus poses significant challenges. This study proposes a novel approach by conceptualizing the task of relations extraction as a single-choice question-answering (QA) activity. This method entails generating a comprehensive list of all possible entities within a sentence, drawing from the existing entities and their relationships as delineated in the training dataset. Each potential pair of entities is then evaluated to ascertain if a specific relation, such as "version_of", appropriately links them. For instance, considering the relationship "version_of", a sentence may be formulated as "8 is the version of SPSS", representing a possible relation between the version number and the software entity.

For every sentence in the dataset, this process yields a set of single-choice questions, each positing a potential relationship between entities. These questions are then prompted to a LLM for answering. The LLM's task is to select the most plausible relation from among the given options, thereby facilitating the extraction of accurate entity relations from the text. However, this approach is not without its challenges. A primary source of error stems from instances where multiple relations could plausibly link a pair of entities, leading to ambiguity and complicating the single-choice question-answering framework. Despite the challenges, we demonstrate that treating relation extraction as a single-choice QA task provides a structured and innovative approach to extracting valuable insights from complex textual data.

5 Experiments

5.1 Models

Fine-tuned Model. In our experimental setup, the initial phase focused on fine-tuning a language model specifically for Subtask 1, which involved NER. Following the methodology similar to Schindler [7], we employed the SciBERT model [1], given its pre-training on scientific corpus, making it apt for NER fine-tuning within scholarly texts. The fine-tuning process entailed rigorous parameter optimization, including adjustments to the batch size, learning rate, and the relative share of negative samples-sentences that do not contain any annotations. This optimization utilized a 90/10 train/evaluation data split from the available training dataset. Subsequent to parameter tuning, we conducted a final training run with a modified data split of 95/5 train/evaluation to maximize the training data's utility.

Generative Large Language Models. For a comparative analysis, our study integrated LLMs, specifically examining the performance differences between GPT-3.5-fast and GPT-4-fast models accessed via the OpenAI API. To ensure deterministic outputs for comparison, we set the temperature parameter to zero, eliminating randomness in the model's response generation.

5.2 Prompting

Software NER. The prompting strategy for Software NER involved providing a concise task description, which included specifying the task as NER and intention classification, alongside an enumeration of target labels. We also highlighted the domain specificity of the texts (i.e., scientific publications) and requested the output in JSON format, delineating separate labels for entity type and intention. Sample sentences from the training set, along with their corresponding JSON output, were included as illustrative examples. This setup varied in the number and order of displayed examples based on the retrieval method, exploring the impact of these factors on model performance through separate experiments.
Attributive NER. For Attributive NER, the prompt construction mirrored the approach taken in Subtask 1 but incorporated more detailed rules, emphasizing the necessity for attributes to relate directly to software entities. Known software entities were additionally provided as input, even for test instances, adhering to the oracle setup defined in the shared task guidelines. This method aimed to refine the model's ability to extract and classify attributive information accurately.
Relation Extraction. The approach to Relation Extraction reimagined the task as a series of single-choice question-answering challenges. Each potential relation, given the software and attributive entities within a sentence, was listed, with claims formulated for each possible relationship (e.g.,"IBM is the developer of Windows"). The model was tasked with identifying the veracity of each claim through a single-choice question format, where all claims were enumerated and solutions provided in a batch format for each example sentence. This setup culminated in presenting the test instance alongside its single-choice questions, expecting the model to deliver decisions on the relational claims.

5.3 Train Sample Retrieval for Few-Shot Generation

Our experimental framework explored various methods for test sample retrieval to ascertain the most effective approach in enhancing model performance. These methods included the use of random illustrative samples to represent every possible signature, retrieval based on entity similarity, and retrieval based on sentence similarity. Each method was evaluated against a baseline to determine its impact on the accuracy and efficiency of the information extraction tasks at hand.

5.4 Relation Extraction Baseline

In the development of our baseline for relation extraction, we established a robust heuristic framework derived from an analysis of potential relations indicated within the text. Our strategy was guided by two principal rules aimed at simplifying the decision-making process for identifying accurate relationships between entities. Firstly, we limited our consideration to relations that necessitate the presence of at least one related entity, deliberately excluding optional inter-software entity relations such as "specification_of" and "PlugIn_of." Given

Table 1. Results of Different Models and Retrieval Methods on Subtask 1

Paradigm	Retrieval	F1	Model	parameter
Finetuned	–	0.599	SciBERT	–
Prompt	Random	0.483	GPT 3–5	Random k = 7
Prompt	Random	0.525	GPT 3–5	Random all entity types shown
Prompt	Sim. sentences	0.647	GPT 3.5	topn = 10
Prompt	Sim. entities	0.624	GPT 3.5	topn = 7 x n entities
Prompt	Random	0.574	GPT 4	Random all entity types shown
Prompt	Sim. sentences	0.677	GPT 4	topn = 10
Prompt	Sim. entities	**0.679**	GPT 4	topn = 7 x n entities

the infrequent occurrence of these cases within the training dataset, we anticipated only a minimal impact on the overall performance metric, specifically the weighted mean macro F1 score for the subtask. Secondly, for all remaining relation types, our approach favored selecting the closest possible entity positioned to the left of the focal software entity as the most likely relation partner. This heuristic was not only straightforward but proved to be highly effective, aligning our baseline performance with that of the top contenders in the shared task. This methodology highlights the potential of leveraging simple, rule-based strategies to achieve competitive results in complex relation extraction challenges.

6 Results

Our analysis in Subtask 1 (Software NER) shows a varied performance landscape across different models and retrieval methods (Table 1). The finetuned baseline, which uses SciBERT, achieved a solid foundation with a 59.9% F1 score. However, LLMs that use random samples without fine-tuning showed a decrease in performance, with the highest F1 score reaching only 57.4%.

A closer examination of retrieval-based models indicates that LLMs perform better. The highest F1 score of 67.9% was achieved by sentence similarity retrieval models, while entity retrieval showed the best performance at 67.7% F1. The transition from GPT 3.5 to GPT 4 models resulted in a significant improvement of approximately 3–5%, although it required around three times more computation time. Notably, our best models were able to perform within a mere 3% below the theoretical maximum by utilizing SciBERT for sentence selection, leveraging oracle positive sentences.

In Subtask 2 (Attributive NER), our methodology showed a significant improvement of +10% in F1 performance compared to our competitors, demonstrating the effectiveness of our approach in a low data regime (Table 2). For Subtask 3 (Relation Extraction), our LLM Single-Question Answering model further improved the F1 score by 5.1%, building on the already competent performance of the heuristic baseline and highlighting the advantage of our method.

Table 2. SOMD Performance Rankings (Weighted Average Macro)

Task	#	User	F1	Precision	Recall	Task	#	User	F1	Precision	Recall
1	1	**phinx**	**.740**	**.761**	**.750**	2	1	***ours***	***.838***	***.835***	***.847***
1	2	david-s477	.692	.739	.711	2	2	phinx	.743	.745	.748
1	3	ThuyNT03	.678	.729	.649	3	1	***ours***	***.916***	*.911*	***.924***
1	4	*ours*	*.652*	*.679*	*.664*	3	2	phinx	.897	.900	.897
1	5	vampire	.648	.682	.637	3	–	*baseline*	*.864*	*.857*	*.875*
						3	–	*necessary*	*.562*	***.933***	*.415*

Furthermore, it has been demonstrated that using 7–10 samples is the most effective strategy, as it optimises the balance between input complexity and model performance.

This analysis highlights the potential of utilising advanced LLM techniques and carefully selected retrieval methods to significantly improve the accuracy and efficiency of NER tasks in specialised domains.

7 Conclusion

Our research on enhancing Relation Extraction (RE) with LLMs through Single-Choice QA has introduced a novel intersection of methodologies aimed at improving the precision of information extraction in the context of scientific texts. By integrating Retrieval-Augmented Generation (RAG) with LLMs and adopting a methodical approach to fine-tuning and leveraging large language models such as SciBERT and use the model to support GPT variants, we have demonstrated the capability of LLMs to navigate the complexities inherent in the extraction of software entities and their attributes.

The exploration of different retrieval strategies-ranging using entity and sentence similarity underscores our commitment to refining the inputs for generative models, ensuring that they are fed the most relevant and contextually appropriate data. This meticulous preparation has allowed us to significantly boost the performance of LLMs in recognising nuanced distinctions among software-related entities and accurately extracting relation types within scholarly articles. Our experiments have not only highlighted the efficacy of LLMs in addressing domain-specific tasks with a limited set of examples but also revealed the inherent challenges, such as the difficulties in matching entity mentions accurately. Despite these hurdles, our single-choice QA approach for RE lead to a strong heuristic baseline for relation extraction and show how altering the task simplifies the problem.

The outcomes of our research indicate a promising direction for future work in leveraging LLMs for NER and RE tasks. The development of our system for participation in the SOMD shared task has illustrated the potential of a single-choice QA approach to relation extraction, offering a structured and scalable method for extracting meaningful insights from textual data. Our findings contribute to the growing body of knowledge on the application of generative models in the field of computational linguistics, paving the way for more sophisticated and efficient methodologies in information extraction from scientific literature.

Acknowledgements. We thank the anonymous reviewers for their constructive feedback. This work has been partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) as part of the Projects BERD@NFDI (grant number 460037581) as well as NFDI4DS (grant number 460234259).

A Dataset Overview

Each subtask in the SOMD Shared Task is evaluated independently. Subtask 2's attributive NER uses the identified software entities from Subtask 1 as inputs, and the annotations from the first two subtasks are combined for Subtask 3's Relation Extraction. The dataset is hierarchically structured, with each subtask's test set forming a subset of the previous one. This lead to data leakage in Subtask 1 and to a lesser extent in Subtask 2. This design ensures that tasks are interconnected, but it requires careful interpretation of results, particularly with respect to error propagation. Although this design creates data leakage, it ensures the interconnectedness and cumulative nature of the challenges. The initial presentation of the Corpus is as the SoMeSci knowledge graph [7]. It is a comprehensive dataset consisting of 1,367 documents and containing 399,942 triples that represent 47,524 sentences. The dataset includes 2,728 software entities, totaling 7,237 labeled entities. The dataset includes positive and negative examples, i.e. sentences without software mentions, which improves the accuracy of the model. The presence of duplicate sentences, headings, and varying sentence lengths increases the complexity of the text and makes extraction tasks from scientific texts more challenging. For possible domains and ranges of the relations int the train set compare Table 3.

B Similarity Search Examples

B.1 Search by Entity Similarity

Table 3. Object Properties, Domain, and Range Classes with Inner Software Relations.

Range Class ⇒ Object Property	Application	Operating System	PlugIn	Programming Environment
Abbreviation ⇒ Abbreviation_of	✓	✓	✓	✓
Developer ⇒ Developer_of	✓	✓	✓	✓
Release ⇒ Release_of	✓	✓	✓	✓
Version ⇒ Version_of	✓	✓	✓	✓
Citation ⇒ Citation_of	✓	×	✓	✓
PlugIn ⇒ PlugIn_of	✓	×	✓	✓
URL ⇒ URL_of	✓	×	✓	✓
Extension ⇒ Extension_of	✓	✓	×	×
License ⇒ License_of	✓	×	✓	×
AlternativeName ⇒ AlternativeName_of	✓	×	×	×
Application ⇒ PlugIn_of ⇒ Specification_of	✓	×	×	×
ProgrammingEnvironment ⇒ Specification_of	×	×	×	✓

Table 4. Retrieval example based on entity similarity for RAG. The first row is the query entity. "sim" reflects the cosine similarity of the enities.

entity	label	sim	split	sentence
PhosphOrtholog	Application Creation	1.00	test	To this end , we have developed an automated web - based tool, PhosphOrtholog , which allows batch processing and mapping of large species - specific PTM datasets to compare overlap at a site - specific level
SNPdetector	Application Creation	0.93	train	We developed a software tool, SNPdetector, for automated identification of SNPs and mutations in fluorescence - based resequencing reads
ESPRIT - Forest	Application Creation	0.92	train	In this paper we developed a new algorithm called ESPRIT - Forest for parallel hierarchical clustering of sequences

B.2 Search by Sentence Similarity

Table 5. Retrieval example based on sentence similarity for RAG. The first row is the query entity. "sim" reflects the cosine similarity of the sentences. No information about the annotated entities are given for retrieval. The example sentence is the same as in Table 4

split	sim	text
test	1.00	To this end , we have developed an automated web - based tool, PhosphOrtholog , which allows batch processing and mapping of large species - specific PTM datasets to compare overlap at a site - specific level
train	0.99	Here we present a tool , Podbat (Positioning database and analysis tool) , that incorporates data from various sources and allows detailed dissection of the entire range of chromatin modifications simultaneously
train	0.99	We designed and developed a new method , MSACompro, to synergistically incorporate predicted secondary structure, relative solvent accessibility , and residue - residue contact information into the currently most accurate posterior probability - based MSA methods to improve the accuracy of mult
train	0.99	In this paper , we present a lossless compression tool, MAFCO, specifically designed to compress MAF (Multiple Alignment Format) files

C Prompting Examples

See Figs. 1, 2 and 3.

```
# Task: Exctract entities in a closed NER setting and classify their intention.
## Entity Types (closed setting):
The entity types in this closed setting are: ['Application', 'ProgrammingEnvironment', 'SoftwarePackageOrPlugin', 'OperatingSystem', 'SoftwareCoreference']
Do not include version information for the entities.
if no entity of the given entity types in the context return an empty list

## Intention Classification:
For each entity mention classify the intention of the usage in the context.
Use only and exact one intention per entity.
The intention classes are: ['Usage', 'Deposition', 'Creation', 'Mention']

## The Contexts:
The contexts to extract the entities are about software and from scientific publications.

## Output Format:
Return the result in a json list of objects.

# Examples:
## Context:
"""We developed a software tool , SNPdetector , for automated identification of SNPs and mutations in fluorescence - based resequencing reads ."""
## Detected Named Entities with Intention:
[ {
   "text": "SNPdetector",
   "label": "Application",
   "intention": "Creation" } ]
## Context:
"""In this paper we developed a new algorithm called ESPRIT - Forest for parallel hierarchical clustering of sequences ."""
## Detected Named Entities with Intention:
[ {
   "text": "ESPRIT␣-␣Forest",
   "label": "Application",
   "intention": "Creation" } ]
## Context:
"""To this end , we have developed an automated web - based tool , PhosphOrtholog , which allows batch processing and mapping of large species - specific PTM datasets to compare overlap at a site - specific level ."""
## Detected Named Entities with Intention:
```

Fig. 1. Software NER Few-Shot prompt (n = 2). The shown sample is the same as in the similarity examples in Tables 4 and 5.

```
# Task:
## Task Description
Exctract attributive text spans about software related entities in a closed attibutive NER setting.

## The Contexts:
The contexts to extract the attributive text spans are about software and from scientific publications.

## The Software Related Entities:
The software related entities are given as a list of json object with mention and entity type.
Do not annotate any related software entities.

## Attribute Types (closed setting):
The attribute types to identify in this closed setting are:
 ['Version', 'Developer', 'Citation', 'URL', 'Release', 'Abbreviation', 'License', 'Extension', 'AlternativeName']
Keep the "Version" note separate, without overlap with other entities or attribute spans.
No span overlaps are allowed inbetween attributive spans of to the given software related entites.
Note that the attribute types must be associated with a software-related entity of one of the following types: ['Application', 'ProgrammingEnvironment', 'PlugIn', 'OperatingSystem', 'SoftwareCoreference'].
To help you, you will receive all software-related entities as json for each context.

## Output Format:
Return the result in a json list of objects.

# Examples:
## Context:
"""LAMINA uses the Java Advanced Imaging ( JAI ) package http://java.sun.com/javase/technologies/desktop/media/jai/ to support common image file formats , which is bundled with the installation and hence no additional installation should be required ."""
### Detected Software Related Named Entities:
[{
   "text": "LAMINA",
   "label": "Application"
 }, {
   "text": "Java␣Advanced␣Imaging",
   "label": "Application"}]

### Detected Attributive Text Spans:
[{
   "text": "JAI",
   "label": "Abbreviation"
 }, {
   "text": "http://java.sun.com/javase/technologies/desktop/media/jai/",
   "label": "URL"}]
```

Fig. 2. Attibutive NER Few-Shot prompt (n = 2).

```
# Answer the following single-choice questions on sentences from scientific publications on the subject of software.
 * Please note that there is exactly one answer to each question.
## Sentence:
"""The web service [44] available at http://www.fragit.org enables users to upload their structure , fragment it and download the resulting input file to GAMESS ."""
### Question(s):
Which of the statements about the Citation '[44]' is true?
idx_0: '[44]' is the citation of 'web␣service'.
idx_1: '[44]' is the citation of 'GAMESS'.

Which of the statements about the URL 'http://www.fragit.org' is true?
idx_2: 'http://www.fragit.org' is the url of 'web␣service'.
idx_3: 'http://www.fragit.org' is the url of 'GAMESS'.

### Answers to the questions about the sentence(s):
idx_0: True
idx_1: False
idx_2: True
idx_3: False

## Sentence:
"""ASPASIA , released under the Artistic License ( 2.0 ) , can be downloaded from http://www.york.ac.uk/ycil/software ."""
### Question(s):
Which of the statements about the Version '2.0' is true?
idx_4: '2.0' is the version of 'ASPASIA'.
idx_5: '2.0' is the version of 'Artistic␣License'.

Which of the statements about the URL 'http://www.york.ac.uk/ycil/software' is true?
idx_6: 'http://www.york.ac.uk/ycil/software' is the url of 'ASPASIA'.
idx_7: 'http://www.york.ac.uk/ycil/software' is the url of 'Artistic␣License'.

### Answers to the questions about the sentence(s):
idx_4: False
idx_5: True
idx_6: True
idx_7: False

## Sentence:
"""The web - tool can be accessed from http://www.phosphortholog.com / using any typical web browser ( except Internet Explorer ) . """

### Question(s):
Which of the statements about the URL 'http://www.phosphortholog.com/' is true?
idx_8: 'http://www.phosphortholog.com/' is the url of 'web␣-␣tool'.
idx_9: 'http://www.phosphortholog.com/' is the url of 'Internet␣Explorer'.

### Answers to the questions about the sentence(s):
```

Fig. 3. Single-Choice QA prompt for Relation Annotation (n = 2).

References

1. Beltagy, I., Lo, K., Cohan, A.: Scibert: a pretrained language model for scientific text. In: Conference on Empirical Methods in Natural Language Processing (2019). https://doi.org/10.18653/v1/d19-1371
2. Du, C.F., Cohoon, J., Lopez, P., Howison, J.: Softcite dataset: a dataset of software mentions in biomedical and economic research publications. J. Am. Soc. Inf. Sci. **72**, 870–884 (2021). https://doi.org/10.1002/asi.24454
3. Gao, Y., et al.: Retrieval-augmented generation for large language models: a survey (2024). https://doi.org/10.48550/arXiv.2312.10997
4. Lewis, P., et al.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 9459–9474. Curran Associates, Inc. (2020)
5. Otto, W., Zloch, M., Gan, L., Karmakar, S., Dietze, S.: GSAP-NER: a novel task, corpus, and baseline for scholarly entity extraction focused on machine learning models and datasets. In: Bouamor, H., Pino, J., Bali, K. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2023. pp. 8166–8176. Association for Computational Linguistics, Singapore, December 2023. https://doi.org/10.18653/v1/2023.findings-emnlp.548, https://aclanthology.org/2023.findings-emnlp.548
6. Saji, A., Matsubara, S.: Extracting information about research resources from scholarly papers. In: International Conference on Asian Digital Libraries, pp. 440–480 (2022). https://doi.org/10.1007/978-3-031-21756-2_35
7. Schindler, D., Bensmann, F., Dietze, S., Krüger, F.: Somesci- a 5 star open data gold standard knowledge graph of software mentions in scientific articles. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4574–4583 (2021). https://doi.org/10.1145/3459637.3482017
8. Wadhwa, S., Amir, S., Wallace, B.: Revisiting relation extraction in the era of large language models. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 15566–15589. Association for Computational Linguistics, Toronto, Canada, July 2023. https://doi.org/10.18653/v1/2023.acl-long.868, https://aclanthology.org/2023.acl-long.868
9. Wan, Z., et al.: GPT-RE: in-context learning for relation extraction using large language models. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp. 3534–3547. Association for Computational Linguistics, Singapore, December 2023. https://doi.org/10.18653/v1/2023.emnlp-main.214, https://aclanthology.org/2023.emnlp-main.214
10. Wang, S., et al.: GPT-NER: named entity recognition via large language models (2023). https://doi.org/10.48550/arXiv.2304.10428
11. Xie, T., Li, Q., Zhang, Y., Liu, Z., Wang, H.: Self-improving for zero-shot named entity recognition with large language models (2024). https://doi.org/10.48550/arxiv.2311.08921
12. Xu, D., et al.: Large language models for generative information extraction: a survey. ArXiv (2023). https://doi.org/10.48550/arxiv.2312.17617

Author Index

G. Rehm et al. (Eds.): NSLP 2024, LNAI 14770, pp. 307–308, 2024.
https://doi.org/10.1007/978-3-031-65794-8