BUILDING KNOWLEDGE GRAPHS FROM MESSY ENTERPRISE DATA

Thesis approved by
the Department of Computer Science
Technische Universität Kaiserslautern
for the award of the Doctoral Degree
Doctor of Engineering (Dr.-Ing.)

to

Markus Schröder

Date of Defense: 2022-12-06
Dean: Prof. Dr. Jens Schmitt
Reviewer: Prof. Dr. Prof. h.c. Andreas Dengel
Reviewer: Prof. Dr. Maria-Esther Vidal
Reviewer: Jun.-Prof. Dr. Sophie Fellenz

DE-386

Markus Schröder: *Building Knowledge Graphs from Messy Enterprise Data*

REVIEWERS:
Prof. Dr. Prof. h.c. Andreas Dengel
Prof. Dr. Maria-Esther Vidal (Leibniz Universität Hannover)
Jun.-Prof. Dr. Sophie Fellenz

ABSTRACT

Undocumented enterprise data can easily pile up in companies in form of datasets and personal information. In absence of a data management strategy, such data becomes rather messy and may not fit for its intended use. Since there is often no documentation available, only a limited number of domain experts are aware of its contents. Therefore, for companies it becomes increasingly difficult to use such data to its full potential. To provide a solution, this PhD thesis investigates the construction of enterprise and personal knowledge graphs by semantically enriching messy data with meaning using semantic technologies. Since real world entities and their interrelations are organized in a graph, knowledge graphs serve as a semantic bridge between domain conceptualization and raw data.

Spreadsheets are a prominent example of such enterprise data, since they are widely used by knowledge workers in the industrial sector. Two distinct approaches are investigated to construct knowledge graphs from them: a global extraction & annotation method and a local mapping technique. The latter is further complemented with a predictor of mapping rules on messy data.

Different human-in-the-loop strategies are considered to include experts depending on their user group. Since non-technical users usually lack understanding of semantic technologies, they need appropriate tools to be able to give feedback. In case of developers, approaches are proposed to close the technology gap between industry and Semantic Web related concepts. Semantic Web practitioners participate with ontology modeling and linked data applications.

Enterprise and personal data is typically confidential which is why it cannot be shared with a research community to discuss its challenges. However, for evaluation and reproducibility reasons publicly available datasets are mandatory. The thesis proposes ways to generate synthetic datasets with the goal to be as authentic as possible. Besides that, for internal evaluations a crawler of personal data on desktops is implemented.

There are further contributions related to this thesis in diverse domains. One is about the motivation to support users in their daily work using personal knowledge assistants. Others are the agricultural field and the data science domain which also benefit from knowledge graph approaches.

In conclusion, this PhD thesis contributes to the construction of knowledge graphs from especially messy enterprise data, while users from different groups take part in this process in various ways.

PUBLICATIONS AS PART OF THIS THESIS

Parts of the research and material (including figures, tables, listings and algorithms) in this PhD thesis have already been published in:

- M. Schröder, C. Jilek, J. Hees, S. Hertling, and A. Dengel. "RDF Spreadsheet Editor: Get (G)rid of Your RDF Data Entry Problems." In: *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*. Vol. 1963. CEUR Workshop Proceedings. CEUR-WS.org, Oct. 2017. URL: `http://ceur-ws.org/Vol-1963/paper635.pdf`.
- M. Schröder, J. Hees, A. Bernardi, D. Ewert, P. Klotz, and S. Stadtmüller. "Simplified SPARQL REST API - CRUD on JSON Object Graphs via URI Paths." In: *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*. Vol. 11155. Lecture Notes in Computer Science. Springer, June 2018, pp. 40–45. DOI: `10.1007/978-3-319-98192-5_8`. arXiv: `1805.01825`.
- M. Schröder, C. Jilek, and A. Dengel. "Deep Linking Desktop Resources." In: *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*. Vol. 11155. Lecture Notes in Computer Science. Springer, June 2018, pp. 202–207. DOI: `10.1007/978-3-319-98192-5_38`. arXiv: `1805.03491`.
- M. Schröder, C. Jilek, J. Hees, and A. Dengel. "Towards Semantically Enhanced Data Understanding." In: *CoRR* abs/1806.04952 (2018). arXiv: `1806.04952`.
- M. Schröder, C. Jilek, J. Hees, S. Hertling, and A. Dengel. "An Easy & Collaborative RDF Data Entry Method using the Spreadsheet Metaphor." In: *CoRR* abs/1804.04175 (2018). arXiv: `1804.04175`.
- M. Schröder. "Efficient High-Level Semantic Enrichment of Undocumented Enterprise Data." In: *The Semantic Web: ESWC 2019 Satellite Events - ESWC 2019 Satellite Events, Portorož, Slovenia, June 2-6, 2019, Revised Selected Papers*. Vol. 11762. Lecture Notes in Computer Science. Springer, June 2019, pp. 220–230. DOI: `10.1007/978-3-030-32327-1_41`.
- M. Schröder, C. Jilek, and A. Dengel. "Interactive Concept Mining on Personal Data - Bootstrapping Semantic Services." In: *CoRR* abs/1903.05872 (2019). arXiv: `1903.05872`.
- J. Klose, M. Schröder, S. Becker, A. Bernardi, and Arno Ruckelshausen. "Datenaufbereitung in der Landwirtschaft durch automatisierte semantische Annotation." In: *40. GIL - Jahrestagung, Informatik in der Land-, Forst- und Ernährungswirtschaft, Fokus: Digitalisierung für Mensch, Umwelt und Tier, 17. - 18. Februar 2020, Campus Weihenstephan, Freising, Germany*. Vol. P-299. LNI. Gesellschaft für Informatik e.V., Feb. 2020, pp. 133–138. arXiv: `1911.06606`. URL: `https://dl.gi.de/20.500.12116/31882`.

- M. Schröder. *A Pattern Language for Spreadsheets*. Mar. 1, 2021. URL: https://www.dfki.uni-kl.de/~mschroeder/pattern-language-spreadsheets/.
- M. Schröder, C. Jilek, and A. Dengel. "A Linked Data Application Framework to Enable Rapid Prototyping." In: *CoRR* abs/2104.13605 (2021). arXiv: 2104.13605.
- M. Schröder, C. Jilek, and A. Dengel. "Dataset Generation Patterns for Evaluating Knowledge Graph Construction." In: *The Semantic Web: ESWC 2021 Satellite Events - Virtual Event, June 6-10, 2021, Revised Selected Papers*. Vol. 12739. Lecture Notes in Computer Science. Springer, June 2021, pp. 27–32. DOI: 10.1007/978-3-030-80418-3_5. arXiv: 2104.13576.
- M. Schröder, C. Jilek, and A. Dengel. "Mapping Spreadsheets to RDF: Supporting Excel in RML." In: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*. Vol. 2873. CEUR Workshop Proceedings. CEUR-WS.org, June 2021. arXiv: 2104.13600. URL: http://ceur-ws.org/Vol-2873/paper3.pdf.
- M. Schröder, C. Jilek, and A. Dengel. "Spread2RML: Constructing Knowledge Graphs by Predicting RML Mappings on Messy Spreadsheets." In: *K-CAP '21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021*. Best Research Paper. ACM, Dec. 2021, pp. 145–152. DOI: 10.1145/3460210.3493544. arXiv: 2110.12829.
- M. Schröder, C. Jilek, M. Schulze, and A. Dengel. "Interactively Constructing Knowledge Graphs from Messy User-Generated Spreadsheets." In: *CoRR* abs/2103.03537 (2021). arXiv: 2103.03537.
- M. Schröder, C. Jilek, M. Schulze, and A. Dengel. "The Person Index Challenge: Extraction of Persons from Messy, Short Texts." In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 2, Online Streaming, February 4-6, 2021*. SCITEPRESS, Feb. 2021, pp. 531–537. DOI: 10.5220/0010188405310537. arXiv: 2011.07990.
- M. Schröder, M. Schulze, C. Jilek, and A. Dengel. "Bridging the Technology Gap between Industry and Semantic Web: Generating Databases and Server Code from RDF." In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 2, Online Streaming, February 4-6, 2021*. SCITEPRESS, Feb. 2021, pp. 507–514. DOI: 10.5220/0010186005070514. arXiv: 2011.07957.
- M. Schröder. *Hephaistos Toolkit*. Apr. 14, 2022. URL: https://www.dfki.uni-kl.de/~mschroeder/hephaistos/.
- M. Schröder, C. Jilek, and A. Dengel. "A Human-in-the-Loop Approach for Personal Knowledge Graph Construction from File Names." In: *Proceedings of the 3rd International Workshop on Knowledge Graph Construction (KGCW 2022) co-located with 19th Extended Semantic Web Conference (ESWC 2022), Hersonissos, Greek, May 30, 2022*. Vol. 3141. CEUR Workshop Proceedings. CEUR-WS.org, 2022. URL: http://ceur-ws.org/Vol-3141/paper2.pdf.

ACKNOWLEDGMENTS

I would like to express deep gratitude to Prof. Andreas Dengel to give me the opportunity to conduct research on the topic. He gave me freedom to investigate in the field and has always been open for discussions and feedback meetings. Further, I would like to thank Prof. Maria-Esther Vidal to write a second review for the thesis. She gave me the opportunity to present my work to her group and discuss the topics with them. Moreover, I would like to thank Jun.-Prof. Sophie Fellenz for a third review and interesting questions during the defense. Last but not least, I would like to thank the whole PhD committee including Prof. Stefan Deßloch for organizing the process at the university.

I would also like to thank Ansgar Bernardi who was a mentor for me when I started at DFKI. He plunged me in at the deep end and showed me how project work is done properly. Since I later changed to the topic field knowledge work, I would like to thank the leader Heiko Maus for his numerous opportunities to apply my research in industry. Additionally, I would like to thank Sven Schwarz to always give technical advice and a practical point of view. Moreover, I would like to thank Prof. Heiko Paulheim during his time at TU Darmstadt for introducing me to Semantic Web technologies with his lecture and the opportunity to write a Bachelor thesis in this field.

On the dissertation journey you are never alone. Thus, I would like to express deep gratitude to Christian Jilek who became a room mate, peer, companion, comrade and friend. He supported me in improving my writing style in papers and gave me always valuable feedback for my research. During the PhD we had a lot of fruitful discussions and drawings on our whiteboard. I also would like to thank Sven Hertling with whom I visited academic high school, university and DFKI until he changed to University of Mannheim. We always supported each other in the challenges we had to overcome. Moreover, I would like to thank Jörn Hees for his expertise in all kinds of questions around Semantic Web. In the end, I would like to thank the DFKI and its always supporting Smart Data & Knowledge Services group, especially all members of the CoMem and SensAI Team.

Regarding my family, I would like to thank my parents Jutta and Karl for always supporting me in my life situations, education and career. I would like to thank my beloved wife Katharina for adding color to my life and for reminding me how to relax in stressful times. Last but not least, I would also like to thank our cat Filou for waking me up early in the morning to keep me on the go.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

AI	Artificial Intelligence	47
API	Application Programming Interface	4
BNode	Blank Node	5
BSBM	Berlin SPARQL Benchmark	69
CRISP-DM	Cross-Industry Standard Process for Data Mining	98
CRUD	Create, Read, Update, and Delete	4
CSS	Cascading Style Sheets	56
CSV	Comma-Separated Values	7
CURIE	Compact URI	5
DSL	Domain Specific Language	17
FnO	Function Ontology	7
FOAF	Friend of a Friend	7
GUI	Graphical User Interface	9
HCI	Human-Computer Interaction	41
HTML	HyperText Markup Language	4
HTTP	Hypertext Transfer Protocol	4
HumL	Human-in-the-Loop	39
IRI	Internationalized Resource Identifier	5
JSON	JavaScript Object Notation	4
KE	Knowledge Engineer	10
KECS	Knowledge Extraction from Classification Schemes	47
KG	Knowledge Graph	5
LD	Linked Data	7
LDAF	Linked Data Application Framework	73
LOD	Linked Open Data	7
MAS	Multi-Agent System	81
NER	Named Entity Recognition	26
OWL	Web Ontology Language	7
P2P-O	Purchase-to-Pay Ontology	81
P2P	Purchase-to-Pay	81
PIM	Personal Information Management	9
PIMO	Personal Information Model	9

PKG	Personal Knowledge Graph	9
POJO	Plain Old Java Object	68
ptpDG	Purchase-To-Pay Dataset Generator	81
R2RML	RDB to RDF Mapping Language	7
RDB	Relational Database	7
RDF	Resource Description Framework	5
RDFS	RDF Schema	6
REST	Representational State Transfer	4
RML	RDF Mapping Language	7
RPC	Remote Procedure Call	5
RQL	Resource Query Language	65
SKOS	Simple Knowledge Organization System	47
SPARQL	SPARQL Protocol and RDF Query Language	5
SPO	Subject, Predicate, Object	5
SQL	Structured Query Language	66
Turtle	Terse RDF Triple Language	6
UEQ	User Experience Questionnaire	61
URI	Uniform Resource Identifier	5
URL	Uniform Resource Locator	4
URN	Uniform Resource Name	5
UUID	Universally Unique Identifier	5
XML	Extensible Markup Language	7
XSD	XML Schema Definition	7

1 INTRODUCTION

1.1 MOTIVATION

In the age of data, digitization has become an integral part in many companies. By drawing benefits from electronic data processing, they increasingly digitizing their work and gather data in dedicated stores. As a consequence, resources naturally pile up as soon as maintenance receives little to no attention and data management strategies are absent. Data easily accumulates in places like intranet or cloud storages and employees' devices in form of personal information. Such a dataspace is typically heterogeneous, arbitrarily structured, diverse and distributed in isolated stores over several years [63]. In a nutshell, data has become "messy" which means it exhibits low quality and is therefore often not fit for its intended use [125]. In daily business messy data can hinder employees to work with it efficiently. Companies which would like to discover and make use of such data face usually many difficulties in this process. For instance, to gain valuable insights, complex data mining analyses are typically performed. However, this is hindered by messy data and make additional steps necessary [161]. This is why data preparation is a usual task performed by data scientists to first clean and organize data [41, p. 7].

To give a small example, Table 1 illustrates a messy spreadsheet. After a closer look, the following questions may arise: What does the

Table 1: An exemplary spreadsheet to demonstrate messy data (already published in [151]).

Document ID	Dep.	Editor	Type	Published	Sent
*AB-ztad.63/23	GA	~~Cooper~~ Smith	C	42415	x
AB-hzyx-78/24	GA/BZ	Emma Thomas	N	TODO	
AB 5-pbga.67	BZ	(new) Smith Thomas, E.	ed.c	15.05.2010	-

inconsistent asterisk sign in a document's ID mean? Why is the editor Cooper crossed out? Does an 'x' symbol indicate the document has been sent or was not sent yet? What date representations are used for the published date? These examples[1] show that the comprehension of the presented data is hampered by its vagueness which is caused by its messiness.

To be able to give answers to such questions, its meaning or, more specifically, its semantics could be made explicit in form of formal

1 More such challenges are discussed in detail in Section 2.1.

statements. Semantic technologies and especially knowledge graphs provide suitable methods to semantically represent and enrich data with meaning [48]. Since such graphs organize real world entities and their interrelations in a graph [114], they could serve as a semantic bridge between domain conceptualization (mostly in people's minds) and raw data (e.g. in storage systems) [133].

However, because of its messiness, there are significant challenges in constructing such graphs. To initially bridge the gap, knowledge and information extraction techniques have to be adapted in order to bootstrap knowledge graphs from typical enterprise data, ranging from semi-structured spreadsheets to hierarchical folder structures. These very special data structures require procedures that exploit the unique nature of their contents. For example, instead of large document collections with natural language texts, short ungrammatical text snippets [80] have to be processed. These information pieces are usually about a special domain, contain personal or technical terms and are not comprehensible with common knowledge alone.

Since there is often insufficient documentation available, the consultation of domain experts becomes necessary. These people work daily with their usual data assets, which is why required knowledge to understand the data is hidden in their mindsets. However, employees have a limited period of time for giving interviews and a small number of them are usually available. Dedicated tools need to be designed to let them formalize their expertise, consume already acquired knowledge and identify where feedback is still required. Thus, depending on their technical skill level, appropriate integration strategies need to be selected and properly scheduled before, during or after knowledge graphs are built.

To assess the quality of constructed knowledge graphs, suitable evaluations can be conducted in research [124]. They measure how well proposed approaches perform on data or in certain scenarios. However, enterprise and personal data is typically confidential which is why it cannot be shared with a research community to discuss its challenges. Yet, publicly available datasets are mandatory for reproducibility reasons. In such cases, the generation of artificial datasets could be an appropriate method, but also entails some challenges. Synthetic generated datasets need to be as authentic as possible to show similar evaluation measures comparable with real-world data. To consider authenticity, a sufficient understanding about the challenges messy data causes in practice is necessary. For internal evaluation only, real-world enterprise data and personal information of employees need to be compiled to datasets, too.

1.2 RESEARCH QUESTIONS

The discussed challenges lead to the following major research questions (RQs).

RQ1: How can knowledge graphs be built from especially messy data?

An answer to this question includes the identification of major challenges messy data causes and proposals for solutions to tackle them. These methods should be able to semi-automatically construct knowledge graphs, while messy data is considered and appropriately processed. Chapter 2 proposes two distinct approaches and shows their benefits and limitations in case studies conducted in an industrial scenario. A third complementary approach predicts mapping rules to reduce effort. Throughout this chapter spreadsheets serve as a prominent representative of messy enterprise data.

RQ2: How can domain experts be integrated in the construction process?

Replying to this question leads to the inclusion of various user groups in different phases of knowledge graph construction. Chapter 3 presents diverse methods to include domain experts depending on their roles in a project. The communication of knowledge formalized with semantic technologies in a comprehensible way to users turns out to be an integral part. Therefore, proposed approaches resort to familiar metaphors, environments and technologies users feel comfortable with. Only then modeling and feedback mechanisms can be applied to enrich or correct evolving knowledge graphs.

RQ3: How can datasets be generated for evaluation purpose?

Special generators are required to produce synthetic datasets which are authentic enough to conduct meaningful experiments. Since messy data is subject of investigation, produced data should also show a similar degree of messiness observed in real data. To be able to calculate evaluation measures, ground truth data has to be provided by generators, too. Chapter 4 proposes methods which consider these requirements. An answer to the question also includes that real-world datasets needs to be “generated” (i.e. collected) to be able to study its characteristics in private experiments.

Before subsequent chapters give answers to these questions, necessary background and a scenario is provided in the next sections.

1.3 BACKGROUND

This section provides background knowledge to the standards and technologies used throughout the PhD thesis. First in Section 1.3.1, the Semantic Web vision and its commonly used standards are introduced for readers who are not familiar with them. Second, a mapping language is covered (Section 1.3.2) which is a key technology in the construction approaches in Chapter 2. Third, concepts of the Semantic Desktop are described in Section 1.3.3. Regarding these topics, a more practical tutorial about semantic technologies has been published online [138].

For a rough temporal classification Figure 1 presents a timeline of notable publications and technologies which will be mentioned in the following.

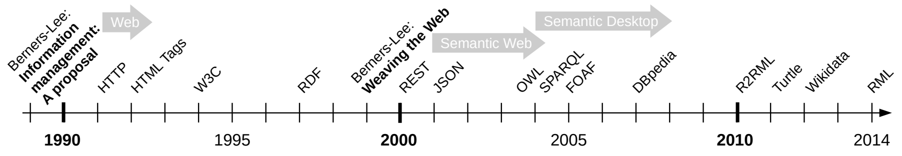


Figure 1: Timeline of publications and technologies mentioned in Section 1.3.

1.3.1 *Semantic Web*

At the beginning of the nineties, the (World Wide) **Web** has been invented as we know it today. Starting with a proposal by Berners-Lee to use hypertext in linked information systems [17], the first implementation of the Hypertext Transfer Protocol (HTTP) [18] and the initial design of the HyperText Markup Language (HTML) [170] came into existence. HTTP specifies certain request methods[2] a client can send to a server: while the `GET` operation transfers a resource to the client, `PUT` replaces and `DELETE` removes a resource's representation. A `POST` operation performs processing on a given payload. The special `PATCH`[3] method is used to apply changes to a resource. Since then people use browsers to retrieve and render Web pages by following Uniform Resource Locators (URLs). In 1994 the World Wide Web Consortium[4] (W3C) was founded to maintain and develop standards for the Web. Later on, Representational State Transfer (REST) was proposed, an architectural style for the Web, to meet its requirements in performance and behavior [59]. Many services in the Web which provide Application Programming Interfaces (APIs) adopted this style by letting users Create, Read, Update, and Delete (CRUD) resources in so-called REST(ful) APIs. With the popularity of the JavaScript Object

2 https://datatracker.ietf.org/doc/html/rfc7231#section-4
3 https://datatracker.ietf.org/doc/html/rfc5789#section-2
4 https://www.w3.org/

Notation (JSON) [39] as a data format[5], it became a de facto standard for services to send and receive content. Interaction with APIs is also possible via Remote Procedure Calls (RPCs), for example with the JSON-RPC standard [93].

At the beginning of the 21st century, the vision of a ***Semantic* Web** was stated by Web creator Berners-Lee [19, Chapter 13] and other researchers [20]. They argued that instead of only human-readable web pages, content should be formalized in a machine-readable way. This will enable agents to understand the well-defined meaning and solve tasks more effectively. To let machines comprehend what is otherwise written in complex natural language sentences, simple and formal statements are provided in the Semantic Web. To describe meta data about Web resources, a formalism has been formed since 1997[6]: the **Resource Description Framework (RDF)** [182]. Each RDF statement consists of the three most important grammatical parts in a sentence: a subject, a predicate and an object. Because the three parts are always present, they are also called triple (statements) or Subject, Predicate, Object (SPO) triples. This standard also includes that these resources have to be identified with Uniform Resource Identifiers (URIs) using a specified syntax[7]. URIs generalize the concepts of Uniform Resource Locators (URLs) and Uniform Resource Names (URNs). Randomly generated Universally Unique Identifiers (UUIDs) are commonly used to form globally unique but not resolvable URNs, like for example `urn:uuid:96e40e3a-22e8-4ec3-939e-d7f9afc3984d`[8]. Sometimes Internationalized Resource Identifiers (IRIs) are used to permit an expanding set of characters[9]. Since all of these identifiers tend to be very long, they are usually shortened by using Compact URIs (CURIEs) [173] where long prefixes are replaced with short names. If a URI is not given for a resource, it is considered to be anonymous, a so-called Blank Node (BNode). Besides resources, the standard also considers literals on the object position to store data values (character strings). They also provide meta data for expressing the data type and a text's language.

RDF statements are stored and managed as a set in a dedicated store: a triplestore. This store is usually accessed and updated with the query language SPARQL Protocol and RDF Query Language (SPARQL) [181]. Since the subjects and objects of triples can also be interpreted as nodes in a directed edge-labeled graph, it is also named an RDF graph and since statements express knowledge, it is also called **Knowledge Graph (KG)**. In this PhD thesis, KGs are characterized similar to Paulheim's definition [114]: they organize entities and their interrelations in a graph with a focus on instances and possibly but

5 https://datatracker.ietf.org/doc/html/rfc8259
6 https://www.w3.org/TR/WD-rdf-syntax-971002/
7 https://datatracker.ietf.org/doc/html/rfc2396
8 https://datatracker.ietf.org/doc/html/rfc4122
9 https://datatracker.ietf.org/doc/html/rfc3987

Listing 1: An example showing the Turtle serialization format.

```
1  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
2  @prefix xsd:  <http://www.w3.org/2001/XMLSchema#> .
3  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
4
5  <https://www.dfki.uni-kl.de/~mschroeder> a foaf:Person ;
6    foaf:knows <https://www.dfki.uni-kl.de/~jilek> ;
7    foaf:dateOfBirth "1990-07-12"^^xsd:date ;
8    foaf:worksAt [
9      a foaf:Organization ;
10     rdfs:label "Deutsches Forschungszentrum für Künstliche Intelligenz"@de ,
11                "German Research Center for Artificial Intelligence"@en
12   ] .
```

not necessarily define a schema. A powerful feature of them is the potential to interrelate any entity with another one and in doing so they may cover several domains.

RDF has many serialization formats [174]. In this PhD thesis, the **Terse RDF Triple Language (Turtle)** [183] format is almost always considered, since it proved to be human-friendly and human-readable. An example of it is shown in Listing 1 which covers the introduced features of RDF. Prefixes are defined first, to be able to use CURIEs in the document (Line 1–3). In Line 5, a triple is defined: it states that a resource identified with a URI surrounded by angle brackets (<...>) represents a person. With the semicolon symbol (;) more statements are made about the same resource. Line 7 shows a literal with a data type indicated by two carets (^^). After that, a BNode is shown with surrounding square brackets ([...]) which represents an organization (Line 8–9). It has two labels (Line 10–11) which are repeated with a comma sign (,) and which are in different languages using the at sign (@). Such an RDF document with Turtle syntax is only one representation a resource can have. Usually, content negotiation[10] is used to serve the representation which is best suited for an agent. For example, while Semantic Web clients consume RDF triples, human users would like to see the same information in rendered HTML pages.

So far, the examples showed statements about persons or things with assertions (also known as ABox). However, RDF is also able to define a terminology which is a vocabulary of a domain of interest (also known as TBox). To model terminologies, the Semantic Web utilizes **ontologies**: according to Gruber an ontology is an "explicit specification of a conceptualization" [68]. In its simplest form, it clarifies what exists with classes and how things relate with properties. Once instances are assigned to classes, the process is known as ontology population [115]. RDF Schema (RDFS) [184] describes in RDF a schema terminology which enables users to model their own

10 https://datatracker.ietf.org/doc/html/rfc2616#section-12

ontologies. A more expressive formalism provides the Web Ontology Language (OWL) [177] which is, however, not required in this PhD thesis. The example in Listing 1 made use of the Friend of a Friend (FOAF) ontology [28] (expressed in RDFS) by referring to the class of all persons (`foaf:Person`) and properties of persons such as the date of birth (`foaf:dateOfBirth`). Ontologies are also used to specify data types like in the XML Schema Definition (XSD) [179]. In the example the definition of a date (`xsd:date`) is used to clarify that its value is formatted to `YYYY-MM-DD`.

Motivated from the Semantic Web vision, many data sources were transformed into **publicly available Knowledge Graphs (KGs)**. Since 2007 the well-known online encyclopedia Wikipedia[11] has been automatically converted into a large RDF graph called DBpedia [10]. Making use of "infobox" tables in Wikipedia pages, RDF statements are formed and semantic resources are interlinked. Another notable KG is Wikidata [165] which is, in contrast to DBpedia, manually created by contributors. These and other KGs naturally link to each other, which is why such data is called Linked Data (LD). Together all (usually open) KGs are summarized under the umbrella term Linked Open Data (LOD) [162].

How such KGs can be constructed from existing data using a formalism is covered in the next section.

1.3.2 *RDF Mapping Language*

A mapping language consists of declarative rules which allow to define how input data is mapped to another representation. Dedicated mapping languages have been designed to be able to make data available as RDF datasets. In case of Relational Databases (RDBs), the RDB to RDF Mapping Language (R2RML) [178] has been proposed. To support also other semi-structured data, its formalism was further extended to the specification of the RDF Mapping Language (RML) [50, 51]. It is able to map the following data formats: Comma-Separated Values (CSV), Extensible Markup Language (XML), and JavaScript Object Notation (JSON). Moreover, functions defined with the Function Ontology (FnO) [105] can be executed during the mapping to transform (or convert) data with a programming language.

In order to demonstrate RML's basic usage, Listing 2 presents an RML mapping which maps the tabular data in Table 1 by assuming its a CSV file. First, a logical source refers to the input CSV file with the appropriate reference formulation in Line 9–10. A subject map (Line 13) turns each CSV record into a resource having a URI as specified in the given template (Line 14). Additionally, these resources are assigned to a document class (Line 15). Using predicate object maps, CSV record values are mapped to object values by referring to the corresponding

11 https://www.wikipedia.org/

Listing 2: A demonstration of an RML mapping for Table 1.

```
 1 @prefix rr: <http://www.w3.org/ns/r2rml#> .
 2 @prefix rml: <http://semweb.mmlab.be/ns/rml#> .
 3 @prefix ql: <http://semweb.mmlab.be/ns/ql#> .
 4 @prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
 5 @prefix foaf: <http://xmlns.com/foaf/0.1/> .
 6 @prefix : <file://table-1.rml.ttl#> .
 7 
 8 :mapping a rr:TriplesMap;
 9   rml:logicalSource [
10     rml:source "table-1.csv" ;
11     rml:referenceFormulation ql:CSV
12   ];
13   rr:subjectMap [
14     rr:template "file://table-1.csv#{Document ID}";
15     rr:class foaf:Document
16   ];
17   rr:predicateObjectMap [
18     rr:predicate rdfs:label;
19     rr:objectMap [
20       rml:reference "Document ID"
21     ]
22   ];
23   rr:predicateObjectMap [
24     rr:predicate :department;
25     rr:objectMap [
26       rml:reference "Dep."
27     ]
28   ] .
```

column name (Line 20 and 26). Maps can further state a term type to define the kind of RDF term that should be generated (IRI, literal or BNode). Similarly, an additionally given data type defines what type a literal will have. Listing 3 presents the RDF result after the mapping in Listing 2 is performed by an RML engine. As expected, three documents are generated with their corresponding label and department information.

Having a formalism allows the implementation of RML engines such as RML Mapper[12], CARML[13], RocketRML [158] and SDM-RDFizer [81]. Moreover, a mapping language provides room for extensions which extend its capabilities. For example, RML Fields [46] is a proposed solution for nested data, while RML-star [45] extends RML with capabilities to generate RDF-star [72]. Approaches in Chapter 2 will make use of RML and enhance its capabilities to also cope with messy spreadsheets.

12 https://github.com/RMLio/rmlmapper-java
13 https://github.com/carml/carml

Listing 3: The RDF output after execution of the RML mapping in Listing 2.

```
<file://table-1.csv#%2AAB-ztad.63%2F23> a foaf:Document;
  rdfs:label "*AB-ztad.63/23" ;
  <file://table-1.rml.ttl#department> "GA" .

<file://table-1.csv#AB-hzyx-78%2F24> a foaf:Document;
  rdfs:label "AB-hzyx-78/24" ;
  <file://table-1.rml.ttl#department> "GA/BZ" .

<file://table-1.csv#AB%205-pbga.67> a foaf:Document;
  rdfs:label "AB 5-pbga.67" ;
  <file://table-1.rml.ttl#department> "BZ" .
```

1.3.3 *Semantic Desktop*

Far back in 1945, Bush already anticipated with his Memex system how desktops and their applications basically work in the future [32]. With increasingly smaller hardware and improving technology, desktop computers became reality in today's offices. In the early 2000s, intensive research began on the transformation from the desktop to the (Networked) *Semantic* Desktop [43] in parallel to the evolution of the Web to the Semantic Web. In contrast to usual desktop systems, a Semantic Desktop manages a user's digital information as Semantic Web resources using RDF [129]. Several building blocks are required to fully realize this technology. A key concept is that personal mental models of its users are respected. Here, Personal Knowledge Graphs (PKGs) are utilized since they mainly contain entities users are personally related to [12]. In case of Personal Information Management (PIM) such graphs are Personal Information Models (PIMOs) which try to represent a user's mental model with personal concepts and common ontologies [130]. They are also able to explicitly consider contextual information among resources. User observation can capture user activity which collects more information about individuals and relations among resources [156]. Once all relevant documents are semantically recorded, the capability to search is an important feature in the Semantic Desktop: besides browsing, full-text searches allow users to find concepts on their computer by keywords. To present results, a Semantic Desktop usually comes with a Graphical User Interface (GUI) reminding of Web browsers which let users navigate to resources, follow their relations or annotate them. Data sources in the personal information sphere of users are made available in the Semantic Desktop with adapters by transforming information elements to RDF (as shown in Section 1.3.2). All in all, the Semantic Desktop combines various approaches utilizing semantics in a single desktop system to support users in their knowledge work.

The concept of a Corporate Memory[14] (CoMem) further extends the Semantic Desktop vision to a bottom-up and group-wide ecosystem. Here, KGs are applied on a company-wide scale and thus are called Enterprise Knowledge Graphs (EKGs) [64]. Since knowledge workers from various departments share concepts and collaborate in groups, such graphs form a so-called Organizational Memory. To allow that, group management, visibility settings and concept sharing among users are additional features of such a system. Novel knowledge services [49] become available in various office applications for employees by the introduction of a semantic middleware (cloud server) and additional desktop integrations with plug-ins [104].

Usually, KGs do not exist and thus need to be bootstrapped first. How this can be conducted in a dedicated project is covered in the next section.

1.4 SCENARIO

To further motivate the research in this PhD thesis, a scenario about a Knowledge Graph (KG) construction project is outlined in the following and depicted in Figure 2. As already stated, intranet storages in

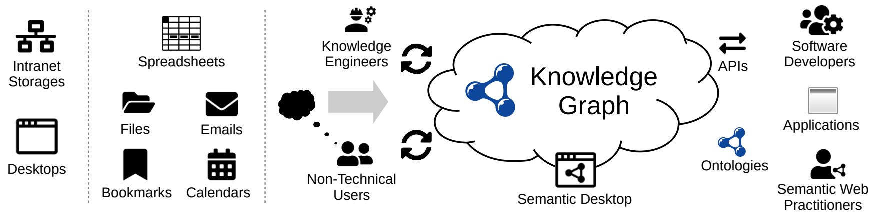


Figure 2: Motivational scenario about a project which builds a KG from messy enterprise data.

companies and desktop environments of employees are places where (messy) data accumulates. In this regard, spreadsheets are considered as frequently used office documents. Yet, more data can be found on desktop systems related to PIM, such as files, emails, bookmarks and calendars. As project leaders, Knowledge Engineers (KEs) have set the goal to build a KG from this potentially messy data pool (illustrated by a gray arrow). To do so, domain experts are consulted, since they have the necessary knowledge and mindsets to interpret the data and to identify personal concepts (indicated by a thought bubble). This user group is called “non-technical” because they usually do not have a computer science education. Both, KEs and non-technical users contribute to the construction and maintenance of a KG depicted as rotating arrows. However, there are more user groups to consider in such a project. Software developers also participate in this scenario by implementing applications on top of the evolving KG. To integrate

14 https://comem.ai/

them, appropriate Application Programming Interfaces (APIs) are provided to let them read and write the graph's content. Practitioners in the Semantic Web field also take part in the project. As ontology engineers they are able to model ontologies and also implement Linked Data (LD) applications. In the end, the KG's content and all related apps become available for a Semantic Desktop system which provides novel support in knowledge work, for example in the form of personal knowledge assistants.

In this PhD thesis various approaches are proposed to enable the building of KGs from such data and to include these user groups in the process.

1.5 OVERVIEW

In Chapter 1 a motivation to the topic was presented (Section 1.1) and resulting research questions were listed (Section 1.2). To give the reader more background knowledge, the topics Semantic Web, RML and Semantic Desktop were described in Section 1.3. A motivational scenario were outlined in Section 1.4. The remainder of the thesis is structured as follows.

Chapter 2 covers KG **construction approaches** and introduces in Section 2.1 challenges with messy data which is followed by related work (Section 2.2). After that, three main contributions are presented: an interactive approach in Section 2.3, the utilization of RML mappings in Section 2.4, and a procedure to predict such mapping rules in Section 2.5. Section 2.6 concludes this chapter and answers the first research question.

Chapter 3 focuses on various ways to **integrate users** in the construction process. First, user groups are discussed in Section 3.1 and related work is listed in Section 3.2. Each succeeding section covers dedicated approaches for another user group: non-technical users (Section 3.3), software developers (Section 3.4), and Semantic Web practitioners (Section 3.5). A conclusion of this chapter is given in Section 3.6 with an answer to the second research question.

Chapter 4 discusses **dataset generation methods** by first emphasizing on the open dataset problem for PIM data (Section 4.1). After related work (Section 4.2), a generator for person mentions is presented in Section 4.3. The next sections extend this generation procedure with a pattern language for spreadsheets (Section 4.4) and a suitable generator which reproduces them (Section 4.5). Finally, a crawler is presented in Section 4.6 which collects PIM datasets from users. The chapter is concluded in Section 4.7 with the third research question answered.

Chapter 5 presents **contributions related to this PhD thesis** in diverse domains. Besides many inputs for research on personal knowledge assistants (Section 5.1), it is also shown that KGs are beneficial in agricultural domain (Section 5.2) and in data science (Section 5.3). Section 5.4 closes this chapter with a summary.

Chapter 6 concludes the thesis with a summary in Section 6.1 and lessons learned in Section 6.2. An outlook on future work is given in Section 6.3. Closing remarks are stated in Section 6.4.

2 KNOWLEDGE GRAPH CONSTRUCTION

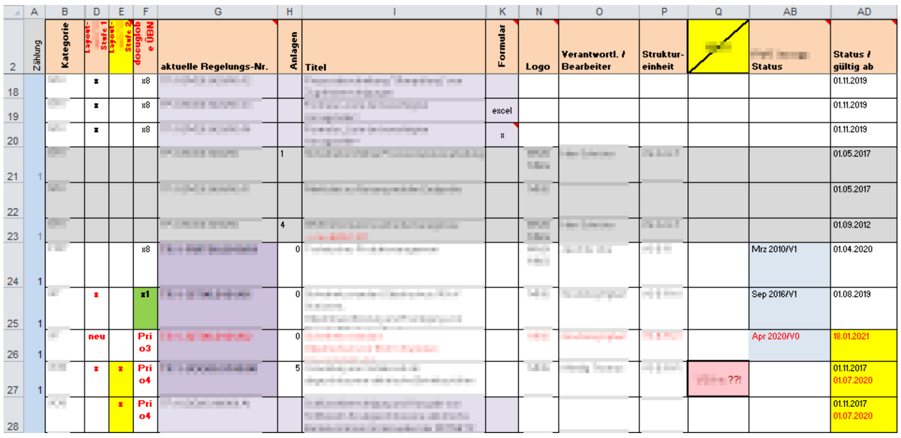
This chapter covers approaches for constructing knowledge graphs, especially from messy data. Parts of it have already been published in [145, 146, 151].

2.1 CHALLENGES WITH MESSY DATA

Usually, data lifting approaches [61] are used when Knowledge Graphs (KGs) are built, as shown in Section 1.3.2. These methods typically assume well (semi-)structured data with well-known schemata. From this starting situation, the main challenge lies in the correct definition of the mapping to receive a desired KG. However, this effort gets more complicated when data is not, as expected, in the right shape.

Enterprise data produced by employees such as labels or notes can easily pose unexpected challenges. Often these short texts do not contain regular grammar, provide only few statistical signals and are rather ambiguous [80]. For instance, such (sometimes noisy) text snippets can be found in file names where users follow various naming strategies [40, 77] and use differently ordered and concatenated keywords in form of technical terms, made-up words and even puns [33]. Similarly, this behavior can be observed in spreadsheets where people tend to insert data in cells in a "sloppy" way [9]. An experiment about people creating spreadsheets confirms that sheets often contain miscellaneous errors [29].

Especially spreadsheets are frequently used by knowledge workers in the industrial sector since they provide an easy and fast possibility to enter data in a well-understood way. Since spreadsheets do not predetermine how they should be filled, data is entered freely which causes the discussed unstructured content. Figure 3 depicts a censored spreadsheet obtained from an industry project which illustrates how sheets can appear in practice. To make contained challenges more concrete, Table 2 demonstrates a spreadsheet which combines several aspects of messy data. A KG constructed from it would contain three document instances (Line 1, 2 and 4) and one attachment instance (Line 3). Regarding the other cells, it would instantiate two departments (GA and BZ), three editors with their first and last name, three types, three change resources with separate version numbers and date information, two published dates in a standardized format and one sent statement with a Boolean value. The struck out editor would be modeled with a dedicated property to distinguish this person from the others. To build



	A	B	D	E	F	G	H	I	K	N	O	P	Q	AB	AD
2	Zählung	Kategorie	Layout-Stufe 1	Layout-Stufe 2	docuglobe ÜBN	aktuelle Regelungs-Nr.	Anlagen	Titel	Formular	Logo	Verantwortl. / Bearbeiter	Struktur-einheit	[redacted]	[redacted] Status	Status / gültig ab
18		[redacted]	x		x8	[redacted]		[redacted]							01.11.2019
19		[redacted]	x		x8	[redacted]		[redacted]	excel						01.11.2019
20		[redacted]	x		x8	[redacted]		[redacted]	x						01.11.2019
21	1	[redacted]				[redacted]	1	[redacted]		[redacted]	[redacted]	[redacted]			01.05.2017
22		[redacted]				[redacted]		[redacted]		[redacted]					01.05.2017
23	1	[redacted]				[redacted]	4	[redacted]		[redacted]	[redacted]	[redacted]			01.09.2012
24	1	[redacted]			x8	[redacted]	0	[redacted]		[redacted]	[redacted]	[redacted]		Mrz 2010/V1	01.04.2020
25	1	[redacted]	x		x1	[redacted]	0	[redacted]		[redacted]	[redacted]	[redacted]		Sep 2016/V1	01.08.2019
26	1	[redacted]	neu		Pri o3	[redacted]	0	[redacted]		[redacted]	[redacted]	[redacted]		Apr 2020/V0	18.01.2021
27	1	[redacted]	x	x	Pri o4	[redacted]	5	[redacted]		[redacted]	[redacted]	[redacted]	[redacted] ??!		01.11.2017 01.07.2020
28		[redacted]		x	Pri o4	[redacted]		[redacted]							01.11.2017 01.07.2020

Figure 3: A spreadsheet from an industry project. Some parts are censored due to confidentiality.

such a KG several challenges have to be tackled which are discussed in the following.

Table 2: Typical challenges with messy data occurring in an exemplary spreadsheet (already published in [151]).

Line	Document ID	Dep.	Editor	Type	Changes	Published	Sent
1	*AB-ztad.63/23	GA	~~Cooper~~ Smith	C	V1: 2015-03-02	42415	x
2	AB-hzyx-78/24	GA/BZ	Emma Thomas	N		TODO	
3	AB-hzyx-78/24 A1	GA/BZ	Smith, Leo				
4	AB 5-pbga.67	BZ	(new) Smith Thomas, E.	ed.c	V1: Dec2009 V2: Mar2010	15.05.2010	-

Multiple Surface Forms. In the *Editor* column it looks like more than three people are mentioned. This comes from the fact that various surface forms are used (e.g. "Emma Thomas" and "Thomas, E."). Such occurrences have to be recognized and reconciled by an appropriate method, for example, with named entity normalization [85]. Still algorithms may not consider all cases which is why human intervention becomes necessary to correct possible errors. Section 4.3 addresses this particular challenge in more detail.

Mixed Date Representations. Looking at the *Changes* and *Published* columns, one recognizes several date formats, such as `YYYY-MM-DD`, `MonthYYYY`, `DD.MM.YYYY` and days since 1970 epoch. The latter is the native way spreadsheets store dates, while the others are various textual representations. Whatever form is chosen, robust methods should detect and unify them to one representation, preferably to a common one like XSD's date type.

Acronyms and Symbols. The *Dep.*, *Type* and *Sent* columns do not contain usual words, they rather refer to entities or values with acronyms and symbols. While department names are reduced to two-letter acronyms, types are abbreviated from Changed, New, and editorial change. The sent status is expressed with an 'x' symbol to indicate the truth value *true*. Users tend to use such shortcuts to reduce typing efforts, which unfortunately makes the recognition of entities and values much more difficult.

Free Comments. The *Document ID* column contains a inconsistent asterisk symbol (*), while '(new)' is written in front of a persons last name in the *Editor* column. Since spreadsheets allow to edit cells freely, users are able to enter additional comments. However, this practice can easily become a distraction in named entity recognition and information extraction procedures.

Style Usage. Regarding the *Editor* column, typographical emphasis is used to stuck out Cooper. In fact, spreadsheets provide numerous ways to style cells and their texts with borders, fonts and colors. By changing styles, users can additionally express certain peculiarities, like for example, that an editor is not responsible anymore. If only plain text is analyzed, such nuances would not be recognized. Figure 3 illustrates more style usages in practice.

Multiple Entities in a Cell. While cells in the *Editor* column contain more than one person, multiple change entries are listed in the *Changes* column as well. When tables are not normalized, such data redundancies can occur and as a result cells may have multiple entities. In these cases, cell contents need to be split appropriately to retrieve the correct number of entities.

Multiple Types in a Table. It looks like the spreadsheet lists only documents, however in fact Line 3 is an attachment. This can be recognized by the capital letter 'A' and a number at the end of its document ID. Since entities of different types share some properties, they can be present in one table using identical columns. Such circumstances need to be considered once extracted entities are assigned to their classes.

Implicit Relationship. The attachment in Line 3 belongs to the document in Line 2, because they have matching prefixes in their document IDs. Since such relationships cannot be stated well in spreadsheets among cells or rows, users tend to write textual clues to express this fact. Thus, such implicit indications need to be discovered to make relationships explicit.

This list of challenges is by far not complete, but it demonstrates what peculiarities has to be considered in case of messy data. The next section examines how related work builds KGs and tackles some of the challenges. After that, approaches developed in this PhD thesis are presented.

2.2 RELATED WORK

Various research areas in literature also investigate how raw data can be turned into semantic representations. For tabular data it often involves (semantic) table interpretation or table understanding (for a survey on this topic see [25]). This typically includes semantic labeling (or annotation) of a table's schema and content with resources from ontologies and knowledge graphs [116]. More flexibility is provided by data lifting approaches where a formalism (or language) let users define how data is mapped (a survey is provided by [61]).

Since many tables exist on the Web with a wealth of information, several works focus on their interpretation. By linking tables in Web pages to Knowledge Graphs (KGs) with schema matching and schema mapping techniques, their contents become available to the Semantic Web. The following KGs are commonly exploited for this task: DBpedia [10], Wikidata [165], Wikitology [160], YAGO [159] and Probase [187]. For instance, Wang et al. [166] extract tables on the Web, detect their headers and tries to discover entities in it using Probase. A similar methodology is followed by Mulwad et al. [107]: instead of Probase, they exploit Wikitology to assign classes to columns, link cell contents to its resources and make column relationships explicit with properties. Quercini and Reynaud [118] additionally consider the use of a Web search engine in case of unknown entities. A trained text classifier assigns classes to entities by using their text snippets from the search result. MantisTable [38] contributes to the semantic annotation of tabular data with an interactive GUI to perform data preparation, for example.

All these methods have in common that they utilize structured knowledge to discover named entities or to recommend classes (types) and properties using already existing ontologies. They often assume well (semi-)structured HTML tables with well-known schemata. However, in the discussed scenario (Section 1.4), data can neither be assumed well structured, nor that schemata are already conceived, nor that suitable ontologies exist in advance. Moreover, enterprise data such as private datasets in companies typically consists of personal data and very specific information. Thus, it is to be expected that public datasets about common knowledge will not cover them properly. Due to confidentiality the query of private concepts in public Web search engines provided by foreign companies are usually not recommendable. For all these reasons, a more manual methodology is necessary to intervene in the construction process.

Data lifting techniques provide more flexibility by allowing to map input data to semantic representations in order to form KGs. These mappings can be performed automatically with algorithms, as shown by Any23[1] (Anything To Triples). Its algorithms are able to convert structured data to RDF. For example, in case of CSV files, each row is turned into a resource, while columns serve as their properties. Similarly, the Python tool excel2rdf[2] makes some assumptions about the input spreadsheets to map them properly. Similar approaches are so-called direct mappings which can, for example, map RDB to RDF

1 https://any23.apache.org/
2 https://pypi.org/project/excel2rdf

without any manual effort [175]. To exert influence in the process, procedures usually consider a Domain Specific Language (DSL) to let users express mappings. Such a mapping language has already been demonstrated in Section 1.3.2 with RML [50, 51]. Besides that, there are also similar methods available, like the LinkED methodology [123]: by combining D2RQ [23] for accessing RDB as virtual RDF graphs and SILK [164] to discover links between entities, it enables the publication of linked enterprise data. D2RML [35] adopts the RML formalism, but introduces RESTful web services and SPARQL endpoints as input sources and provides additional mapping features.

Since spreadsheets are observed to be a prevalent data management technology in companies, several tools and languages where developed to map their contents. These tools come with different languages and features. RDF123 [71] transforms spreadsheet rows to RDF graphs and additionally provides conditions, string manipulations and arithmetic calculations. The tool spread2rdf[3] defines its mappings with a Ruby-internal language. Sheet2RDF [60] utilizes the projection of annotations rule language (PEARL) to express the transformation. M^2 (Mapping Master) [111] builds upon OWL's Manchester syntax for its mapping language. Tarql[4] utilizes SPARQL construct queries to map spreadsheets. XLWrap [99] refers to spreadsheet contents with a special expression language and use template graphs for the construction. Mindswap's Excel2RDF[5] and its successor Convert2RDF[6] provide an interactive GUI but with very limited mapping functionalities. Besides a separate transformation language, some approaches exploit the fact that spreadsheets can be annotated or have a common structure. TabLinker[7], for example, let users style their sheets in advance to annotate table structures. Similarly, ExcelRDF [70] use native spreadsheet comments to add mapping declarations. In case of Bernardo et al. [16], common schema patterns in the biological domain are automatically recognized and matched with OWL ontologies.

All such approaches have in common that they insufficiently consider messy data, because they assume rather clean data or a separate pre-processing step in advance. In such cases, csvtk[8] (CSV Toolkit) can be used to perform data cleansing tasks on CSV files before they are mapped. A more sophisticated approach is provided by OpenRefine[9] which has capabilities to explore, transform and clean messy data. KGs can be formed with its RDF extension[10], but with limited map-

3 https://github.com/marcelotto/spread2rdf/
4 https://tarql.github.io/
5 https://web.archive.org/web/20080520234848/http:/www.mindswap.org/~rreck/excel2rdf.shtml
6 https://web.archive.org/web/20080327181331/http://www.mindswap.org/~mhgrove/convert/
7 https://github.com/Data2Semantics/TabLinker
8 https://github.com/shenwei356/csvtk
9 https://openrefine.org/
10 https://github.com/stkenny/grefine-rdf-extension/

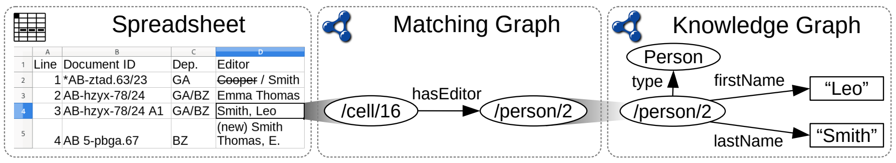


Figure 4: Annotation mechanism of AnnoSpreadKGC: a spreadsheet (left) is annotated via a matching graph (middle) with extracted resources stored in a KG (right). The figure has been published in [151].

ping capabilities compared to RML. The additional step to perform such pre-processing results in a heterogeneous mapping process, since messy data is not handled *during* the mapping.

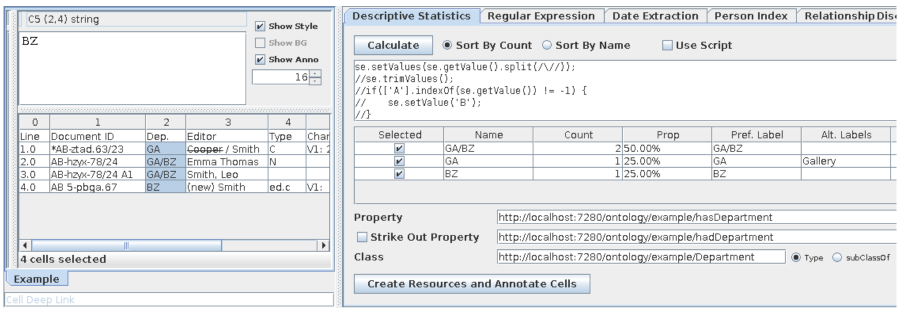
2.3 AN INTERACTIVE APPROACH: ANNOSPREADKGC

A first solution which handles the challenges discussed in Section 2.1 is proposed with an interactive approach. This includes a GUI for KEs to let them incrementally annotate spreadsheet data to construct a Knowledge Graph (KG). Due to this specific procedure it is named AnnoSpreadKGC (an abbreviation for Annotated Spreadsheet for Knowledge Graph Construction). To illustrate the annotation mechanism, Figure 4 gives a small example. On the left side, a spreadsheet is shown which can be explored by the user, in this case the sheet from Table 2. A unique URI is assigned to each cell to be able to refer to them in a matching graph (middle). Once a method extracts information from a cell (e.g. a person), corresponding statements are stored in a knowledge graph (right). The matching graph associates spreadsheet cells and KG resources which enables the enrichment of sheets with knowledge-related annotations.

The annotation process can be separated in three main steps. First, in a staging phase, a KE manually selects cells in a spreadsheet and invokes an extraction method on them (see below). The outcome can be reviewed by the user and adjustments can be made. Second, once the KE is satisfied with the result, the commit phase persists all acquired statements in the matching graph and knowledge graph. These two steps are repeated, until all spreadsheet cells are annotated, as depicted in Figure 4. In the third and final step, spreadsheet rows are interpreted as KG resources to form with their annotated cells a interconnected graph.

To tackle the challenges mentioned in Section 2.1, AnnoSpreadKGC provides configurable extraction methods. Figure 5 shows screenshots of their GUIs to demonstrate how they are used. In the following, each module is discussed in more detail.

Descriptive Statistics (Figure 5a). Summary statistics help to get an overview over categorical data, for example, by counting how often certain categories occur. This procedure can also be used to discover



C5 (2,4) string

BZ

0	1	2	3	4	
Line	Document ID	Dep.	Editor	Type	Char
1.0	*AB-ztad.63/23	GA	~~Cooper~~ / Smith	C	V1: 2
2.0	AB-hzyx-78/24	GA/BZ	Emma Thomas	N	
3.0	AB-hzyx-78/24 A1	GA/BZ	Smith, Leo		
4.0	AB 5-pbga.67	BZ	(new) Smith	ed.c	V1:

4 cells selected

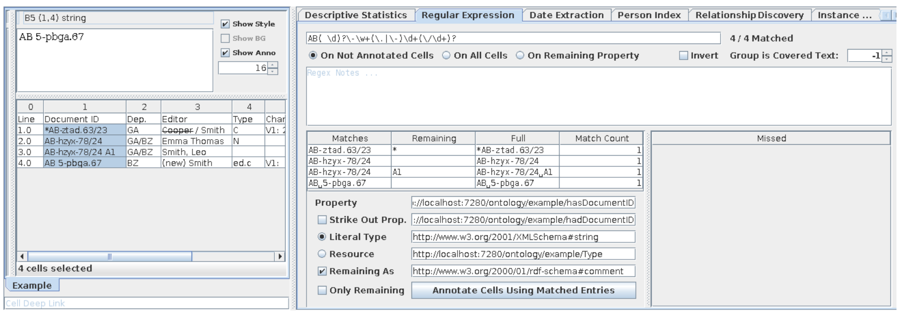
[✓ Sort By Count] Sort By Name; Use Script

```
se.setValues(se.getValue().split(/\//));
//se.trimValues();
//if(['A'].indexOf(se.getValue()) != -1) {
//    se.setValue('B');
//}
```

Selected	Name	Count	Prop	Pref. Label	Alt. Labels
✓	GA/BZ	2	50.00%	GA/BZ	
✓	GA	1	25.00%	GA	Gallery
✓	BZ	1	25.00%	BZ	

Property: http://localhost:7280/ontology/example/hasDepartment
Strike Out Property: http://localhost:7280/ontology/example/hadDepartment
Class: http://localhost:7280/ontology/example/Department [✓ Type] subClassOf

(a) Descriptive Statistics: a summary statistic lists extracted department names.



B5 (1,4) string

AB 5-pbga.67

0	1	2	3	4	
Line	Document ID	Dep.	Editor	Type	Char
1.0	*AB-ztad.63/23	GA	~~Cooper~~ / Smith	C	V1: 2
2.0	AB-hzyx-78/24	GA/BZ	Emma Thomas	N	
3.0	AB-hzyx-78/24 A1	GA/BZ	Smith, Leo		
4.0	AB 5-pbga.67	BZ	(new) Smith	ed.c	V1:

4 cells selected

```
AB( \d)?\-\w+(\.|\-)\d+(\/\d+)?
```

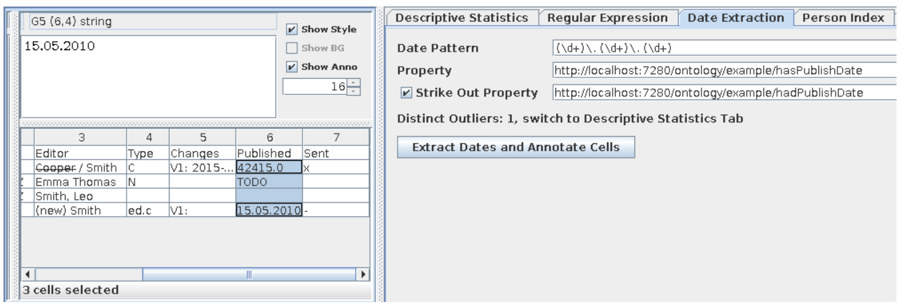
4 / 4 Matched

[✓ On Not Annotated Cells] On All Cells; On Remaining Property; Invert; Group is Covered Text: -1

Matches	Remaining	Full	Match Count
AB-ztad.63/23	*	*AB-ztad.63/23	1
AB-hzyx-78/24		AB-hzyx-78/24	1
AB-hzyx-78/24	A1	AB-hzyx-78/24␣A1	1
AB␣5-pbga.67		AB␣5-pbga.67	1

Property: ://localhost:7280/ontology/example/hasDocumentID
Strike Out Prop.: ://localhost:7280/ontology/example/hadDocumentID
[✓ Literal Type]: http://www.w3.org/2001/XMLSchema#string
Resource: http://localhost:7280/ontology/example/Type
[✓ Remaining As]: http://www.w3.org/2000/01/rdf-schema#comment
Only Remaining

(b) Regular Expression: document IDs are correctly extracted using a pattern.



G5 (6,4) string

15.05.2010

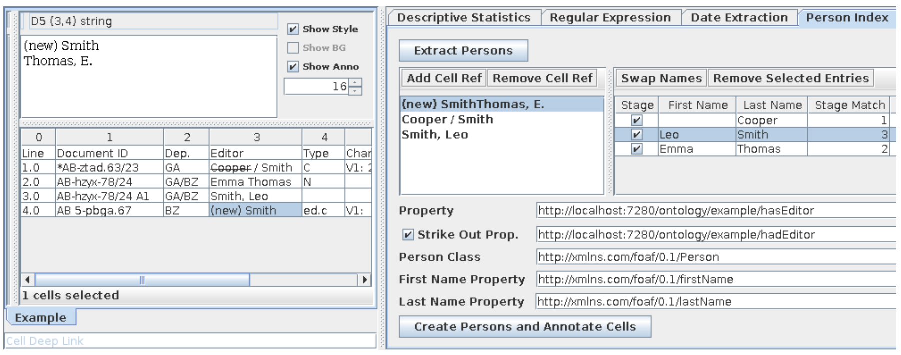
3	4	5	6	7
Editor	Type	Changes	Published	Sent
~~Cooper~~ / Smith	C	V1: 2015-...	42415.0	x
Emma Thomas	N		TODO	
Smith, Leo				
(new) Smith	ed.c	V1:	15.05.2010	-

3 cells selected

Date Pattern: (\d+)\.(\d+)\.(\d+)
Property: http://localhost:7280/ontology/example/hasPublishDate
[✓ Strike Out Property]: http://localhost:7280/ontology/example/hadPublishDate

Distinct Outliers: 1, switch to Descriptive Statistics Tab

(c) Date Extraction: published dates are discovered with a pattern.



D5 (3,4) string

(new) Smith
Thomas, E.

0	1	2	3	4	
Line	Document ID	Dep.	Editor	Type	Char
1.0	*AB-ztad.63/23	GA	~~Cooper~~ / Smith	C	V1: 2
2.0	AB-hzyx-78/24	GA/BZ	Emma Thomas	N	
3.0	AB-hzyx-78/24 A1	GA/BZ	Smith, Leo		
4.0	AB 5-pbga.67	BZ	(new) Smith	ed.c	V1:

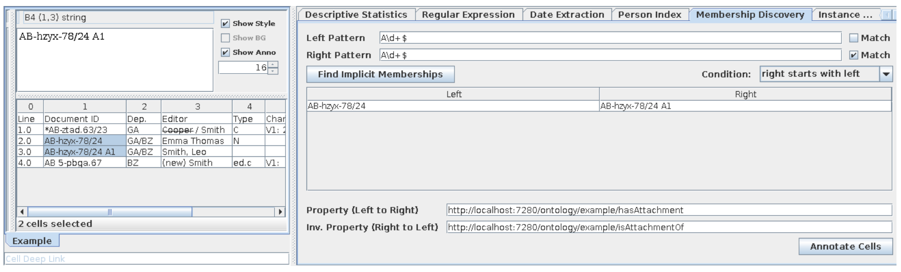
1 cells selected

(new) SmithThomas, E.
Cooper / Smith
Smith, Leo

Stage	First Name	Last Name	Stage Match
✓		Cooper	1
✓	Leo	Smith	3
✓	Emma	Thomas	2

Property: http://localhost:7280/ontology/example/hasEditor
[✓ Strike Out Prop.]: http://localhost:7280/ontology/example/hadEditor
Person Class: http://xmlns.com/foaf/0.1/Person
First Name Property: http://xmlns.com/foaf/0.1/firstName
Last Name Property: http://xmlns.com/foaf/0.1/lastName

(d) Person Index: distinct editors are listed with their first and last names.



B4 (1,3) string

AB-hzyx-78/24 A1

0	1	2	3	4	
Line	Document ID	Dep.	Editor	Type	Char
1.0	*AB-ztad.63/23	GA	~~Cooper~~ / Smith	C	V1: 2
2.0	AB-hzyx-78/24	GA/BZ	Emma Thomas	N	
3.0	AB-hzyx-78/24 A1	GA/BZ	Smith, Leo		
4.0	AB 5-pbga.67	BZ	(new) Smith	ed.c	V1:

2 cells selected

Left Pattern: A\d+$ — Match
Right Pattern: A\d+$ — [✓ Match]
Condition: right starts with left

Left	Right
AB-hzyx-78/24	AB-hzyx-78/24 A1

Property (Left to Right): http://localhost:7280/ontology/example/hasAttachment
Inv. Property (Right to Left): http://localhost:7280/ontology/example/isAttachmentOf

(e) Membership Discovery: document and attachment relationships are made explicit.

Figure 5: Screenshots of the extraction methods in the interactive approach (some have been published in [151]).

potential knowledge resources by recognizing multiple equal values. If cells are improperly formatted, a JavaScript-based script can transform (or split) their texts in a desired manner. Regarding the example, three departments are discovered for which typed KG resources are created. They are annotated on the cells with `hasDepartment`-relations.

Regular Expression (Figure 5b). Whenever cells contain information in form of structured text, regular expressions (regex) are suitable to extract them. The module let users define and apply regex patterns until desired matches are found. Covered texts can be turned into literal values or a match annotates a constant resource to its cells. Remaining unmatched text can be stored in a separate property. The example shows how only document IDs are extracted, while the inconsistent asterisk remark ($*$) is captured as a comment.

Date Extraction (Figure 5c). Spreadsheets can contain native dates (days since 1970 epoch) or a textual representation of them (strings). While the former is easily interpretable, the latter requires text extraction with a regex pattern. This can be seen in the example where published dates in different formats are recognized and annotated. Outliers like the "TODO" text are handled by the Descriptive Statistics module.

Person Index (Figure 5d). When multiple persons are mentioned in spreadsheet cells, a KG should distinctly catalog them by their names which can be seen as a person index[11]. However, name variations (selection, ordering, abbreviation) makes it difficult for an unsupervised extraction algorithm to discover persons properly. The module makes a first suggestion and let the KE correct possible errors (e.g. by swapping first and last name). Cell references can be inspected and changed, too. Figure 4 illustrates how a final result looks like.

Membership Discovery (Figure 5e). To discover implicit relationships, two regular expressions are used to divide cells into two groups. A relationship condition is tested in a pair-wise manner. This way documents are related to their attachments in the example by looking at equal prefixes.

Regarding **styled texts**, the following methods are able to recognize struck out occurrences: Regular Expression, Date Extraction and Person Index. Once such text parts are discovered, a separate "strike out property" is used to distinguish this fact in the KG. This way, for example, the struck out person Cooper is correctly referred to as a former editor.

In the third and final step, an **Instance Collector** turns spreadsheet rows into KG instances by connecting statements from annotated cells to them. This also includes the assignment of a default class if a type-statement is missing. Consequently, the spreadsheet's content is finally available as a KG with linked entities.

11 The generation of a person index for evaluation purpose is discussed in more detail in Section 4.3.

Case Study: Industrial Scenario

For a case study a power supply company provided five spreadsheets from an industrial scenario. The sheets are used to manage meta data of various types of documents: guidelines, board decisions, directives and hazardous substance (haz.sub.) directives. The last sheet lists document revisions and related process information. In Table 3 statistics about the spreadsheets' size and cell types are presented. Numbers reveal that string cells are most common followed by numeric cells which often represent dates. Formulae are ignored in further considerations, since they are not considered in KG construction.

Table 3: Statistics about the spreadsheet data used in a case study (has been published in [151]).

Sheet	Rows	Columns	Cells	String	Numeric	Formula
Guidelines	1,218	30	21,854	16,451	3,921	1,198
Board Decision	57	22	525	472	53	0
Directive	126	21	1,437	1,244	193	0
Haz.Sub. Directive	201	15	1,174	913	261	0
Revision	722	61	16,105	7,683	5,528	2,894
Sum	2,324	149	41,095	26,763	9,956	4,092

These spreadsheets were processed with AnnoSpreadKGC which results in the use of several extraction methods. Table 4 lists how many columns were considered and how often certain methods were applied. Since string cells often contained regular syntax (IDs and symbols), the regex method was frequently used. Especially in the revision sheet, date extractions were applied often, because dates are recorded to track finished tasks. Once entities were mentioned, the descriptive statistics method was performed. Transformation scripts were used whenever multiple entities were mentioned to split them. After the application of methods for 82 times and the final instance collection step, the resulting KG contains 25,016 triples with 2,719 instances assigned to 15 classes. It was loaded into CoMem to let domain experts browse, inspect and work with its content. Experts confirmed that the KG correctly modeled what was stated in the spreadsheets.

Limitations

The case study shows that an interactive approach like AnnoSpreadKGC involves considerable time and effort. Although, discussed challenges can be solved with the combination of the proposed extraction methods and an annotation mechanism, a lot of configurations and operations in the GUI are necessary. Since all these steps are performed

Table 4: Statistics about the used extraction methods on columns (processed/total) in a case study (has been published in [151]).

Sheet	Columns	Desc. Stat.	Regex	Date Extraction	Person Index
Regulation	24/30	5	16	2	1
Board Decision	14/22	1	11	1	1
Directive	14/21	3	5	3	3
Haz.Sub. Directive	12/15	2	5	2	3
Revision	18/61	3	1	12	2
Sum	82/149	14	38	20	10

manually by a KE without any tracking, similar executions cannot be redone in an automated manner. Such "imitations" would require a formalism that can specify and execute steps as if they would be manually performed in the GUI. Since mapping languages provide these advantages, a solution to utilize RML to cope with the challenges is discussed in the next section.

2.4 UTILIZING RML FOR SPREADSHEETS

The RDF Mapping Language (RML) supports the processing of CSV, XML and JSON files [50], however, its specification does not consider spreadsheets. Compared to the CSV format, they have a more complex data model involving workbooks, multiple sheets, and meta data rich cells. For example, a cell has several options for its appearance (colors, styles and borders) and content (plain texts, formatted texts, numbers with data types). State-of-the-art RML mappers do not implement a directly access to such meta data in a mapping.

To enable the mapping of spreadsheets with RML, it is necessary to extend the capabilities of RML engines. The Java-based RML Mapper[12] implementation is well-suited for this approach, since its code structure lets developers easily integrate new input sources. In order to parse and access spreadsheets, Apache POI[13] provides the necessary functionalities for Microsoft Excel[14] files. Other libraries could be considered as well to also support spreadsheets from OpenDocument[15] or Google[16]. In the following the implemented features are discussed in more detail.

In RML a logical source needs to be defined to map its data. To be compliant with the specification, a small spreadsheet ontology [136] (prefixed ss) is modeled which is used to refer to a workbook. Listing 4 demonstrates how users properly define logical sources: after

12 https://github.com/RMLio/rmlmapper-java
13 https://poi.apache.org
14 https://products.office.com/excel
15 https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office
16 https://www.google.com/sheets/about/

Listing 4: A logical source for an Excel spreadsheet (has been published in [145]).

```
 1 [
 2   a rml:LogicalSource ;
 3   rml:referenceFormulation ql:Spreadsheet ;
 4   rml:source [
 5     a ss:Workbook;
 6     ss:url "workbook.xlsx" ;
 7     ss:sheetName "Papers" ;
 8     ss:range "A2:A5" ;
 9     ss:javaScriptFilter "/Know\\w*/.test(valueString)" # optional
10   ]
11 ]
```

the right reference formulation (Line 3), the workbook's URL (Line 6), a sheet's name (Line 7) and a cell range (Line 8) is specified. These information tell an iterator what cells should be traversed in a triples map. Optionally, a JavaScript-based filter can be defined to only iterate cells that meet a certain condition. Definition 1 formally describes the semantics of the logical source for a spreadsheet.

Definition 1. Let W be a spreadsheet workbook and $S_{i,j} \in W$ a sheet in this workbook. $S_{i,j}$ is considered to be a two-dimensional cell matrix with column index $i \geqslant 0$ and row index $j \geqslant 0$. Start and end references of Excel's address range can be expressed equivalently with matrix indices (a, b) and (c, d). An iterator traverses the cells in a sheet $S_{i,j}$ for a given range (a, b) to (c, d) in column-first order with $a \leqslant i \leqslant c$ and $b \leqslant j \leqslant d$. The JavaScript-based filter is a predicate $\text{JS}(S_{i,j})$ which let the iterator only traverse $S_{i,j}$ if the predicate is true on an interpretation.

By having a logical source, RML mappings are able to access meta data of the iterated cells. This can be seen in Listing 5 where in Line 2 a cell's address is used in a template expression. To access nearby cells from the currently iterated cell, two ways are proposed: an relative and absolute reference. Both use a bracket notation to let the user state column and row indices. For relative reference, parenthesis $((\ldots))$ state a column shift (x-axis) and a row shift (y-axis) which let object maps reach any cells relative to the subject map's cell. Line 9 in Listing 5 demonstrates how a numeric value of a cell two column away on the right is accessed. Similarly, squared brackets $([\ldots])$ allow for an absolute reference as shown in Line 6. In contrast to the CSV format where column names are used, references with relative and absolute distances provide a higher mapping flexibility in tabular sheets. A formal definition about the reference is provided by Definition 2.

Listing 5: A subject and predicate-object map which access Excel data (has been published in [145]).

```
rr:subjectMap [
  rr:template "http://example.org/{address}"
] ;
rr:predicateObjectMap [
  rr:predicateMap  [
    rr:template "http://example.org/{[2,0].valueString}"
  ] ;
  rr:objectMap [
    rml:reference "(2,0).valueNumeric"
  ]
]
```

Definition 2. Let $S_{i,j}$ be a currently iterated cell. Given a distance (x, y), a relative reference would select the cell $S_{i+x,j+y}$, while an absolute reference would pick the cell $S_{x,y}$.

Once cells are referenced, their meta data can be accessed as already shown in Listing 5. All implemented attributes to acquire various meta data of a cell are listed in Table 5. Besides the cell's location, various access methods for its value and appearance are provided.

Case Study: Industrial Scenario

The spreadsheets from the previous section (see Table 3) serve again for a case study to test the implementation. To map their data, an RML mapping has been defined which consists of 25 logical sources, 26 triples maps, 126 predicate object maps and 51 function maps. Some FnO functions were frequently applied to cope with messy data: `parseDate` discovers dates and return them in a XSD date representation solving the *Mixed Data Representation* problem which is similar to the *Date Extraction* method in the interactive approach (Figure 5c). The `ifRegexReturnElse` function acts as a ternary operator known from programming languages. A match of a regular expression on a given cell value evaluates the operator's condition. This way, object maps are able to map string occurrences to KG resources which comes in handy for the *Acronyms and Symbols* challenge. Comparably, the `ifRegexReturnGroup` function is able to extract and return structured text using regex, similar to the *Regular Expression* method (Figure 5b). To tackle the *Multiple Types in a Table* issue, suitable JavaScript filters were defined in logical sources. However, not all desired statements could be generated with the mapping alone which is discussed in the following.

Table 5: List of attributes to access meta data of a spreadsheet cell.

Attribute Name	Description	Example
`address`	Location in a sheet	B3
`column`	Column index 0-based	1
`row`	Row index 0-based	2
`valueNumeric`	Floating point value	7.3
`valueInt`	Integer value	7
`valueBoolean`	Boolean value	true
`valueFormula`	Formula	`SUM(F5:F9)`
`valueError`	Possible error code, e.g. in case of division by zero	7
`valueString`	Plain text value	This is bold.
`valueRichText`	Formatted text in HTML syntax	This is <b>bold</b>.
`value`	String representation regardless of the data type	"7"
`json`	JSON representation [39] to retrieve several meta data at once	`{"address": "E2", "cellType": "string", "valueNumeric": 0.0, "valueString": "18.06.2009", ...}`
`backgroundColor`	Background color in hexadecimal RGB value	`#FFFFFF`
`foregroundColor`	Foreground color in hexadecimal RGB value	`#000000`
`fontColor`	Font color in hexadecimal RGB value	`#FF0000`
`fontName`	Font name	Arial
`fontSize`	Font size	12

Limitations

Once a set of cells needs to be analyzed coherently, a local cell-by-cell mapping procedure turns out to be inadequate. In case of multiple entities with several surface forms (e.g. person mentions), sophisticated Named Entity Recognition (NER) approaches need to be performed in a global setting. Similarly, the discovery of implicit relationships usually requires a global search and non-trivial matching or complex join conditions. Since such functionalities cannot be sufficiently covered with FnO functions alone, a separate post-processing became necessary in the case study. After the performed RML mapping, an additional Java program with around 1,500 lines of code had to be implemented with several conditions and operations to interconnect, correct and extend the KG. Again experts confirmed its correctness by working with the final version in CoMem.

The case study has shown that the definition of proper RML mappings tends to be time-consuming when messy data is involved. Especially large collections of unexamined files require users to inspect data, recognize discrepancies and model suitable mappings. To support them, a system could be able to predict mapping definitions in some cases automatically. In the next section the generation of initial mapping suggestions for spreadsheets is investigated.

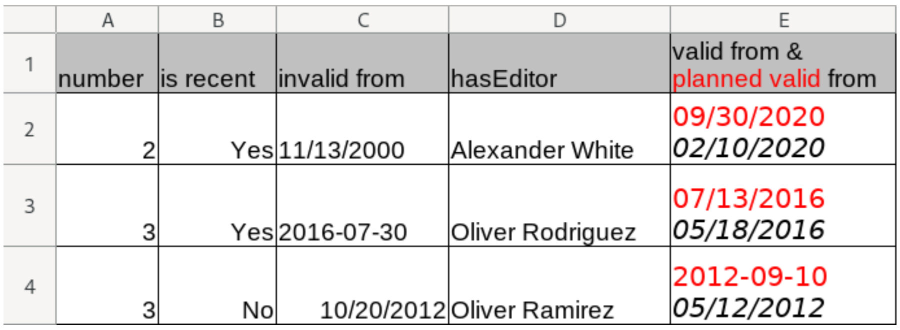
2.5 PREDICTING RML MAPPINGS: SPREAD2RML

For predicting RML mappings [50] on spreadsheets, some assumptions are made on the input data to make the problem more feasible. First, sheets with a 1-dimensional vertical layout and without nested headers should already be localized, segmented and functional as well as structural analyzed [25]. Second, subject maps are expected to be defined for each table. An example for such a spreadsheet can be seen in Figure 6. The first gray colored row is the header of the table, rows with white background (2–4) model entities and columns (A–E) represent their properties. With this initial situation, the prediction of RML mappings is defined as follows:

Definition 3. An RML mapping predictor suggests for each column in a spreadsheet table an RML predicate-object map with a reference, term type, datatype and possible FnO [105] function call.

To illustrate this, Figure 7 lists for each column in the spreadsheet (Figure 6) expected predicate-object maps which are discussed in the following.

COLUMN A (Figure 7a) The datatype `xsd:integer` and the reference `valueInt` are chosen, since the column solely contains integer numbers.



	A	B	C	D	E
1	number	is recent	invalid from	hasEditor	valid from & planned valid from
2	2	Yes	11/13/2000	Alexander White	09/30/2020 *02/10/2020*
3	3	Yes	2016-07-30	Oliver Rodriguez	07/13/2016 *05/18/2016*
4	3	No	10/20/2012	Oliver Ramirez	2012-09-10 *05/12/2012*

Figure 6: A spreadsheet to demonstrate the prediction of RML mappings (has been published in [146]).

COLUMN B (Figure 7b) Because cells have a certain data format, their numeric values are interpreted as Boolean values: a 1 displayed as "Yes" means *true* and a 0 displayed as "No" means *false*. Therefore, the datatype `xsd:boolean` and the reference `valueBoolean` are suggested.

COLUMN C (Figure 7c) The datatype `xsd:date` is suggested (Line 14), since this column contains dates, but in different formats and types (texts and native dates stored as numbers). To homogenize them, a function `parseDate` is used (Line 6) on the cells' JSON representation (Line 10) to acquire the prescribed date format `YYYY-MM-DD`.

COLUMN D (Figure 7d) Instead of simply mapping the editors' names to string literals, they should be linked to resources representing them. Therefore, the object map uses an IRI term type (not literal) and considers the execution of the function `entityLinking`.

COLUMN E (Figure 7e) Similar to Column C, dates are written in the cells, thus datatype is again `xsd:date`, but their texts are formatted. To only map the planned date in red, the function `getEntitiesByColor` with a suitable parameter expressing the color red in hexadecimal RGB value (Line 11) is suggested. Analogously, in a separate predicate-object map the italic styled date can be mapped as well.

To provide solutions for such cases, an RML mapping predictor as stated in Definition 3 is proposed. The approach, called Spread2RML, is able to fully automatically suggest RML object maps for spreadsheets assuming suitable triple maps and subject maps are defined in advance. It uses an extensible set of templates that predefine RML object maps. Spread2RML consists of fifteen templates which are listed in Table 6. Based on formally defined heuristics for each template, the algorithm selects the most promising one for a column (set of cells). The template is used to form a predicate-object map with the stated reference, term type, datatype and function.

```
<ex:A> rr:predicate gl:hasNumber ;                                 # Column A
  rr:objectMap [
    rml:reference  "(0,0).valueInt" ;
    rr:datatype    xsd:integer ;
    rr:termType    rr:Literal        ] .
```

(a) Mapping to integer literals using the `valueInt` attribute.

```
<ex:B> rr:predicate gl:isRecent ;                                  # Column B
  rr:objectMap [
    rml:reference  "(1,0).valueBoolean" ;
    rr:datatype    xsd:boolean ;
    rr:termType    rr:Literal       ] .
```

(b) Mapping to Boolean literals using the `valueBoolean` attribute.

```
<ex:C> rr:predicate gl:validFrom ;                                 # Column C
  rr:objectMap [
    fnml:functionValue [
      rr:predicateObjectMap [
        rr:predicate  fno:executes ;
        rr:object     <java:parseDate>
      ] ;
      rr:predicateObjectMap [ (...)
        rr:objectMap [
          rml:reference "(2,0).json"
        ]
      ]
    ] ;
    rr:datatype xsd:date ;
    rr:termType rr:Literal   ] .
```

(c) Mapping to XSD date literals using an FnO parsing function `parseDate` on the cell's `json` attribute.

```
<ex:D> rr:predicate gl:hasEditor ;                                 # Column D
  rr:objectMap [
    fnml:functionValue [
      rr:predicateObjectMap [
        rr:predicate  fno:executes ;
        rr:object   <java:entityLinking>
      ] ; (...)
    ] ;
    rr:termType rr:IRI   ] .
```

(d) Mapping to entities using term type IRI and the FnO function `entityLinking`.

```
<ex:E> rr:predicate gl:plannedValidFrom ;                          # Column E
  rr:objectMap [
    fnml:functionValue [
      rr:predicateObjectMap [
        rr:predicate  fno:executes ;
        rr:object   <java:getEntitiesByColor>
      ] ; (...)
      rr:predicateObjectMap [
        rr:predicate  (...) ;
        rr:objectMap [
          rr:constant "#ff0000"
        ]
      ] ; (...)
    ] ;
    rr:datatype xsd:date ;
    rr:termType rr:Literal   ] .
```

(e) Mapping to XSD date literals, but only red colored ones by using the FnO function `getEntitiesByColor`.

Figure 7: Expected RML predicate-object maps to map the spreadsheet in Figure 6 (has been published in [146]).

Table 6: Fifteen RML object map templates in the Spread2RML approach. Each template states the object map's reference, term type, datatype and function. A formally defined heuristic and a rank is used in the selection process. This table has been published in [146].

Object Map Template	Heuristic	Rank	Reference	Term Type	Datatype	FnO Function
Formatted Text	-	-	valueRichText	*child dependent*	*child dependent*	`getEntitiesByTag`
						`getEntitiesByColor`
						`getEntitiesByUnformatted`
Integer as String	$\lvert\{c \in S \mid int(c)\} \cup \{c \in N \mid dp(c) = 0\}\rvert \div \lvert C\rvert$	2	json	Literal	xsd:integer	`parseNumber`
Decimal as String[17]	$\lvert\{c \in S \mid dec(c, \text{"."})\} \cup \{c \in N \mid dp(c) > 0\}\rvert \div \lvert C\rvert$	1	json	Literal	xsd:decimal	`parseNumber`
Decimal as String[18]	$\lvert\{c \in S \mid dec(c, \text{","})\} \cup \{c \in N \mid dp(c) > 0\}\rvert \div \lvert C\rvert$	1	json	Literal	xsd:decimal	`parseNumber`
Date as String	$\lvert\{c \in S \mid date(c)\} \cup N\rvert \div \lvert C\rvert$	2	json	Literal	xsd:date	`parseDate`
DateTime as String	$\lvert\{c \in S \mid datetime(c)\} \cup N\rvert \div \lvert C\rvert$	3	json	Literal	xsd:dateTime	`parseDateTime`
Integer List as String	$\lvert\{\hat{s} \in \hat{S} \mid int(\hat{s})\}\rvert \div \lvert\hat{S}\rvert$	0	json	Literal	xsd:integer	`parseNumber`
Boolean as String	if $\lvert\{str(c) \mid c \in S\}\rvert \leqslant 2$ then …	4	json	Literal	xsd:boolean	`parseBoolean`
String	$1 - dup(C)$	0	value	Literal	xsd:string	-
Single Entity	$dup(C)$	3	valueString	IRI	-	`entityLinking`
Multiple Entities	$dup(\hat{S} \cup N \cup B)$	4	valueString	IRI	-	`entityLinking`
Native Boolean	$\lvert B\rvert \div \lvert C\rvert$	0	valueBoolean	Literal	xsd:boolean	-
Native Integer	$\lvert\{c \in N \mid dp(c) = 0\}\rvert \div \lvert C\rvert$	3	valueInt	Literal	xsd:integer	-
Native Decimal	$\lvert\{c \in N \mid dp(c) > 0\}\rvert \div \lvert C\rvert$	4	valueNumeric	Literal	xsd:decimal	-
Numeric with	$\forall \delta \in D$	5	json	Literal	xsd:date	`parseDate`
Data Format	$\lvert\{c \in N \mid df(c) = \delta\}\rvert \div \lvert C\rvert$				xsd:dateTime	`parseDateTime`

Since standard RML is often not enough to map messy data, functions are needed to parse, extract or link cell values. In the Spread2RML approach eight functions are utilized for these tasks which are described in the following.

- `parseNumber` With the help of regular expressions, this function is able to extract numbers with different decimal separators (comma vs. point) and returns them with or without decimal places.
- `parseBoolean` With a list of strings that indicate *true* and *false* values, this function uses string matching and majority voting to return a Boolean value.
- `parseDate(Time)` Both functions recognize various date formats in text and return them in XSD's date or dateTime format, while native (numeric) dates are converted according to Excel.
- `entityLinking` Using the Aho-Corasick string matching algorithm [5] and a given set of labeled entities, simple NER is performed on texts to return the found entities' IRIs.

To be able to extract specific parts of formatted text expressed in HTML syntax, dedicated functions are provided. After text extraction, they call as a subroutine the previously mentioned functions to return values in the correct format.

- `getEntitiesByTag` By passing an HTML tag, the following typographical emphasized text can be extracted: bold (`<b>`), italic (`<i>`), underlined (`<u>`) and struck out (`<strike>`) text.
- `getEntitiesByColor` With a given color in hexadecimal RGB value (e.g. `#ff0000`), only text in this color is extracted.
- `getEntitiesByUnformatted` Text which is neither typographical emphasized nor colored is considered by this function.

In the following, Spread2RML's heuristics in Table 6 are discussed in detail. *Formatted Text*, however, does not have any heuristic, since it is automatically applied once cells contain formatted text. According to the typographical emphasis and color, texts are grouped, including also a group for unformatted text. Virtual columns are created from these groups containing only plain text. This way multiple predicate-object maps with different properties are defined, each for a virtual column representing equally formatted text. Based on the format the right `getEntityBy*` function is selected, while typographical emphasis is given preference. Further analysis on virtual columns, now containing only plain text, decide which template is suggested for them.

Since the heuristics are defined on several concepts, they are covered first. C represents a column, while $c \in C$ are cells of this column. Cells

17 With "." decimal point
18 With "," decimal point

are separated among disjoint sets based on their value type: B contains Boolean valued cells, N numeric valued ones, and S cells with plain text (strings). The function $dp : N \to \mathbb{N}$ is able to return the number of decimal places for numeric valued cells. All possible data formats of cells are contained in the set D and the function $df : N \to D$ returns for a numeric cell its data format. Similarly, *str* returns a cell's string value. The function *sep* is able to separate a string by using non-alphabetical delimiters. This is used to define a set $\hat{S}$ which contains the substrings separated by *sep*: $\hat{S} := \{s \mid s \in sep(str(c)) \land c \in S\}$. To verify if a string valued cell can be parsed successfully, the predicate *int* checks for integer, *dec* for decimal number, *datetime* for date with time information and *date* without it.

For the *as String* templates in Table 6, parsing is involved to calculate the proportion of successfully parsed cells. In this estimation numeric valued cells N are considered, too. The heuristic in the *Integer List as String* template uses the substrings in $\hat{S}$ and checks how many are integer numbers. A more challenging task is the detection of Boolean values in the *Boolean as String* template. Since they have only two states (*true* and *false*), the heuristic considers at most two distinct string values, for example "yes" and "no". An average string length of below 3.5 distinguishes Boolean values from entity mentions, since users tend to write logical values with rather few characters.

More challenging is the distinction between string labels (like descriptions or names) and entity mentions (like organizations or persons). With the assumption that users tend to refer to such named entities recurrently and frequently, the heuristics in the *String* and *Single Entity* templates calculate a "degree of duplication". Its function is mathematically defined as follows:

$$dup(C) := \frac{2|C| - |U| - |P| + 1}{2|C|}, \text{ with}$$

$$U := \{c \in S \mid freq(str(c)) = 1\} \cup N \cup B$$

$$P := \{str(c) \mid c \in S\} \cup N \cup B$$

$$dup(X) := 0 \quad \forall X\ |X| \leqslant 1$$

It consists of two parts: element multiplicity expressed with U and element uniformity expressed with P. While U only contains unique string values (they only occur once), P holds distinct values. The highest degree of duplication is archived when all elements occur at least two times ($U = \varnothing$) and P is a one-element set ($|P| = 1$). Conversely, if all values differ from each other and no multiplicity exists ($U = C$), the diversity is also at a maximum ($P = C$). The *dup* function uses both sides to calculate a value between $\frac{1}{2|C|}$ and 1. A threshold of 0.5 decides if string are assumed ($\leqslant 0.5$) or single entities are expected (> 0.5). Similarly, for the *Multiple Entities* template the degree of duplication is computed on substrings $\hat{S}$. To be able to perform entity linking, the templates build temporary RDF graphs containing presumed entities as resources with their labels.

Besides text, spreadsheets provide native ways to store data with numbers. *Native Boolean*'s heuristic looks at the proportion of cells with Boolean values B. While *Native Integer* checks the proportion of cells with no decimal places, *Native Decimal* does the opposite. The *Numeric with Data Format* template counts how often cells have a certain data format by using D which currently contains various date formats.

Evaluation

Spread2RML is evaluated with three datasets. The first one is synthetically generated by the Data Sprout generator which uses generation patterns to produce especially messy data. Its generation mechanism is covered in detail in Section 4.5. A second dataset is acquired from the U.S. Government's open data platform called data.gov[19]. With its API, $8,689$ URLs to spreadsheets are collected, while $3,692$ of them could be downloaded and parsed successfully. To select appropriate spreadsheets for evaluation, two criteria are defined: their table structures should have a 1-dimensional vertical layout without nested headers and messy data as demonstrated in Figure 6 should be existent. An indicator for the latter criterion is a noticeable number of styled texts in the sheets. Twelve spreadsheets fulfilling these criteria are picked and manually annotated by defining RML mappings for them. For the third dataset the spreadsheets introduced in Section 2.3 Table 3 are used. Again, expected RML mappings are manually defined to annotate them.

For a baseline, the tool Any23[20] (Anything To Triples) is utilized. Because spreadsheets are not supported by the tool, they are converted to CSV files with `ssconvert`[21] from Gnumeric as well as `xlsx2csv`[22]. Unfortunately, due to this necessary transformation potential style information get lost.

For an evaluation metric, the actual RDF graph of the approach's result is compared with the expected RDF graph from the ground truth. However, a simple statement equality check between them is not enough: URIs of equal resources will most likely differ in both graphs, since they were created independently. As a solution, the problem is reduced to statement matching per cell by using available provenance information. Since the compared "cell graphs" only contain outgoing edges and thus are planar, a matching with subgraph isomorphism can be solved in linear time [56]. Having an actual graph A and an expected graph E, an injective function $m : R_A \rightarrow R_E$ is required

19 https://www.data.gov/
20 https://any23.apache.org/
21 https://help.gnome.org/users/gnumeric/stable/gnumeric.html#sect-files-ssconvert
22 https://github.com/dilshod/xlsx2csv (version 0.7.8)

Algorithm 1 Greedy Matching Procedure (published in [146])

```
Require: A and B are lists of statements, $|A| \geqslant |B|$, $n := |A|$, depth $:= |B|$
Ensure: a function $m : R_A \to R_B$ that matches resources from A to resources
    from B such that $m(A) \cap B$ has the largest possible number of elements.
 1: procedure ENUMERATE($A$, $B$, $n$, depth, $m_0 : R_A \to R_B$)
 2:     if depth $= 0$ then return
 3:     for $i \in [0, n-1]$ do
 4:         SWAP($A$, $i$, $n-1$)
 5:         for $j \in [n-1, |A|-1]$ do
 6:             $d_j :=$ DISTANCE($A_j$, $B_{|B|-|A|+j}$, $m_k$)
 7:         if $\sum d_j$ is under a certain threshold then
 8:             ENUMERATE($A$, $B$, $n-1$, depth $-1$, $m_{k+1}$)
 9:         SWAP($A$, $i$, $n-1$)
10:     return $\forall k$ largest $m_k$ with largest $\exists r\ m_k(r) = r$
11: procedure DISTANCE($(s_a, p_a, o_a)$, $(s_b, p_b, o_b)$, $m : R_A \to R_B$)
12:     $d > 0$ is a distance
13:     $m^{-1} := R_B \to R_A$, the inverse function of $m$
14:     if $s_a \neq s_b$ then
15:         if $m(s_a) \neq s_b$ or $m^{-1}(s_b) \neq s_a$ then return $d$
16:     $m(s_a) := s_b$
17:     if $p_a \neq p_b$ then
18:         if $m(p_a) \neq p_b$ or $m^{-1}(p_b) \neq p_a$ then return $d$
19:     $m(p_a) := p_b$
20:     if $o_a$ and $o_b$ are literals then
21:         if $o_a \neq o_b$ then return $d$
22:     else if $o_a$ and $o_b$ are resources then
23:         if $m(o_a) \neq o_b$ or $m^{-1}(o_b) \neq o_a$ then return $d$
24:     $m(o_a) := o_b$
25:     return 0
```

which match resources such that $m(A) \cap E$ has a maximum number of elements. Here, $m(A)$ denotes a substitution of resource URIs in all statements of A according to the matching m. Algorithm 1 is able to obtain such a matching with a greedy-based approach. The procedure ENUMERATE acquires all possible matches m_k through permutations. During the permutation, fixed statements are compared with a DISTANCE procedure. Should the distance be above a given threshold, the permutation is not continued. The DISTANCE procedure checks if all pairs of subjects, predicates and objects match consistently according to m and its inverse. If this is the case, the match is recorded in m so that at the end ENUMERATE is able to pick the largest m_k.

With the help of cell matchings, evaluation metrics can be defined as follows. When comparing an actual sheet S_A with an expected sheet S_E, equal column indices (c) and row indices (r) locate their overlapping cells: $c_A := S_A(r, c)$ and $c_E := S_E(r, c)$. With the best match m acquired by Algorithm 1 the cells' statements are compared, resulting in true positive *tp* (correct), false negative *fn* (missed) and false positive *fp* (false alarm) statements:

$$tp := c_E \cap m(c_A) \qquad fn := c_E \setminus m(c_A) \qquad fp := m(c_A) \setminus c_E$$

From this, precision (*p*), recall (*r*) and f-measure are defined:

$$p := \frac{tp}{tp + fp} \qquad r := \frac{tp}{tp + fn} \qquad f := 2 \cdot \frac{p \cdot r}{p + r}$$

Using the metrics, Table 7 presents their calculated values for Any23 and Spread2RML. Results show that Spread2RML outperforms the baseline for all datasets. Several generation patterns in Data Sprout are used, including no patterns at all (Clean) as well as all of them (for details see Section 4.4). Since four individual datasets are involved in this generation, average and standard deviations are calculated. On average, Spread2RML archives f-measure values between 0.27 and 0.47 with a higher variance than Any23. More realistic results are obtained from the data.gov datasets, where on average a 0.83 f-score is reached. The worst result of 0.30 f-measure is archived for the industry dataset. An explanation could be that for the publicly accessible government data a better data management strategy is pursued than for the private company datasets. Considering the given challenges, Spread2RML shows first promising results, although the scores are comparably low.

Limitations

The evaluation reveals some limitations of the approach which are discussed in the following.

Strings and Numbers. It is not uncommon that in a column numbers are mixed with texts, for example to indicate that a value is not available ("N/A"). Spread2RML considers in such cases the *String* template, although numbers occur more frequently. The reason is that some templates count and score differently. More expressive heuristics could make better distinctions and decide for a more suitable solution in such cases.

Entities vs. Strings. A limitation of the *dup* function is that it does not detect entities when they occur infrequently. Moreover, the detection of multiple entities can get very ambiguous. To better detect and discover entities, a preceding Named Entity Recognition (NER) step could be applied.

Boolean Misinterpretation. Rather short words for entities (like abbreviations) can easily be misinterpreted as Boolean values when only one or two of them occur in a column. Moreover, when a cell is empty this could also indicate a value, in many cases *false*. To overcome this limitation, more background knowledge could be provided for a better recognition. For example, a gazetteer list of possible expressions for Boolean values would be helpful.

Table 7: Evaluation metrics obtained after the application of Any23 and Spread2RML on the Data Sprout, data.gov and industry dataset (has been published in [146]).

	Any23			Spread2RML		
Data Sprout	Precision	Recall	F-Measure	Precision	Recall	F-Measure
Clean	0.39 ± 0.17	0.18 ± 0.10	0.24 ± 0.12	0.48 ± 0.33	0.34 ± 0.30	0.39 ± 0.31
Acronyms Or Symbols	0.38 ± 0.19	0.17 ± 0.11	0.23 ± 0.13	0.45 ± 0.27	0.27 ± 0.21	0.33 ± 0.22
Intra Cell Additional Information	0.24 ± 0.16	0.10 ± 0.08	0.13 ± 0.11	0.45 ± 0.32	0.35 ± 0.27	0.39 ± 0.29
Multiple Surface Forms	0.37 ± 0.19	0.16 ± 0.11	0.22 ± 0.13	0.47 ± 0.30	0.28 ± 0.23	0.34 ± 0.26
Multiple Types In A Table	0.40 ± 0.17	0.18 ± 0.10	0.24 ± 0.12	0.43 ± 0.32	0.29 ± 0.33	0.33 ± 0.34
Numeric Information As Text	0.39 ± 0.16	0.18 ± 0.10	0.24 ± 0.12	0.51 ± 0.31	0.33 ± 0.28	0.38 ± 0.29
Outdated Is Formatted	0.37 ± 0.19	0.17 ± 0.11	0.23 ± 0.13	0.50 ± 0.32	0.32 ± 0.29	0.38 ± 0.30
Partial Formatting Indicates Relations	0.23 ± 0.17	0.10 ± 0.09	0.14 ± 0.12	0.39 ± 0.24	0.38 ± 0.26	0.38 ± 0.25
Property Value As Color	0.40 ± 0.16	0.18 ± 0.09	0.24 ± 0.11	0.44 ± 0.26	0.30 ± 0.26	0.34 ± 0.24
All	0.27 ± 0.14	0.11 ± 0.09	0.15 ± 0.12	0.28 ± 0.19	0.27 ± 0.18	0.27 ± 0.18
	Any23			**Spread2RML**		
data.gov	Precision	Recall	F-Measure	Precision	Recall	F-Measure
0154ae39	0.03	0.01	0.02	0.81	0.89	0.85
12920281	0.40	0.21	0.27	1.00	1.00	1.00
17aecc3e	0.13	0.06	0.08	0.60	0.50	0.55
29a77cd1	0.01	0.00	0.00	1.00	1.00	1.00
5a0c9f12	0.19	0.18	0.18	0.65	0.87	0.74
614e3bd1	0.40	0.28	0.33	0.76	0.88	0.82
73b7a715	0.23	0.21	0.22	1.00	1.00	1.00
8cc689af	0.43	0.29	0.35	0.68	0.56	0.61
b193a34e	0.00	0.00	0.00	1.00	1.00	1.00
bc01528b	0.05	0.27	0.08	0.92	0.96	0.94
bf52a35a	0.40	0.28	0.33	0.93	0.95	0.94
c34bdd57	0.38	0.18	0.25	0.48	0.66	0.56
	Any23			**Spread2RML**		
Industry	Precision	Recall	F-Measure	Precision	Recall	F-Measure
	0.11	0.07	0.09	0.29	0.30	0.30

Mixed Text Styles. Since formatted text is separated into substrings per style, texts can get highly fragmented which complicates the recognition of entities. Other formats like font size and font family are currently not considered by Spread2RML. Such heavily styled texts show particular challenges and need more attention in future work.

2.6 CONCLUSION

In this chapter knowledge graph construction approaches were presented which especially consider messiness in data. To get a better understanding of messy data, several challenges were identified from a use case involving spreadsheets. Data lifting techniques in literature showed that they do not consider such messy data sufficiently. As a solution, an interactive approach called AnnoSpreadKGC was proposed which tackles the challenges with dedicated extraction methods and an annotation mechanism. Since this method turned out to be time-consuming and lacks in a formalism, RML was taken into consideration for the mapping of spreadsheets. For the first time, an RML engine was extended to enable the access to spreadsheet data in RML mappings. This contribution allows RML users to map spreadsheets to RDF using the RML mapping language. Because proper definitions of such mappings tend to be time-consuming, the RML mapping predictor Spread2RML was presented. Although its evaluation showed comparably low scores, it is able to suggest initial mappings in several cases of messy data.

Based on the conducted research, an answer is given to the first research question (Section 1.2): **How can knowledge graphs be built from especially messy data?**

Initially, it is crucial to become aware of the challenges in the building process which are caused by messy data. Dedicated extraction methods can be implemented to cope with these challenges, but the more exceptions to handle the higher their customization effort. Nevertheless, an interactive construction process gives Knowledge Engineers (KEs) the advantage to immediately adapt to unforeseen peculiarities in the data. In a case study it was demonstrated that AnnoSpreadKGC could successfully construct a KG from messy spreadsheets with extraction methods applied to 82 columns. Mapping techniques are also useful as long as input data is sufficiently accessible and mapping functions can be applied to extract and transform data. This was shown in another case study in which RML mappings were defined and FnO functions were implemented to map the same data from the first case study. However, not all statements could be mapped due to the approach's mapping nature, requiring the implementation of additional post-processing code. Yet, when messy data can be adequately handled *during* the mapping, an additional pre-processing or post-processing step becomes obsolete which is preferable. Once a mapping language can handle messiness by expressing its peculiarities, rules can be communicated, transferred and reused among participants facing similar challenges in their datasets. Systems become able to support KEs by performing automated analysis on data and provide their findings in form of mapping rules. This aspect was shown in Spread2RML where an evaluation with three datasets archived first promising results suggesting initial RML mappings.

While extraction methods can have a global view on the data, mapping techniques follow a local transformation approach. A combination of both worlds in an integrated methodology would provide many benefits for Knowledge Graph (KG) construction. This PhD thesis paved the way to this with two distinct contributions that consider in particular messy input data. A third achievement contains a complementary approach which is able to suggest mapping rules for such data.

3 HUMAN-IN-THE-LOOP INTEGRATION

This chapter covers approaches to integrate various users in the Knowledge Graph (KG) construction process.

3.1 MOTIVATION

The previous chapter focused on KG construction techniques operated by Knowledge Engineers (KEs), but without considering the source of knowledge required to correctly extract or map relevant data. If KEs autonomously model KGs from data unknown to them, many obscurities and ambiguities would lead to considerable misconceptions. Since written documents containing knowledge about a special domain and its unique datasets are usually very limited, KEs have to rely on expert knowledge from people working in this field. These employees also tend to define their own individual workflows and work-related concepts (e.g. a coding scheme) which are difficult to interpret by external people. Therefore, it is necessary to consult experts throughout the construction process.

In literature this method is recognized as a Human-in-the-Loop (HumL) approach, since human agents are involved in a process (i.e. a loop). It is defined as "a model that requires human interaction" according to the modeling and simulation glossary by the United States Department of Defense from 1998 [44]. The definition shows that human users are an integral part in an abstract model which could be an application or a data collection approach. In the research area of interactive machine learning the same concept applies when humans participate in the learning phase of an computational agent [79]. Here, human not only provide labels for data before or after learning, they are directly involved, for example by influencing measures in a step-by-step manner. Similarly, this PhD thesis considers a HumL case, once experts take part in the KG construction process with their knowledge or mindsets. Here, the mentioned "loop" stands for the typical KG lifecycle or process model: knowledge acquisition, curation (i.e. assessment, cleaning, enrichment), and deployment [58]. In this work, phases for integration of experts are roughly scheduled before (acquisition), during (curation) or after (deployment) KG construction.

Commonly, users show different technical skill levels in a project, ranging from domain experts with limited technical backgrounds to Semantic Web enthusiasts. For optimal integration strategies, HumL approaches need to be designed with their audience in mind. To offer a simple classification, members of a KG construction project

Table 8: A list of human-in-the-loop approaches designed for certain user groups and phases: before (acquisition), during (curation) and after (deployment) KG construction.

User Group	Approach	Before	During	After	Section
Non-Technical Users	Concept Mining	✓			3.3.1
	KECS	✓	✓	✓	3.3.2
	DeepLinker			✓	3.3.3
	RDF Spreadsheet Editor	✓	✓		3.3.4
Software Developers	SPARQL REST API			✓	3.4.1
	RDF2RDB-REST-API			✓	3.4.2
Semantic Web Practitioners	Simple RDFS Editor	✓			3.5.1
	LDAF	✓	✓	✓	3.5.2

are assigned to one of the following user groups: non-technical users, software developers and Semantic Web practitioners. This classification has already been indicated in the outlined scenario (Section 1.4 in Figure 2). The first group usually consists of employees who work in a day-to-day business with various data assets and tools. They are labeled "non-technical", since member of this group usually do not have a computer science education or completed related trainings. Software developers (or software engineers) on the contrary are well-versed with technical topics, but their domain knowledge is not as profound as in the previous group. They have a rather technical perspective of the domain for implementing applications and services. Semantic Web practitioners model ontologies and propose suitable knowledge services. They also ensure that Semantic Web standards are considered and respected throughout the project.

With these user groups in mind, this PhD thesis investigated several HumL approaches for KG construction. An overview of them with their intended user group is given in Table 8. Associated construction phases (i.e. before, during, after) are indicated with checkmarks, while corresponding sections are given. Non-technical users are usually early integrated in the acquisition phase to learn from their domain knowledge. During construction and after a KG is deployed they have the chance to review and feedback its content. Similarly, software developers get access to KGs for implementation tasks once they are available. Semantic Web practitioners support with ontology modeling in knowledge acquisition tasks or accompany the whole project with Linked Data (LD) applications. Knowledge Engineers (KEs) usually manage the KG construction project and apply approaches presented in this and the previous chapter.

The users' roles and applied HumL approaches are discussed in subsequent sections in detail. Before that, the next section presents work related.

3.2 RELATED WORK

Research about integrating Human-in-the-Loop (HumL) is subject of the multidisciplinary field Human-Computer Interaction (HCI) [52]. In the particular case of KG construction, this is the interaction between experts and knowledge-based systems where human intelligence can be exploited to acquire, curate and assess KGs in the construction lifecycle. The following related work is presented according to these major process steps.

In the era of expert systems, knowledge acquisition has been used to extract expertise from experts [113]. This includes data collection in form of interviews and questionnaires which can be performed systematically to acquire knowledge about a domain of interest [168]. In a similar fashion, possibly empty KGs and their ontologies can be bootstrapped with involved experts. Lamparter et al. [98] perform knowledge extraction on folder structures (i.e. classification schemes) which is very much in line with the given scenario (see Section 1.4). Yet, the authors propose an automatic approach without considering the inclusion of user feedback. On the contrary, research in the medical domain often considers to include medical experts in the knowledge modeling process. For example, DocKG [157] puts a "doctor-in-the-loop" to match synonyms, audit concepts and annotate them in electronic medical records. Feedback can also be exploited in the general mapping of concepts in ontologies. This is done by van Elst and Kiesel [55] where users are able to adjust, confirm or reject class relationships for an ontology mapping. Additional feedback results in more known relationships which helps in finding more potential mappings. Acosta et al. [3] enhance answer completeness for SPARQL queries by crowd-sourcing statements in order to form a KG. Their approach generates GUIs with background information to let user complete microtasks with triple patterns about common knowledge (e.g. movies). Human abilities are also very suitable for data pre-processing tasks before KGs are constructed. For instance, Oelen et al. [112] build a scholarly KG by integrating users for selecting suitable survey papers, extracting tables in PDFs and formatting their contents to ensure quality. Compared to manually entering or importing data, they show that a preceding HumL phase is more efficient.

Once KGs are initially built, their curation includes cleaning of false statements and enrichment with new ones. Since clean up tasks can quickly become tedious for users, research investigates the use of games with a purpose [4]. For example, by engaging users in solving crossword puzzles, Jovanovic [92] archives two KG maintenance tasks: the correction of typos in entity names and the reporting of wrong statements. Bu and Kuwabara [31] utilize a chatbot to ask a crowed if certain statements are true which is rewarded with bonus points on agreement. In a similar fashion, Hees et al. [73] apply a feedback-oriented rating of statements in their game to collect association strengths between concepts.

In case of KG enrichment, a common task is the population of existing ontologies with new instances. Clarkson et al. [37] let domain experts align user-defined instances (drugs) with a target ontology and maintain it by adding, splitting and merging them interactively. In the PROPhet approach [126] LOD is exploited to perform ontology population. Using a GUI, users can retrieve and filter instances as well as define suitable ontology mappings for classes and properties. Gentile et al. [66] use subject matter experts from the pharmaceutical domain to annotate medical package inserts. The textual annotations of entity types and relations are used to populate a KG and train annotators.

To assess a KG after its deployment, users need ways to consume its content, for example, with browsing, navigation or search capabilities. While SPARQL [181] is the obvious language to query RDF, many users are not familiar with its syntax and the related Semantic Web concepts. As a result, many tools have been proposed to visualize RDF as a graph for browsing its structure (for a survey see [7]). Still, developers require direct access to KGs to implement software on top of them, but without being hampered with extensive additional training. This is why methods are proposed to navigate KGs with technologies developers are familiar with. One approach is BASILar [42] which exposes SPARQL endpoints as Web APIs by generating REST resources. This idea is also pursued by Battle and Benson [14] who propose a bridge between REST operations and SPARQL queries. Instead of querying, a related concept is the conversion of whole RDF datasets to a form that is easily consumable for a target audience. R2D [122] archives this by transforming RDF graphs to RDBs which enables the reuse of existing visualization tools. RETRO [121] follow a similar approach, but does not physically transform semantic data to an RDB. Instead, RDF predicates are virtually mapped to simple tables to allow a SQL-to-SPARQL translation.

Related work shows that crowdsourcing is often considered to sufficiently obtain signals in knowledge acquisition and maintenance. In an enterprise, however, only a very limited number of employees are available to participate as experts in HumL approaches. Because of their restricted time available, tools should be integrated in their familiar work environment and have a rather flat learning curve to let them easily provide feedback. In literature such feedback is typically given with annotations in documents, the management of concepts or by simply answering questions. This is often complemented with already existing ontologies or KGs acquired from LOD. However, since personal and private enterprise data is subject of investigation, the utilization of LOD is only possible to a limited extent.

3.3 NON-TECHNICAL USERS

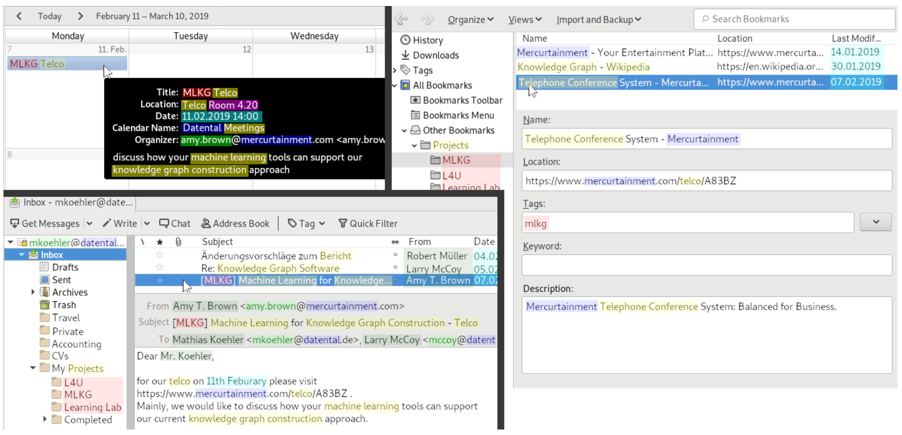
The group of non-technical users consists of knowledge workers who are, due to their daily profession, domain experts in their field. Employees typically have a solid understanding of their tasks, workflows, data stores, tools and work-related concepts. Computer science and especially Semantic Web related topics (like in Section 1.3.1) are understandably foreign to them which is why this user group is labeled "non-technical". Nevertheless, they are able to contribute in a KG construction project with their expertise by providing feedback during its building process. In return, the built KG and established knowledge services are then able to support them in their daily work. Because of their familiar working environment, office workers usually demand for GUIs to browse and manipulate data interactively. Therefore, proposed HumL approaches for this group naturally come with user interfaces to ease the input of feedback. Applied metaphors (i.e. allegories) help them to easily perceive modeled knowledge and comprehend how feedback can be given.

As indicated in the scenario (Section 1.4 in Figure 2), non-technical users work closely with enterprise data, typically on intranet storages or on their own desktops. Especially in such places, relevant personal concepts are hidden in data which are essential for a KG covering the domain. The following approach (Section 3.3.1) tries to identify together with domain experts what concept candidates in PIM data are relevant. Similarly, a method which considers such concepts in file names is covered in Section 3.3.2. To be able to annotate contents of desktop files with higher-level concepts as intended in the Semantic Desktop, Section 3.3.3 proposes a deep linking approach. Last but not least, users are enabled to freely enter their expertise in form of RDF independent of given data (Section 3.3.4).

Parts of this section have been published in [141, 142, 147, 149, 150].

3.3.1 *Concept Mining on Personal Data*

A promising source for Knowledge Graph (KG) construction are the employees' personal information spheres which can consist of tasks, mails, contacts, calendar entries, visited websites, bookmarks, files, and so on. Just representing these information elements identically in a KG would not consider higher level concepts mentioned in their texts such as projects, companies and topics. Conversely, if every term found in texts such as mail subjects, file names or text bodies is represented as a semantic entity, the KG will clutter with many irrelevant resources and relations not useful for the users. Because of their subjective views an automated and thus objective selection of relevant concepts cannot lead to a desired outcome [47]. Therefore,



Title: MLKG Telco
Location: Telco Room 4.20
Date: 11.02.2019 14:00
Calendar Name: Datental Meetings
Organizer: amy.brown@mercurtainment.com <amy.brow

discuss how your machine learning tools can support our knowledge graph construction approach

Name	Location	Last Modif...
Mercurtainment - Your Entertainment Plat...	https://www.mercurta...	14.01.2019
Knowledge Graph - Wikipedia	https://en.wikipedia.or...	30.01.2019
Telephone Conference System - Mercurta...	https://www.mercurta...	07.02.2019

Name: Telephone Conference System - Mercurtainment
Location: https://www.mercurtainment.com/telco/A83BZ
Tags: mlkg
Keyword:
Description: Mercurtainment Telephone Conference System: Balanced for Business.

Subject	From	Date
Änderungsvorschläge zum Bericht	Robert Müller	04.0
Re: Knowledge Graph Software	Larry McCoy	05.0
[MLKG] Machine Learning for Knowledge...	Amy T. Brown	07.0

From Amy T. Brown <amy.brown@mercurtainment.com>
Subject [MLKG] Machine Learning for Knowledge Graph Construction - Telco
To Mathias Koehler <mkoehler@datental.de>, Larry McCoy <mccoy@datent

Dear Mr. Koehler,

for our telco on 11th Feburary please visit
https://www.mercurtainment.com/telco/A83BZ .
Mainly, we would like to discuss how your machine learning tools can support
our current knowledge graph construction approach.

Figure 8: An exemplary personal information sphere which consists of a calendar, bookmarks and emails. Relevant concepts are color-highlighted in green (persons), blue (organizations), red (projects), purple (places), cyan (times), and yellow (general topics). This figure has already been published in [142].

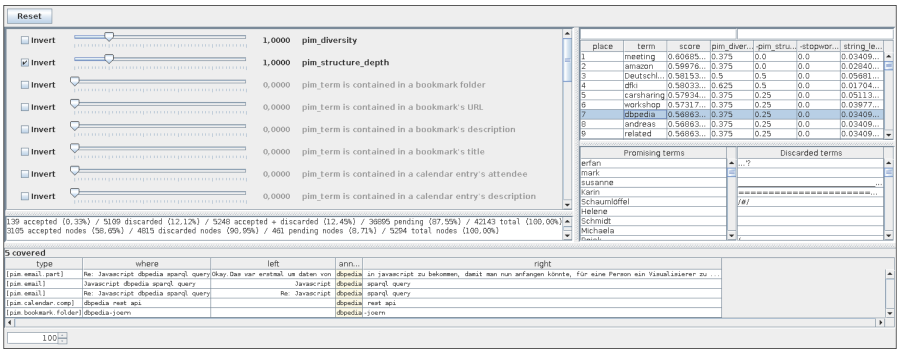
a HumL approach is proposed to let users decide what concepts are relevant/irrelevant to them.

An illustrative example of a user's personal information sphere is presented in Figure 8. Here, three typical sources of personal data are depicted: a calendar (top left), bookmarks (right) and emails (bottom left). They contain several relevant concepts which are highlighted in a specific color: persons (green), organizations (blue), projects (red), places (purple), times (cyan), and general topics (yellow). How such concepts are discovered and selected by users is explained in the following concept mining approach.

To discover promising concept candidates, the proposed method called Concept Miner exploits the structure of PIM data in desktop applications. Since certain text fields mention specific kinds of named entities using similar formats, it is easier to automatically extract them from these fields. For instance, there is a high probability that ...

- the location-field in calendar entries mention places,
- the from-field in emails refer to persons with first and last names, and
- hyperlinks and email addresses have names of organizations in their host parts.

Users' vocabulary (like special topics, technical terms or abbreviations) can be expected from texts they wrote themselves such as event titles, sent email bodies or subjects, and file names. One indicator for relevant terms could be their independent occurrence in different PIM applications. For example, in Figure 8 "MLKG" can be found in various places regardless of the application. Another indicator for relevancy is the spacial location in PIM hierarchies (or classification schemes) created by users. While higher located terms are usually more general



Reset

☐ Invert	1,0000	pim_diversity
✓ Invert	1,0000	pim_structure_depth
☐ Invert	0,0000	pim_term is contained in a bookmark folder
☐ Invert	0,0000	pim_term is contained in a bookmark's URL
☐ Invert	0,0000	pim_term is contained in a bookmark's description
☐ Invert	0,0000	pim_term is contained in a bookmark's title
☐ Invert	0,0000	pim_term is contained in a calendar entry's attendee
☐ Invert	0,0000	pim_term is contained in a calendar entry's description

```
139 accepted (0,33%) / 5109 discarded (12,12%) / 5248 accepted + discarded (12,45%) / 36895 pending (87,55%) / 42143 total (100,00%)
3105 accepted nodes (58,65%) / 4815 discarded nodes (90,95%) / 461 pending nodes (8,71%) / 5294 total nodes (100,00%)
```

place	term	score	pim_diver...	-pim_stru...	-stopwor...	string_le...
1	meeting	0.60685...	0.375	0.0	0.0	0.03409...
2	amazon	0.59976...	0.375	0.0	0.0	0.02840...
3	Deutschl...	0.58153...	0.5	0.5	0.0	0.05681...
4	dfki	0.58033...	0.625	0.5	0.0	0.01704...
5	carsharing	0.57934...	0.375	0.25	0.0	0.05113...
6	workshop	0.57317...	0.375	0.25	0.0	0.03977...
7	dbpedia	0.56863...	0.375	0.25	0.0	0.03409...
8	andreas	0.56863...	0.375	0.25	0.0	0.03409...
9	related	0.56863...	0.375	0.25	0.0	0.03409...

Promising terms	Discarded terms
erfan	...'?
mark	
susanne	...
Karin	========================...
Schaumlöffel	/#/
Helene	
Schmidt	
Michaela	

5 covered

type	where	left	ann...	right
[pim.email.part]	Re: Javascript dbpedia sparql query	Okay.Das war erstmal um daten von	dbpedia	in javascript zu bekommen, damit man nun anfangen könnte, für eine Person ein Visualisierer zu ...
[pim.email]	Javascript dbpedia sparql query	Javascript	dbpedia	sparql query
[pim.email]	Re: Javascript dbpedia sparql query	Re: Javascript	dbpedia	sparql query
[pim.calendar.comp]	dbpedia rest api		dbpedia	rest api
[pim.bookmark.folder]	dbpedia-joern		dbpedia	-joern

100

Figure 9: Graphical user interface of Concept Miner: slides on the left are used to rank and classify terms on the right (has been published in [142]).

(like "Projects"), terms occurring deeper in trees are more specific (e.g. "Learning Lab"). Such heuristics help to rank promising concept candidates.

To involve users in the selection process, a dedicated GUI is provided (Figure 9). Initially, a user's PIM datasets (mails, bookmarks and calendars) are locally crawled by the system (see Section 4.6 for details). In this process, found email addresses are used to discover persons, gazetteer lists detect cities and countries in location-fields of events and terms with only symbols or numbers are automatically discarded. Single-word terms are generated by tokenizing texts in bookmarks (folders, titles, descriptions), emails (folders, subjects, bodies), and calendars (summaries, descriptions, locations). These terms are listed on the right side of the GUI, however there are still a considerable amount of candidates a user needs to binary classify as either "promising" (pressing *enter*) or "discarded" (pressing *delete*). To support them, suitable ranking and filtering capabilities are provided with sliders to adjust metric weights (left side). These settings are necessary, since relevant terms can come in various shapes, for instance, infrequent short folder names, last names in mail addresses, project names in upper case acronyms, or frequent multi-word terms in emails. By moving given slides, users can freely configure combinations of metrics and their weights using a weighted harmonic mean score. Thus, users are able to rank terms from different perspectives to better pick them as relevant concepts. The GUI additionally shows for a selected term its occurrences in the datasets. In the example, the term "dbpedia" has a high PIM diversity, since it is found in four places: in an email body, its subject, an event's title and a bookmark folder.

Limitations

Since Concept Miner is still in an early stage of development, various limitations of this HumL approach can be identified. The high variety of settings by the list of sliders can easily overwhelm users. As a solution, proven presets could be provided to give good starting points in the search space. Although, feedback in form of a binary classification is given by the user, Concept Miner does not learn from it, for example train a machine learning model. Such predictions could support users in their decision making. Promising terms need further considerations to model them as meaningful KG resources. Usually, terms need to be properly disambiguated and reconciled to uniquely identified concepts (synonym sets). In this process, they are typically assigned to classes in an ontology population step [115]. Relations between terms or concepts are also not considered by the approach.

In the observed personal information sphere labels of files and folders from a file system are not included, although they may contain promising terms written by users. Thus, with a focus on file names, a comparable but more mature HumL approach is proposed in the next section which also addresses the discussed limitations.

3.3.2 *Personal Knowledge Graphs from File Names*

Employees manage their documents in day-to-day business often in a form of classification scheme. A very prominent storage is the hierarchical file system where files are named and arranged in folders. Since these elements can be freely named by users (except few technical limitations), they tend to give them labels using their own vocabulary. Thus, file names naturally mention task-related concepts such as persons, projects, organizations and topics, but also technical terms, made-up words and even puns [33]. Frequently, words are concatenated and differently ordered in short ungrammatical and maybe noisy labels due to various file naming strategies by users [40, 77]. Moreover, although files and folders are hierarchically related, this structure does not explicitly state how mentioned entities relate to each other. All of these peculiarities have to be considered when Personal Knowledge Graph (PKG) [12] are constructed from such a source.

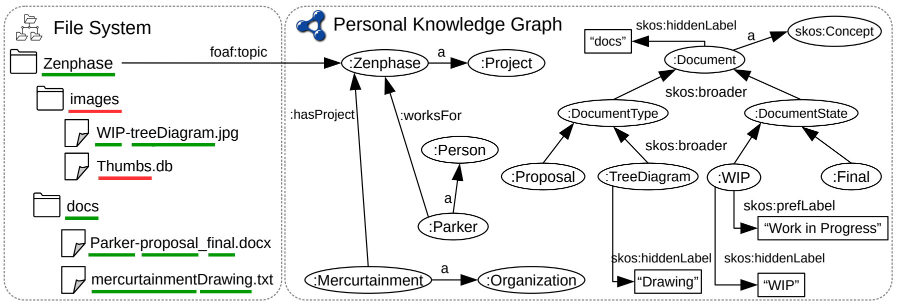


Figure 10: A file system (left) containing relevant (green) and irrelevant (red) terms is used to construct a knowledge graph (right) with taxonomic and non-taxonomic relations. Some edges are omitted due to readability. This figure has been published in [147].

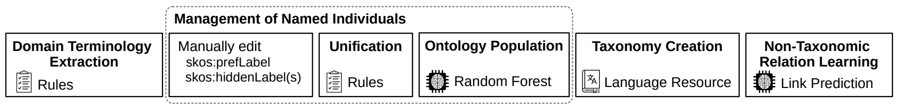

Management of Named Individuals
Domain Terminology Extraction
Rules
Manually edit skos:prefLabel skos:hiddenLabel(s)
Unification
Rules
Ontology Population
Random Forest
Taxonomy Creation
Language Resource
Non-Taxonomic Relation Learning
Link Prediction

Figure 11: From left to right the components of KECS with indicated AI support (has been published in [147]).

Figure 10 presents an example how file names (left) could be used to form a PKG (right). Not all aspects are visualized because of space reasons. Due to the subjective view of a user, some terms may be too general ("images") or technical ("Thumbs") which classifies them as irrelevant (underlined in red). For relevant terms (underlined in green) corresponding PKG resources (`owl:NamedIndividuals`[1] according to OWL [177]) are created and their provenance is tracked (`foaf:topic`-relation). Non-taxonomic relations (`:hasProject`, `:worksFor`) are stated between discovered and typed named individuals (`:Mercurtainment`, `:Parker`, `:Zenphase`). Similarly, taxonomic relations (`skos:broader`) form a taxonomy tree from more general topics (`skos:Concepts`) using the Simple Knowledge Organization System (SKOS) vocabulary [172]. Abbreviations such as "WIP" can be expanded using an individual's preferred label (`skos:prefLabel`), while synonyms and other spellings can be recorded as hidden labels (`skos:hiddenLabel`).

To construct such PKGs from file names the HumL approach Knowledge Extraction from Classification Schemes (KECS) is proposed. It supports Knowledge Engineers (KEs) and domain experts in performing the following construction tasks: domain terminology extraction, management of named individuals, taxonomy creation and non-taxonomic relation learning (Figure 11). Rules and machine learning models are applied during KG construction to predict new RDF statements from feedback which reduces manual efforts. In the following, the approach's special storage model, its components and Graphical User Interface (GUI) are discussed.

Knowledge Graph Model. Since machine learning is also involved in suggesting statements, two agents, namely the Knowledge Engineer (KE) and an Artificial Intelligence (AI) are able to give feedback in form of true but also false assertions. To record this, additional meta data per RDF statement is stored in the KG model:

- the agent who stated the assertion,
- the date and time when it was asserted,
- the assertion's rating which can be *true*, *false* or *undecided*, and
- the agent's confidence as a floating-point number ranging $[0, 1]$.

Suggestions by AI can always be outvoted by the KE, however, as long as the KE does not disagree, affirmative AI assertions are assumed to be true.

1 https://www.w3.org/TR/owl2-syntax/#Named_Individuals

Domain Terminology Extraction. To make suggestions for terms in file names, rules are applied in form of heuristics. Because of uncertain word boundaries, the files' basenames are tokenized by character type and camel case[2]. The tokens are also negatively rated by checking if they consist of a single letter, multiple symbols or digits (except for potential years) or stop words. An example for this procedure is given in Table 9 where a file name is split into tokens which are positively (✓) and negatively (✗) rated. As a result, AI assumes that *WIP*, *2007*, *tree* and *Diagram* are potentially relevant tokens. The KE is always able to merge several tokens to one multi-word term (e.g. *Tree Diagram*). Since accepted terms may occur somewhere else in the

Table 9: An example showing how file names in KECS are automatically tokenized and rated.

File Name	`WIP__for2007-treeDiagram!(28)A.jpg`												
Tokens	`WIP`	`__`	`for`	`2007`	`-`	`tree`	`Diagram`	`!`	`(`	`28`	`)`	`A`	`.jpg`
Rating	✓	✗	✗	✓	✗	✓	✓	✗	✗	✗	✗	✗	✗

classification scheme, regular expressions are used to search them automatically. In case of multi-word terms, potential word concatenations with minus '-', underscore '_', space ' ' and no separator are considered in the search. For instance, *treeDiagram* would have the variations *tree-Diagram*, *tree_Diagram* and *tree Diagram*. In the end, a named individual (`owl:NamedIndividual` [177]) is created in the KG from the set of collected terms T.

Management of Named Individuals. However, the creation of a new named individual is not necessary, if the terms T resemble an already existing individual. To detect this, the Jaccard similarity coefficient [82] between terms T and labels L of named individuals are calculated:

$$J(T, L) = \frac{|T \cap L|}{|T \cup L|}$$

Is the similarity measurement above a certain threshold, terms are added to the individual, otherwise a new one is created. In either case, a `foaf:topic`-relation keeps track which file mentions which named individual.

Unification. Still, it can happen that two or more named individuals are created which share the same meaning. To unify them, their URIs are correctly substituted and their statements are merged in the KG. In order to support the KE in finding potential unifications, AI provides some suggestions by checking label information: preferred labels are compared with the Levenshtein distance [101] and token

2 `https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/StringUtils.html#splitByCharacterTypeCamelCase-java.lang.String-`

equality, while hidden labels are checked for overlaps and prefix/postfix matches. As an example, the following label pairs would result in suggestions to unify their named individuals: ("Tree Diagram", "Diagram"); ("diagram", "diagramm"); ("Peter Parker", "Parker Peter").

Ontology Population. During the assignment of named individuals to classes, the KE is supported with a random forest model [27] which is able to predict classes for individuals without a type. The model uses class labels from already existing positive examples and a gazetteer-based embedding technique to acquire feature vectors from preferred labels. This method uses gazetteer lists to look up words in labels, while remaining characters are classified by their character class (e.g. digits, quotes and spaces). A training feature is the coverage proportion of characters or words in the labels. For example, "Diagram 3" receives the vector $(\text{English Noun} = 0.77, \text{Digit} = 0.11, \text{Space} = 0.11)$, while "Tree Diagram 27" has the vector $(\text{English Noun} = 0.73, \text{Digit} = 0.13, \text{Space} = 0.13)$. Since the former individual is assigned to the class `skos:Concept`, the random forest model will probably predict the same class for the latter due to its very similar feature vector.

Taxonomy Creation. The Simple Knowledge Organization System (SKOS) vocabulary is used to model a taxonomy tree of concepts (`skos:Concepts`) mentioned in file names. To provide suggestions for broader concepts, AI utilizes a language resource (lexical-semantic net) which contains synonym sets (synsets) and hypernym relations (generalizations). First, labels from concepts in the KG are used to find their associated synsets. By using hypernym paths, as soon as two or more concepts show overlapping ancestors with an average path distance below a certain threshold, they are suggested as broader concepts. For example, given the hypernym paths (a) *timetable* → *overview* → *depiction* and (b) *diagram* → *depiction*, the method would propose *depiction* as a broader concept for *timetable* and *diagram*.

Non-Taxonomic Relation Learning. Link prediction on the structure of the classification scheme (CS) is performed to predict non-taxonomic relations. The idea is that neighborhood in the CS can be an indicator to use same non-taxonomic predicates between resources. Due to the `foaf:topic`-relation, the location of instances (i.e. named individuals assigned to an ontology class) in the CS is known. To perform link prediction, this information needs to be transformed into an instance graph. Two instances are connected with an undirected edge if they are mentioned closely (one hop) in the CS. On this graph local similarity measures are calculated (for a survey see [128, Table 1]) to acquire feature vectors. Based on domain and range information a test set is formed which contains all possible instance-property-instance combinations. An Euclidean distance between a training vector and a test vector below a certain threshold indicates a promising non-taxonomic statement, since they show similar neighborhood in the CS.

Graphical User Interface. To enter feedback and build the PKG, a GUI in form of a Web application is provided for the KE (Figure 12). Green/red colored elements together with thumbs-up/thumbs-down buttons let users see and enter positive and negative feedback. At any time the KE can manually edit the PKG through the GUI.

Figure 12a presents its three-column layout with tabs for each component. A file explorer (left) lists for each file in a browsed folder (`/User/Downloads`) its file name, rated terms and attached named individuals (distinguished by a hashtag symbol). All named individuals together with their types are itemized in the middle. By providing a side-by-side view of these lists, Drag&Drop operations can be performed to state statements with a selected property from the drop-down list in the top middle. In a simple ontology view (right side) classes and properties can be defined, renamed and rated.

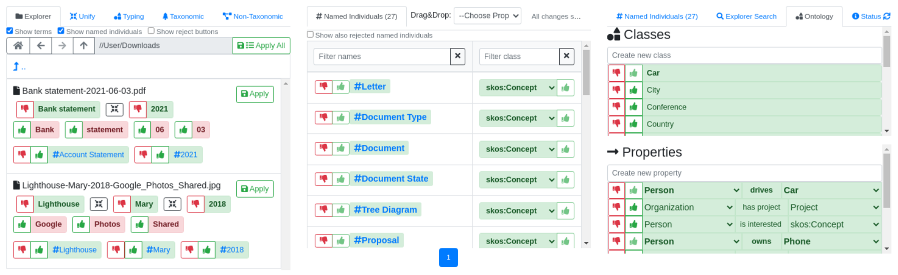
Figure 12b depicts a suggestion component, in this case for typing named individuals. AI proposes a list of suggestions which is accepted/rejected either individually or in bulk by the KE. Given feedback can be inspected below and can be undone in either way. Once a named individual is selected, a detail view (Figure 12c) allows to edit its preferred label, type, synonyms and file attachments. The current PKG construction progress is visualized in a status view (Figure 12d) in several sections: bars indicate the progress in tagging, typing, taxonomic and non-taxonomic relations which is aggregated to an overall assessment score. Such estimations give hints to the KE where further modeling is necessary.

Case Study: Expert Interviews

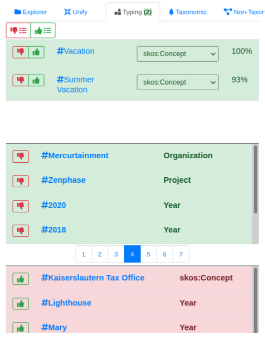
A case study was conducted to examine how KECS performs in real scenarios. Since non-technical users would require considerable training to use the approach on their own, interviews between a KE and an expert were conducted instead. During the interviews experts were asked to describe personal views on their files which were translated in appropriate PKG statements by the KE using the GUI.

Expert Interview Setup. Through industry projects with a large power supply company, it was possible to get in contact with four experts from four departments, namely guideline management, property management, license management and accounting. Table 10 lists meta data about the datasets and corresponding experts interviewed in the case study. While three people have managed files in file systems (FS), one has primarily worked with spreadsheet data (SS). To be able to also apply KECS on spreadsheets, the SS1 dataset was turned into a tree structure in the following way: tables were transformed into folders, their columns were interpreted as subfolders and rather short distinct cell values became file names of files.

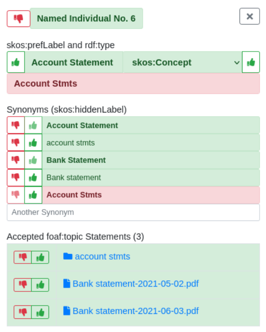
Interviews were conducted as follows: after the dataset was loaded into KECS, the KE invited the corresponding expert to a one-hour



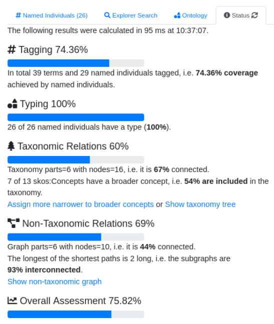
(a) The three-column layout shows a file explorer (left), list of named individuals (middle) and ontology editor (right).



(b) Typing component which suggests classes for named individuals.



(c) Detail view of a selected named individual.



The following results were calculated in 95 ms at 10:37:07.

Tagging 74.36%

In total 39 terms and 29 named individuals tagged, i.e. **74.36% coverage** achieved by named individuals.

Typing 100%

26 of 26 named individuals have a type (**100%**).

Taxonomic Relations 60%

Taxonomy parts=6 with nodes=16, i.e. it is **67%** connected.
7 of 13 skos:Concepts have a broader concept, i.e. **54% are included** in the taxonomy.
Assign more narrower to broader concepts or Show taxonomy tree

Non-Taxonomic Relations 69%

Graph parts=6 with nodes=10, i.e. it is **44%** connected.
The longest of the shortest paths is 2 long, i.e. the subgraphs are **93% interconnected**.
Show non-taxonomic graph

Overall Assessment 75.82%

(d) Status view showing the construction progress.

Figure 12: Screenshots from the graphical user interface of KECS (have been published in [147]).

virtual phone conference to share the screen and show the GUI. During the traversal through files, the KE asked questions about them, while answers from the expert immediately lead to modeling operations. Once AI proposed some suggestions, they were discussed with the expert and feedback was entered accordingly. The status view (Figure 12d) was consulted every ten minutes to change the focus based on the estimated progress. At the end of the interview (after about 50 minutes) the expert was asked to complete a questionnaire. Additionally, several data points are captured during the usage of KECS. The effort of the KE operating the GUI is quantified with the observation of keystrokes, mouse clicks and Drag&Drop operations. To record the evolution of the PKG, every ten inputs a snapshot of the construction metrics shown in the status view (Figure 12d) is recorded. In the following, the collected results are presented and discussed.

Expert Questionnaire. Table 11 shows the questionnaire consisting of seven questions (Q) and the experts' answers (E). Additionally, an average value and standard deviation (Avg. & SD) of their scores is calculated. Q1 was asked to find out how familiar the users are with their datasets. Q2 investigates if experts think that file names actually reflect their vocabulary. Q3 lets participants estimate the recall of the constructed PKG, while Q4 estimates its precision. Questions refer

Table 10: The datasets which were used in the interviews with experts. The table additionally list statistics about the number of folders and files, the tree's maximal and average depth as well as the average file/folder name length (has been published in [147]).

Dataset	Expert	Folders	Files	Max. Depth	Avg. Depth	Avg. Name Length
SS1	E1	103	198	3	2.98 ± 0.16	8.84 ± 9.86
FS1	E2	25,988	95,760	17	9.49 ± 1.93	23.30 ± 16.88
FS2	E3	8,939	64,571	17	9.18 ± 1.68	32.43 ± 16.77
FS3	E4	54,933	325,476	22	10.08 ± 2.22	24.24 ± 14.57

Table 11: A questionnaire together with the experts' answers and average values (has been published in [147]).

Question	E1	E2	E3	E4	Avg. & SD
Q1: *How many years have you been working with the data?*	13	7	4	0	6 ± 5.48
Q2: *How much do words in the file names reflect your language use (vocabulary) at work (scale: 1 – 10)?*	9	8	9	9	8.75 ± 0.50
Q3: *Estimate how much your language use (vocabulary) at work is represented by the established tags (percentage).*	50	15	10	10	21.25 ± 19.3
Q4: *The established tags meaningfully reflect the language use (vocabulary) at your work (scale: 1 – 7).*	7	6	4	6	5.75 ± 1.26
Q5: *The established tags are assigned to meaningful classes (scale: 1 – 7).*	6	7	6	7	6.50 ± 0.58
Q6: *The established tags are meaningfully structured in a taxonomy (scale: 1 – 7).*	7	6	5	4	5.50 ± 1.29
Q7: *The established tags meaningfully relate to each other (scale: 1 – 7).*	5	7	6	7	6.25 ± 0.96

to named individuals as "established tags", since the GUI presented them with hashtags. For the opinion-based questions a seven-point Likert scale is used ranging from 1 ("fully disagree") to 7 ("fully agree"). The PKG's meaningfulness is estimated with Q5 (populated ontology), Q6 (taxonomy) and Q7 (non-taxonomic relations).

The average value of 8.75 out of 10 for Q2 shows that the experts agree that their file names reflect their language use at work. Although, expert E4 stated that file system F3 has not been used in the person's daily work (Q1), it was still possible for the expert to recognize and explain found concepts. Received scores for Q4 to Q7 indicate that the PKGs were meaningfully modeled. These results show that file systems can be a promising source for PKG construction.

AI Performance. Table 12 (together with Table 13) summarizes the recorded quantitative data during the interviews. For each column a dataset-expert pair is stated, while each row lists a measurement.

Per construction phase the true and false assertions by KE and AI are counted, while the latter's accuracy is additionally given. AI's accuracy is the amount how often experts agreed with its suggestions. Unfortunately, due to an software error in taxonomy creation for the first two interviews, no boarder concepts could be predicted.

The accuracy values provide insight into AI's prediction performances. Regarding domain terminology extraction, the values decreased, because multi-word terms are not considered which had to be corrected by the KE frequently. In case of unification, more false positives were suggested due to the rules' tendency in high recall. For ontology population, mediocre performance can be explained with the simple gazetteer-based feature vectors on preferred labels. During taxonomy creation, the language resource often returned too general and thus unsuitable concepts resulting in low scores. The learning of non-taxonomic relations did not archive good results, since the limited number of examples did not lead to a useful prediction model. All in all, some helpful statement could be predicted by AI, however there is still much room for improvements.

Effort Estimation. In Table 13 the number of created named individuals (#Resources) and the KE's effort in the GUI are presented. At the bottom of the table, KE's numbers (see also Table 12) are aggregated to calculate an assertions per inputs ratio. The values show that 0.6 to 0.7 assertions were stated for one input operation by the KE. Thus, two inputs were already enough to express a true or false statement. A value below 1.0 can be explained with not negligible navigation and search operations in the GUI. Suggestions from AI and (bulk) feedback buttons seems to contribute to this outcome. Especially, Drag&Drop operations were perceived as a simple and fast method to make statements. In summary, with moderate effort meaningful PKGs could be modeled.

Limitations

The case study identified also some limitations. Ambiguities are not considered by domain terminology extraction. As a result, similar or equal terms are associated with the same named individual, although they could have different meanings. Conversely, equal terms mentioned in various places in the file system can mean different things. Thus, context needs to be considered in terminology extraction. Additionally, frequently occurring terms having multiple words could also be suggested by the module. Currently missing term metrics could help the KE to select promising ones first, comparable to the concept mining approach (Section 3.3.1). Regarding machine learning, the case study showed that there are potentials for improvements, especially for non-taxonomic relation learning. The prediction performance could be increased by better exploiting given feedback and the training of other models.

Table 12: For each construction task the number of true and false assertions by KE and AI together with AI's prediction performance as accuracy values. Parts of this table has been published in [147].

Measurement	SS1 (E1)	FS1 (E2)	FS2 (E3)	FS3 (E4)
	Domain Terminology Extraction			
KE True	82	50	33	26
KE False	48	44	14	72
AI True	400	270,168	242,149	948,405
AI False	286	220,285	106,573	617,366
AI Accuracy	$0.67 = 45/67$	$0.72 = 59/82$	$0.83 = 35/42$	$0.31 = 25/80$
	Management of Named Individuals			
KE True	102	68	39	58
KE False	30	24	15	25
AI True	462	32,161	8,223	37,159
AI False	4	1	23	155
AI Accuracy	N/A	N/A	N/A	N/A
	Unification			
KE True	10	2	2	0
KE False	6	18	12	4
AI True	8	10	7	2
AI False	0	0	0	2
AI Accuracy	$0.57 = 4/7$	$0.10 = 1/10$	$0.14 = 1/7$	$0.00 = 0/2$
	Ontology Population			
KE True	105	78	61	55
KE False	73	29	22	19
AI True	134	102	92	85
AI False	1	8	6	2
AI Accuracy	$0.23 = 18/78$	$0.65 = 30/46$	$0.66 = 23/35$	$0.48 = 12/25$
	Taxonomy Creation			
KE True	21	19	14	12
KE False	0	0	4	8
AI True	N/A	N/A	9	10
AI False	N/A	N/A	0	0
AI Accuracy	N/A	N/A	$0.56 = 5/9$	$0.20 = 2/10$
	Non-Taxonomic Relation Learning			
KE True	5	23	33	7
KE False	0	42	20	0
AI True	0	52	42	0
AI False	4	11	5	0
AI Accuracy	$0/0$	$0.19 = 10/52$	$0.52 = 22/42$	$0/0$

Table 13: Created resources in the KG, the KE's effort in the GUI and aggregated assertion per inputs ratio. Parts of this table has been published in [147].

Measurement	SS1 (E1)	FS1 (E2)	FS2 (E3)	FS3 (E4)
#Resources	88	50	39	32
KE Clicks	599	602	359	356
KE Enter-Key	60	56	30	47
KE Drag&Drop	26	34	21	18
All KE Assertions	482	397	269	286
All KE Inputs	685	692	410	421
KE Assertions/Inputs	0.70	0.57	0.66	0.68

In almost the same manner with file names, also contents of desktop resources can be associated with semantic concepts. This is enabled by the following approach covered in the next section.

3.3.3 *Deep Linking Desktop Resources*

Usually, employees work with various applications and resources in their own personal desktop environments. As intended in the Semantic Desktop [129], documents should be associated with concepts which semantically describing their contents. To bridge this gap between raw data and domain conceptualization, users should be enabled on their desktops to make connections explicit. The Resource Description Framework (RDF) is particularly suitable for this task, since its core concept is the linkage of resources using statements (i.e. making edges in an RDF graph) [182]. While RDF resources are naturally identified with URIs, desktop resources in form of documents, presentations and semi-structured data are located with absolute paths and the "file" URI scheme[3]. However, when complex and deeply nested files need to be associated with concepts, these surface links are not sufficient. In the Web the practice of deep linking [171] allows users to link to a specific piece in a website by using fragment identifiers[4]. Yet, such fragmentation links are almost completely unavailable for complex documents on the desktop [91], while related media fragments only consider audio, video and image formats [176]. Hence, users are unable to refer to a certain location deep inside their files, for example, to a shape in a slide of a presentation.

To still let users refer to certain locations within files, the approach DeepLinker is proposed. Its locally running server produces and

3 https://tools.ietf.org/html/rfc8089
4 https://datatracker.ietf.org/doc/html/rfc3986#section-3.5

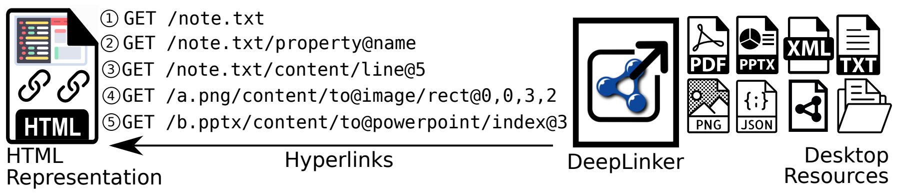
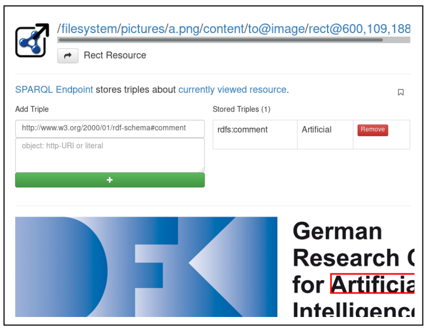

```
① GET /note.txt
② GET /note.txt/property@name
③ GET /note.txt/content/line@5
④ GET /a.png/content/to@image/rect@0,0,3,2
⑤ GET /b.pptx/content/to@powerpoint/index@3
```

Figure 13: Architecture of DeepLinker with exemplary deep links (has been published in [141]).

interprets deep links to desktop resources as demonstrated in Figure 13: common file types on the desktop (right) can be browsed with DeepLinker using HTTP GET requests (middle) which return HTML representations of the referred parts (left). By concatenating path segments, a user can traverse further into a file's structure, for example, by referring to (3) the 6th line of a plain text file or (5) the fourth slide of a PowerPoint presentation. More illustrative examples are provided by Figure 14 which shows four deep links with their rendered HTML websites. The deep link in Figure 14a highlights and comments the word "Artificial" in DFKI's logo, while in Figure 14b a link refers to the shape "Fehler vermeiden" (highlighted in red, in English "avoid errors") in the 4th PowerPoint slide. Highlighting of the third line in a text file is demonstrated in Figure 14c. The last screenshot (Figure 14d) shows how a deep link refers to the element "Participate" (bordered red) in a downloaded Web page.

To archive this, DeepLinker interprets each path segment of a given deep link segment-by-segment as a parameterized method: `/method@param1,...,paramN/`. Each method returns a DeepLinker resource which allows to chain them in order to traverse further into desktop resources. To do so, the following path segment methods are implemented. `child` and `index` let users select sub-resources by name or sequential number. In case of text content, `line` and `substring` highlight corresponding character string parts. Sub-images can be selected with `rect`, while `cssSelector` utilizes Cascading Style Sheets (CSS) selectors to pick an element in an XML structure. The `download` method retrieves an external file and `property` acquires for a given key its value. A special '`to`' method transforms resources into another format if supported (e.g. a PDF document to plain text).

Applying such a HumL approach promises several benefits. With the help of DeepLinker, desktop users are able to browse their files to receive resolvable deep links. By revisiting the links, they directly arrive to the desired fragment again, a mechanism which is usually not present in native desktop environments. In special collaborative working scenarios, links can be shared in a team which makes explaining how to reach certain parts of files obsolete. Search engines could return deep links in their search results to refer to specific parts where search terms are found. Having such hyperlinks also enable users to


/filesystem/pictures/a.png/content/to@image/rect@600,109,188

Rect Resource

SPARQL Endpoint stores triples about currently viewed resource.

Add Triple

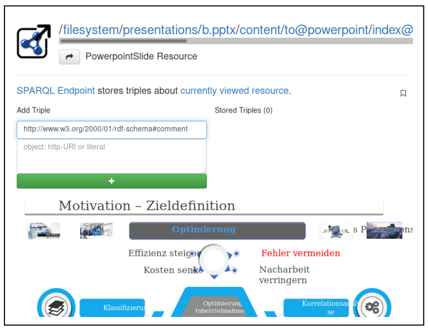
http://www.w3.org/2000/01/rdf-schema#comment

object: http-URI or literal

Stored Triples (1)

rdfs:comment	Artificial	Remove

(a) /a.png/content/to@image/rect@600,109,188,36


/filesystem/presentations/b.pptx/content/to@powerpoint/index@

PowerpointSlide Resource

SPARQL Endpoint stores triples about currently viewed resource.

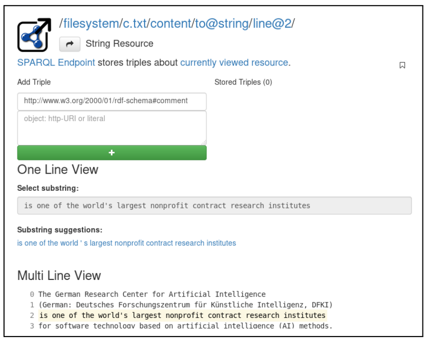
Add Triple

http://www.w3.org/2000/01/rdf-schema#comment

object: http-URI or literal

Stored Triples (0)

(b) /b.pptx/content/to@powerpoint/index@3/cssSelector@svg%2B%253E%2Bg%2B%253E%2Bg%253A...


/filesystem/c.txt/content/to@string/line@2/

String Resource

SPARQL Endpoint stores triples about currently viewed resource.

Add Triple

http://www.w3.org/2000/01/rdf-schema#comment

object: http-URI or literal

Stored Triples (0)

One Line View

Select substring:

```
is one of the world's largest nonprofit contract research institutes
```

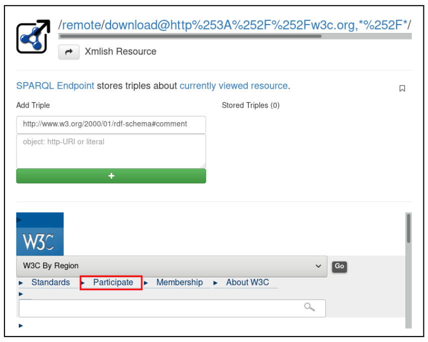
Substring suggestions:

is one of the world ' s largest nonprofit contract research institutes

Multi Line View

```
0 The German Research Center for Artificial Intelligence
1 (German: Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI)
2 is one of the world's largest nonprofit contract research institutes
3 for software technology based on artificial intelligence (AI) methods.
```

(c) /c.txt/content/to@string/line@2


/remote/download@http%253A%252F%252Fw3c.org,*%252F*/

Xmlish Resource

SPARQL Endpoint stores triples about currently viewed resource.

Add Triple

http://www.w3.org/2000/01/rdf-schema#comment

object: http-URI or literal

Stored Triples (0)

(d) /remote/download@http%253A%252F%252Fw3c.org,*%252F*/content/to@html/cssSelector@...

Figure 14: Screenshots from DeepLinker showing deep links, GUI elements and referred parts. Some links are shortened due to their lengths. Images have already been published in [141].

annotate file contents by making statements about them in RDF. This way, constructed KG resources can be associated with desktop users' personal data in a Semantic Desktop environment to semantically describe contents.

Limitations

Limitations due to the approach's prototypical state are discussed next. DeepLinker is a separate tool running beside already existing, well-established and frequently used desktop applications. Therefore, a lower willingness from users to use DeepLinker for browsing files is to be expected. Moreover, in case file contents change over time, generated deep links are not robust to such modifications, because they are composed of names, indices and selectors. Although, the concept of hyperlinks should be familiar for non-technical users who are used to browse the Web, the annotation process using KGs and RDF can be very unusual to them. The creation of bookmarks during

Web surfing might be the annotation activity they probably can relate to. Still, getting used to deep linking needs training and takes time.

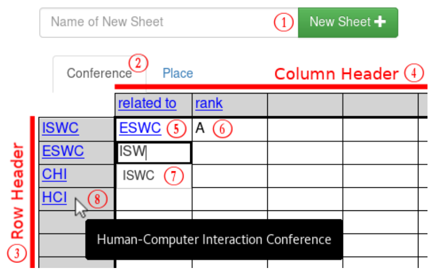
Besides linking, users also need a way to freely enter their expertise in form of instances and their relations. The following approach considers this with an interactive spreadsheet.

3.3.4 *RDF Spreadsheet Editor*

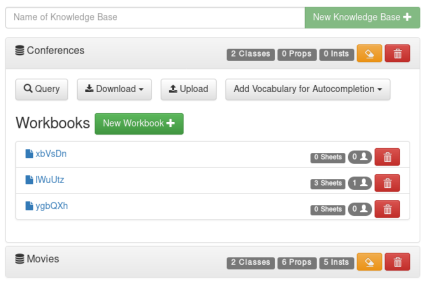
In order to gather especially tacit knowledge about a certain domain, employees who work in this field are usually consulted. One way to involve them is to provide a software which lets them directly and actively communicate their expertise. Ontology editors, like the Protégé editor [108], enable users to model a domain with classes and properties. However, such tools require considerable training and technical knowledge to be configured and used.

Since spreadsheets are widely known in this user group, its metaphor can be exploited for transferring knowledge by entering data manually. This idea leads to the proposed RDF Spreadsheet Editor approach: its sheet generates RDF statements the moment data is entered in cells. With this it enables users to enter and also modify RDF statements in a familiar way. The editor can be utilized before and during KG construction. Starting with an empty graph, the focus lies in incremental instance creation through domain experts. Having a filled KG, the approach is able to present its content in spreadsheets, while edits immediately result in KG changes. Since such spreadsheet modeling typically involves a lot of communication and collaboration, RDF Spreadsheet Editor is implemented as an interactive Web application, thus contributors instantly see each others changes in a sheet. The zero-configuration spreadsheet itself considers a fixed *class per sheet*, *entity per row* and *property per column* mapping, comparable to a conversion procedure from tabular data to RDF [185]. It allows users to manage classes, properties, instances and assertions with a focus on the ABox level. Therefore, the tool has limited ontology engineering features for the TBox.

The basic functionality of RDF Spreadsheet Editor is demonstrated in Figure 15a. Its GUI screenshot is annotated with red numbers indicating basic features. (1) By naming and adding a sheet, (2) a new labeled class and a corresponding sheet is created. All labels' language tag will be English, since this is the selected language in the example. (3) A row header lists labeled instances of the class. (4) Similarly, the column header creates labeled properties which have in their domain the sheet's class. (5) By entering a name in a cell, a labeled resource is instantiated and a statement is formed using the row header's resource and the column header's property. This statement creation is independent of the order in which the cells are populated. (6) To create a literal instead, a preceding single quotation



(a) Graphical user interface of a workbook's sheet with annotated features in red.



(b) An admin view to manage RDF datasets and their workbooks.

Figure 15: Screenshots from RDF Spreadsheet Editor (have been published in [149, 150]).

mark has to be used. This is not shown in the screenshot, since the cell is not edited at the moment. (7) Auto completion helps to pick a resource by its label. (8) In case a resource has an `rdfs:comment`, it is shown when mouse hovering the cell. Listing 6 presents a subset of the associated RDF triples generated by the editor (for readability reasons the URNs' UUIDs are shorted).

Besides the editor's basic features, there are other helpful ones which simplify its usage.

AUTO COMPLETION AND COPY&PASTE To refer to already existing resources, an auto completion feature suggests them by their labels. Additionally, it is possible to copy and paste a resource from one cell to another. Unlike a simple copy of text in spreadsheets, this operation reuses the resource's URI which enables interlinking.

COMMENTS Additional to resource labels, comments can be provided which is helpful in case of disambiguation. Once a resource in a cell is focused, a text area lets users enter a comment which is shown when mouse is hovering it.

DATA TYPES OF LITERALS When a literal is entered in a cell, a data type is automatically suggested: Floating-point numbers receive the type `xsd:float`, while integer numbers are typed `xsd:int`. Should "true" or "false" be entered, `xsd:boolean` is used. If non of them apply, a language string `rdf:langString` is assumed in the language the sheet is configured.

HTTP URIS URIs in the editor are randomly generated with UUIDs. However, if a hyperlink is entered or pasted (starting with `http(s)://`), it is used for the resource's URI.

LABEL MODIFICATION When a cell's content is modified, a resource's label or a literal statement is changed accordingly. If its text is

Listing 6: A subset of the modeled statements in RDF Spreadsheet Editor shown by a screenshot in Figure 15a (has been published in [150]).

```
<urn:uuid:1cfd> a              rdfs:Class ;                 # (2)
                rdfs:label     "Conference"@en .

<urn:uuid:99f2> a              <urn:uuid:1cfd> , owl:Thing ;  # (3)
                rdfs:label     "ISWC"@en ;
                <urn:uuid:ccf1> <urn:uuid:76b9> ;            # (5)
                <urn:uuid:6942> "A"@en .                     # (6)

<urn:uuid:76b9> a              owl:Thing ;                  # (5)
                rdfs:label     "ESWC"@en .

<urn:uuid:ccf1> a              rdf:Property ;               # (4)
                rdfs:label     "related to"@en ;
                rdfs:domain    <urn:uuid:1cfd> .

<urn:uuid:6942> a              rdf:Property ;               # (4)
                rdfs:label     "rank"@en ;
                rdfs:domain    <urn:uuid:1cfd> ;
                rdfs:range     rdf:langString .
```

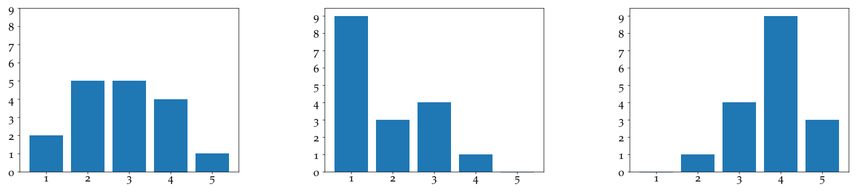
set empty, only the associated statement is removed from the RDF graph, but not any resources. This way resources can be reused later in other cells.

ADMIN VIEW A separate admin view is integrated (Figure 15b) to create and delete RDF datasets. It additionally features a separate page to perform SPARQL queries, exports (downloads) and imports (uploads) of triples in typical RDF serialization formats and the use of common vocabularies (like FOAF [28]). For each dataset, workbooks can be created and their links can be shared with collaborators.

User Study

To evaluate the usability of RDF Spreadsheet Editor, a user study with 17 participants (13 male, 4 female, avg. age 31 $\pm$ 9.6) was conducted. As a baseline, the tool is compared with the ontology editor Protégé (Version 5.2.0) [108] and the writing of RDF using the Turtle syntax [183].

To assess the prior knowledge of the participants, a questionnaire asked them to estimate how experienced they are regarding RDF, Protégé and spreadsheets in general on a scale between *novice* (1) and *expert* (5). Figure 16 visualizes the results on individual histograms. They show that RDF skills are nearly normal distributed, while Protégé is mostly unknown to them and, conversely, spreadsheets are well-known.



(a) RDF (b) Protégé (c) Spreadsheet

Figure 16: Estimations of participants' experiences on a scale between *novice* (1) and *expert* (5) using histograms (has been published in [150]).

The task in the user study was to model as concise as possible the following information in RDF: *"Max attends the conference ESWC 2017. The ESWC 2017 is located in Portoroz. The keywords Semantic Web and Knowledge are related to ESWC 2017. Portoroz is a city and lies within the country Slovenia"* [150]. Each participant were ask to sequentially use the three tools, but in randomized order to reduce learning effects. In case of writing Turtle, participants have chosen their familiar text editor which was Notepad++[5], Atom[6] or Sublime Text[7]. For Protégé a short tutorial was provided and questions about its usage were answered during the task. Regarding RDF Spreadsheet Editor, a screenshot was shown to briefly explain its basic functionality.

For each tool and participant the time to finish the modeling task and the number of created triples were measured. Should the Turtle syntax have parsing errors (13 of 17 cases), statements were counted manually. A scatterplot visualizing number of triples per time for Turtle (○), Protégé (×) and RDF Spreadsheet Editor (△) is presented in Figure 17. The plot clearly shows that RDF Spreadsheet Editor lets users create more RDF statements in less time compared to the baselines. In fact, the approach archived on average 13.8 statements per minute which is approximately four times more than using Protégé (3.4) and writing Turtle (3.1). Participants reported during the use of Protégé that they are overwhelmed by the GUI's richness which also underlines their low experience with the tool (Figure 16b). Bulk editing capabilities seems to be not available in Protégé or were unknown to the participants. Although, writing Turtle in a text editor does not provide a GUI with buttons and lists, its typical features like multiple cursors, copy&paste operations, regular expression based searches and replacements, auto completion, etc. seems to compensate this.

Right after task completion with RDF Spreadsheet Editor, participants were asked to complete a User Experience Questionnaire (UEQ) [100] for it. The UEQ derives six factors, namely attractiveness, perspicuity, efficiency, dependability, stimulation, and novelty. They are

5 https://notepad-plus-plus.org/
6 https://atom.io/
7 https://www.sublimetext.com/

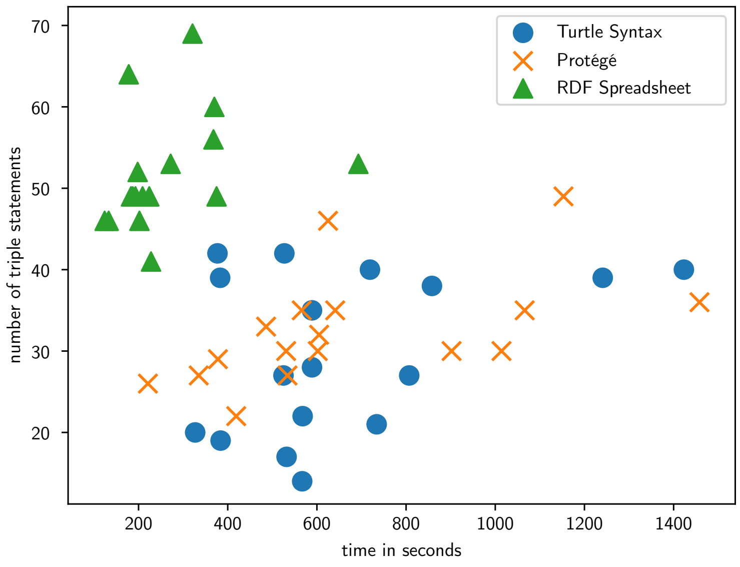


Figure 17: For each participant the number of statements per time for writing Turtle syntax (○), using Protégé (×) and RDF Spreadsheet Editor (△) (has been published in [150]).

presented in Figure 18. Excellent scores have been archived in attractiveness, efficiency and perspicuity, while novelty has ben received the lowest value. These results can be explained by the use of the spreadsheet metaphor: participants felt to work in a familiar environment which is perceived efficient and perspicuous, but naturally not novel.

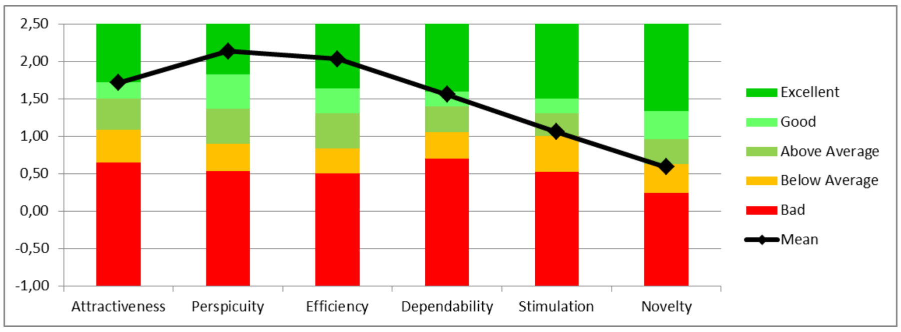


Figure 18: Results of a user experience questionnaire [100] for RDF Spreadsheet Editor (has been published in [150]).

Limitations

Although, RDF Spreadsheet Editor performed well compared to the baselines, it has some limitations. Once a resource's property has more than one object, multiple entities have to be entered in a cell (as discussed in Section 2.1) or the property has to be repeated in the column header. One solution to handle such cardinalities could be multiple inner cells in an outer cell as proposed by the Related Worksheets approach [11]. However, the editor's readability and usability could decrease in such cases. This is also true when large RDF datasets are edited, with a high number of classes, instances and properties. Typical searching, sorting and filtering capabilities in spreadsheets could help users to keep an overview. Regarding ontology engineering, RDF Spreadsheet Editor provides very limited features which could be compensated with dedicated GUI modules. For example, a tree view is better suited for the management of class and property hierarchies than a table.

3.4 SOFTWARE DEVELOPERS

The group of software developers (or software engineers) have a rather technical perspective on a domain. Usually, they implement services for knowledge workers, model data structures and access (legacy) data stores. Because of their tasks in an enterprise, this group has background knowledge about the domain, but not as profound as domain experts. Although, software developers have a higher technical understanding, they may not be familiar with Semantic Web standards, semantic technologies and KGs in general (see topics described in Section 1.3). Since their time is usually limited, additional training to study these new technologies is often not granted. To still involve them in a KG construction project, dedicated HumL approaches for them become necessary. One way could be the deployment of an interface which lets them access KGs through technologies they are familiar with. In Section 3.4.1 such an API is proposed to enable them to work with KGs. A similar idea is the transformation of constructed KGs to more familiar data structures. Section 3.4.2 covers such an approach which is able to convert RDF datasets to Relational Databases (RDBs). With these methods, developers become able to implement software modules which read from and also write to such graphs. Especially writing operations enable them to contribute to KG construction and maintenance tasks after its deployment. Reading operations on familiar RDB models let them easily analyze constructed contents. An active involvement via software development qualifies developers to also give feedback about modeled structures.

Parts of this section have already been published in [140, 153].

3.4.1 *SPARQL REST API*

SPARQL [181] is the standard query language to access an RDF triplestore. However, to use and understand its mechanisms and behavior, fundamental topics about Semantic Web technologies (see Section 1.3.1) need to be learned first. To reduce this initial hurdle for Semantic Web newcomers, an approach is proposed which turns a given SPARQL endpoint into a JSON-based REST API. Since developers are commonly used to these Web technologies, they instantly become able to perform Create, Read, Update, and Delete (CRUD) operations on KGs and implement services on top of that. This way, developers are enabled to take part in KG construction and maintenance tasks by providing suitable software assets. To archive this, the proposed approach called SPARQL REST API provides two mechanisms: First, it uses the path metaphor in a request URL to let developers navigate through the KG. Second, it bidirectionally transforms graph representations into an object view using a nested JSON format.

Listing 7 summarizes the possible paths by providing a formal grammar. The API considers two main entry points which are `/resource` and `/class` (Line 1). While the former continues with a resource's CURIE to query it (Line 2), the latter is used to browse instances of a given class (Line 3). Both use the `PATH` rule (Line 4) to traverse the graph by alternating resources (`RES`) and properties (`PROP`). Optional expressions (`RQL`) can be added to further filter the result.

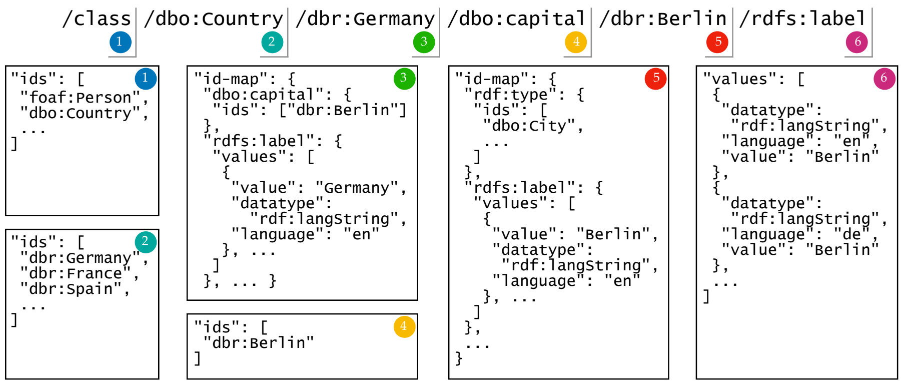
Listing 7: Part of the formal path grammar of SPARQL REST API (has been published in [140]).

```
1 API_PATH = "/api" (RESOURCE | CLASS)
2 RESOURCE = "/resource" PATH RQL?
3 CLASS    = "/class" RES PATH RQL?
4 PATH     = ("/" RES "/" PROP)* ("/" RES)?
```

The results are rendered as JSON objects [39] which contain a reached resource's outgoing edges. Resources are identified with CURIEs which are listed in the output as simple JSON arrays of `ids`. Analogously, literals are itemized in `values` arrays. However, to ensure a correct round trip between output and input (writing operations), their meta data is also given in form of `value`, `datatype` and `language`. Special `id-maps` are JSON objects mapping predicate CURIEs to their RDF objects. Once responses consist of complex subgraphs, the elements `ids`, `values`, `id-map` and `value-map` are nested.

Figure 19 shows how SPARQL REST API browses DBpedia [10] given some exemplary paths and results. Through the `/class` endpoint which lists classes (1), `dbo:Country` is selected resulting in a response containing countries (2). By adding `dbr:Germany` as a path segment, its outgoing edges are returned (3). Following the path metaphor, an additional property `dbo:capital` just returns the desired value (5). This traversal through the graph can be continued arbitrarily long (5 & 6).

Besides reading with HTTP `GET` requests, the API also supports the other CRUD operations. To create resources, `POST` and `PUT` requests expect the same JSON payload as shown in `GET` requests enabling a seamless round trip. While `POST` creates new resources with statements passed in the request's payload, `PUT` updates (or creates) a



```
"ids": [                         1
  "foaf:Person",
  "dbo:Country",
  ...
]
```

```
"ids": [                         2
  "dbr:Germany",
  "dbr:France",
  "dbr:Spain",
  ...
]
```

```
"id-map": {                      3
  "dbo:capital": {
    "ids": ["dbr:Berlin"]
  },
  "rdfs:label": {
    "values": [
      {
        "value": "Germany",
        "datatype":
           "rdf:langString",
        "language": "en"
      }, ...
    ]
  }, ... }
```

```
"ids": [                         4
  "dbr:Berlin"
]
```

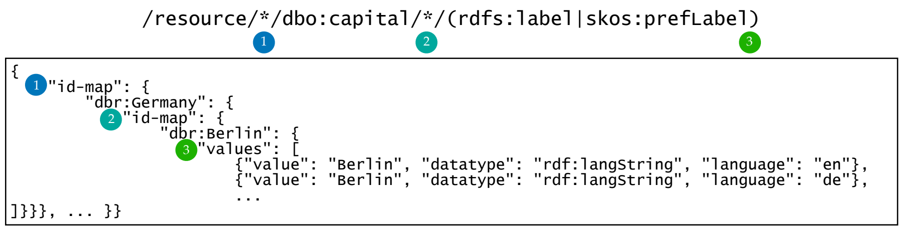
```
"id-map": {                      5
  "rdf:type": {
    "ids": [
      "dbo:City",
      ...
    ]
  },
  "rdfs:label": {
    "values": [
      {
        "value": "Berlin",
        "datatype":
           "rdf:langString",
        "language": "en"
      }, ...
    ]
  },
  ...
}
```

```
"values": [                      6
  {
    "datatype":
       "rdf:langString",
    "language": "en",
    "value": "Berlin"
  },
  {
    "datatype":
       "rdf:langString",
    "language": "de",
    "value": "Berlin"
  },
  ...
]
```

Figure 19: Paths and their JSON responses of SPARQL REST API browsing through DBpedia (has been published in [140]).

resource identified through a given path. `DELETE` operations remove resources depending on the number of used path segments: if only the resource is mentioned (one segment), all its incoming and outgoing RDF statements are removed. Is a resource and property mentioned (two segments), the properties outgoing edges are erased. In case three segments are defined, only the specified SPO triple is removed.

Additional features in the API provide more expressiveness for users. An asterisk wildcard '∗' (known from Linux shells) can replace any `RES` and `PROP` segments in a path. Its "all of them" semantic introduces an additional nesting level in the response which creates a partial view of nested objects. Moreover, at `PROP` positions the API supports the use of SPARQL property paths [180] (indicated by parentheses) which enable multiple hops, inverse directions and alternatives. The property path itself is collapsed (i.e. not shown) in the result. An example of these features is presented in Figure 20. Two wildcards in (1) and (2) extend the response with all resources having capitals using nested `id-maps`. Berlin's normal and preferred labels are listed by using the alternative path operator '|' (3).



```
/resource/*/dbo:capital/*/(rdfs:label|skos:prefLabel)
```

```
{
 1"id-map": {
    "dbr:Germany": {
     2"id-map": {
        "dbr:Berlin": {
         3"values": [
            {"value": "Berlin", "datatype": "rdf:langString", "language": "en"},
            {"value": "Berlin", "datatype": "rdf:langString", "language": "de"},
            ...
]}}}, ... }}
```

Figure 20: Demonstration of wildcards and property paths in SPARQL REST API (has been published in [140]).

SPARQL REST API also implements some Resource Query Language (RQL)[8] operators, such as filters like `limit`, `sort` and `regex` as well as aggregations like `sum`, `avg`, and `count`. To reduce the number

8 https://github.com/persvr/rql

of requests, batch processing is also provided to bundle similar operations to a single JSON-RPC [93]. An additional `/namespace` entry point can be used to manage namespace prefixes. To also be able to traverse BNodes in the graph, they are automatically skolemized [103] to URIs by the API.

Limitations

Although, developers can communicate with a KG through SPARQL REST API without extensive training, there are still some issues with the tool to be discussed. Since the path metaphor lets them traverse through the graph, its structure is transparent to users. Together with the virtual JSON object view showing finite documents, this can lead to misconceptions in their mental model. For example, paths can be formed to traverse a cycle in the graph. Consequently, different paths in the graph can lead to the same resource which makes URLs for them not unique. Due to the support of a round trip, RDF literals are complex JSON objects rather than simple JSON literals (strings, numbers, Boolean values) which can be odd for developers. Although, SPARQL is an expressive query language, only rather simple queries are possible with the API due to its intentionally reduced functionality.

SPARQL REST API is a wrapper around a SPARQL interface, so data is converted at query time. However, if analyses involve a large part of the graph, a high number of requests will be necessary. For such cases, a complete conversion of the KG to a familiar data structure is a more appropriate solution. The next section covers such an approach.

3.4.2 *RDF2RDB-REST-API*

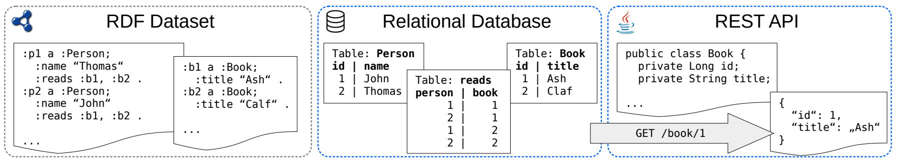
Regarding the application of KGs, the following experiences have been made during research in several industry-related projects. Semantic Web standards and their technologies as described in Section 1.3.1 are rather seldom used in industry. This observation was also made by a case study in the manufacturing domain [57]. When it comes to system critical data, the storage of choice are often Relational Databases (RDBs), since they are researched for over 50 years now [15]. This technology is often complemented with appropriate Create, Read, Update, and Delete (CRUD) APIs to access and manipulate data more conveniently. Distinct solutions from Semantic Web and industry regarding storage and query methods are summarized and compared in Table 14. While triplestores with RDF statements expressing assertions and ontologies are preferred by the Semantic Web community, industry is used to store database tables with schemata. The identification with URIs is opposed to primary keys in RDBs. SPARQL can be compared with Structured Query Language (SQL) queries or dedicated REST calls.

Table 14: Comparison table between Semantic Web and Industry in storage and query approaches (has been published in [153]).

Approach	Semantic Web	Industry
Storage	Triplestore	SQL Database
Domain Modeling	Terminology in Ontology	Database Schema
Data Modeling	RDF Statement Assertions	Database Records
Identification	URIs	Primary Keys
Query Interface	SPARQL	SQL / REST API
Exchange Format	Result Set / RDF	Result Set / JSON

To give developers the chance to work with semantic data, but in a familiar storage system (like RDBs), RDF datasets could be completely transformed to another data model. With the known trade-off to lose the flexibility of semantic technologies, they are able to query and maintain generated data in familiar stores without much prior knowledge. An additionally generated CRUD REST API would enable developers to directly work with converted data, comparable with the previous approach (Section 3.4.1). Regarding HumL integration, after a KG is constructed and deployed, developers can receive a converted version of it which can be queried and analyzed by them without further ado. However, such one-way transformations imply that for each new KG version a conversion step has to be performed again.

To provide a fully automated solution, the approach RDF2RDB-REST-API is proposed and illustrated in Figure 21. The example shows an RDF dataset about persons reading books (left) which is automatically converted by the approach to three RDB tables (middle): `Person`, `reads` and `Book`. Additionally, Java source code is generated which includes corresponding Java classes and a server implementation to provide a REST API, for example to `GET` the first book in a JSON representation [39]. To archive this, RDF2RDB-REST-API has three phases:



```
:p1 a :Person;
  :name "Thomas"
  :reads :b1, :b2 .
:p2 a :Person;
  :name "John"
  :reads :b1, :b2 .
...
```

```
:b1 a :Book;
  :title "Ash" .
:b2 a :Book;
  :title "Calf" .
...
```

```
Table: Person
id | name
 1 | John
 2 | Thomas
```

```
Table: reads
person | book
     1 |    1
     2 |    1
     1 |    2
     2 |    2
```

```
Table: Book
id | title
 1 | Ash
 2 | Claf
```

```
public class Book {
  private Long id;
  private String title;
...
```

```
{
  "id": 1,
  "title": „Ash"
}
```

Figure 21: Illustration of the RDF2RDB-REST-API approach (has been published in [153]): an RDF dataset (left) is transformed to RDB tables (middle) and server code implemented in Java (right).

first, it analyzes given RDF statements to derive a suitable database model in the second step, while the third step generates Java code.

Analysis of RDF. To simplify the processing, a first step is to skolemize all BNodes [103] by consistently substituting them with randomly generated URIs. In a next step, classes and their instances are collected by scanning through all assertion statements (ABox). This also includes the examination of properties with their domain and range information. According to OWL [177], properties are divided into data type properties (literal range) and object properties (resource range). For the former, suitable SQL storage classes (text, real, integer or binary large object) are found for the literal types. For the latter, resource types are looked up and dangling resources are detected if they are not further mentioned in the dataset or have no type. In both cases, by scanning through the ABox again, their cardinalities are inspected which can be one-to-one, many-to-one, one-to-many or many-to-many.

Conversion to RDB. The analysis results from the previous step is used to model RDB tables in a type-store fashion [102], since developer may easier grasp this structure than a vertical or horizontal one. Following this model, an "entity table" is created for each identified type (class), while its instances will be the table's records. As a primary key they have a mandatory *id*-column. Further columns come from RDF properties which have the entity table's class in their domain. However, for certain cardinalities different actions are performed. In case of one-to-many, to satisfy the third normal form [94] the referring table will receive the column. In case of many-to-many, a separate table with two columns is created which refers to subjects and objects.

After the RDB schema is defined, tables are filled with records. First, for each RDF resource a unique numeric ID is assigned to have distinct primary keys in entity tables. The outgoing edges' objects are used to allocate the records' fields. Many-to-many tables are filled with subject and object IDs of its associated property if subject types match its domain and object types match its range.

Finally, SQL syntax is generated to create a SQLite[9] database with all tables and records.

Generation of REST API Code. The template engine Apache FreeMarker[10] is utilized to generate Java source code in two separate projects. For an *api* project Plain Old Java Objects (POJOs) [62] are created for each entity table, while class attributes reflect table columns. Compatible many-to-many tables become `java.util.List` attributes due to their multiple values once joined. A special `LangString` POJO class is used for language string properties to capture language meta data. To control the database, an SQL-based controller is generated for each entity table which allows to select, insert, update and delete records with corresponding Java methods. A separate *server* project includes the *api* project to reuse POJOs and database access methods. By using the Spark framework[11], a RESTful server is implemented which

9 https://www.sqlite.org/
10 https://freemarker.apache.org/
11 http://sparkjava.com/

provides endpoints for each Java class (entity table). The common HTTP methods `GET`, `POST`, `PUT`, `PATCH` and `DELETE` are supported and POJOs are bidirectionally converted to JSON documents to exchange data.

Evaluation

To check how RDF2RDB-REST-API handles various RDF datasets in comparison with a similar approach, an evaluation is conducted.

Dataset Conversions. Six relatively small sized RDF datasets which are randomly selected from the LOD cloud [162] and a generated one from the Berlin SPARQL Benchmark (BSBM) [22] are used to investigate generation effects of the algorithm. Table 15 lists them together with their sizes, number of properties, cardinalities and generation results. As intended, the dataset's classes are reflected by entity tables

Table 15: Dataset characteristics and generation results of RDF2RDB-REST-API which has been published in [153]. Used abbreviations: statements (Stmts), classes (Cls), multi-typed instances (MT), object property (OP), datatype properties (DP), one-to-one (OO), many-to-one (MO), one-to-many (OM), many-to-many (MM), entity tables (ET), many-to-many tables (MMT), average number of columns per ET (avgCol).

Dataset		Size			Properties & Cardinalities						Generation Result		
No.	Name	Stmts	Cls	MT	OP	DP	OO	MO	OM	MM	ET	MMT	avgCol
1	TBL-C[12]	109	5	1	26	18	31	19	7	1	5	11	12.2 ± 12.4
2	CTB[13]	10,853	4	0	10	5	4	6	1	4	4	7	3.0 ± 1.6
3	EAT[14]	1,674,376	2	0	3	3	1	5	0	0	3	1	2.7 ± 2.1
4	Pokédex[15]	26,562	19	0	9	29	13	28	5	3	19	40	3.5 ± 6.7
5	BOW[16]	4,041,676	15	349,195	7	19	2	9	0	15	15	180	5.3 ± 3.0
6	S-IT[17]	4,477	406	81	9	25	18	6	3	7	406	3,056	2.6 ± 1.0
7	BSBM[18]	40,177	22	100	12	28	16	22	0	2	22	17	13.7 ± 5.5

and properties becoming either columns or additional many-to-many relations. The outputs show that databases sufficiently reflect the information content of given RDF datasets. However, several many-to-many tables with equivalent data were unnecessarily generated because of resources with multiple types. This effect can be seen for Dataset No. 6 having the second lowest number of statements, but the highest number of many-to-many tables. Although, Dataset No. 5 has a large number of multi-typed (MT) instances (349,195), only

13 Tim Berners-Lee's electronic business card, http://www.w3.org/People/Berners-Lee/card.rdf
14 Copyright Term Bank, https://lod-cloud.net/dataset/copyrighttermbank
15 Edinburgh Associative Thesaurus, https://lod-cloud.net/dataset/associations
16 Pokémons, https://lod-cloud.net/dataset/data-incubator-pokedex
17 Between our Worlds (Release 2020-06), https://betweenourworlds.org/
18 Traveling website about Salzburg in Italian language, https://lod-cloud.net/dataset/salzburgerland-com-it

180 many-to-many tables are generated. The reason is that the average number of types per MT instance is rather low 2.9 ± 0.9.

Comparison with RDF2RDB. The similar tool RDF2RDB[19] is used as a baseline to compare the output of Dataset No. 1 and Dataset No. 7. Since the authors also provide an output for TBL-C[20] it can be used for comparison. RDF2RDB produced more tables (59 compared to 16) since more properties are converted into many-to-many tables instead of columns. They also produce a separate *labels*-table to enable label-based searches and *uris*-table which lists all resources with some meta data. Regarding the BSBM dataset (No. 7), again RDF2RDB generates more tables (145 compared to 39) because for each product type a table is produced. Entity tables are nearly equal, except some properties are modeled as tables (not columns). To conclude, RDF2RDB tends to generate more tables (especially many-to-many) in order to comply with the RDF model. RDF2RDB-REST-API is more data driven and tries to reduce the number of tables when observed cardinalities allow that.

Limitations

During the evaluation, limitations of RDF2RDB-REST-API could be identified. A main challenges is the handling of instances with multiple types. The problem is that resources which have more then one type are redundantly distributed among tables. Since particular domains are related to certain ranges in many-to-many tables, their number also increase. Such unwanted and unnecessary data redundancies should be avoided or kept to a minimum. Another optimization is the decision whether properties are converted to simple columns or many-to-many tables. When choosing the first extreme, the second normal form would be violated because data redundancy occurs. In case of the other extreme, a lot of tables are produced resulting in more join operations and thus complex queries. This trade-off may be very dependent on given use cases: RDF2RDB-REST-API minimizes the numbers of many-to-many tables by inferring cardinalities from the data. However, this fixes the database schema and makes changing cardinalities later not easy. Since the approach is a one-way conversion, large or frequently changed RDF datasets will take considerable conversion time. To reduce this, an update-mechanism could avoid the rebuilding method.

3.5 SEMANTIC WEB PRACTITIONERS

Users in the group of Semantic Web practitioners model, query and exploit knowledge in form of ontologies and Knowledge Graphs (KGs). They are well-versed with Semantic Web standards, related technolo-

19 https://github.com/michaelbrunnbauer/rdf2rdb
20 https://www.netestate.de/Download/RDF2RDB/timbl.txt

gies and tools (see Section 1.3). While Knowledge Engineers (KEs) focus on the creation of KGs mainly with approaches from Chapter 2, Semantic Web experts contribute to ontology modeling and knowledge service implementation tasks with their expertise and technical skills. Compared to non-technical users, members of this group have usually limited knowledge about the special domain in which they join the project. Therefore, during maintenance of ontologies and knowledge services, they rely on contributions from the other groups: while domain experts provide insights into their topics and workflows, developers add their competency on software engineering and technical environments. Thus, Semantic Web practitioners need ways to foster cooperation and at the same time motivate colleagues to use Semantic Web related technologies.

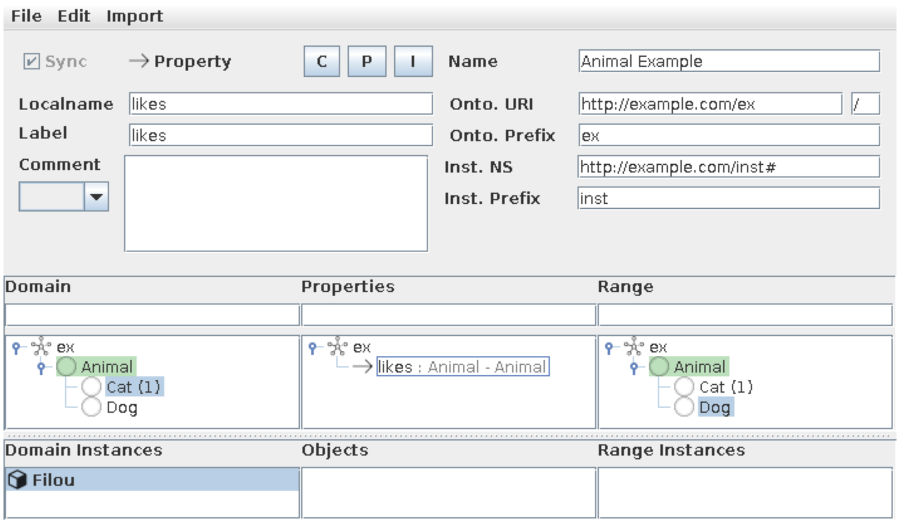
They take part in the KG construction process in two ways: One aspect is the modeling of ontologies, especially in collaboration with other members. To archive this, an ontology editor is proposed in Section 3.5.1 to let them collaboratively model relevant concepts and their relation while they get to know the domain. Another aspect is the rapid prototyping of knowledge services on top of KGs in form of Linked Data (LD) applications. Since this usually requires collaboration among all stakeholders involved, a framework for building such applications is provided in Section 3.5.2.

Parts of this section have already been published in [143].

3.5.1 *Simple RDFS Editor*

When it comes to formally model a terminology, ontologies [68] are commonly used in the Semantic Web community. To create them, several tools have been implemented for ontology engineers. A popular one is Protégé [108] which is a powerful ontology editor designed to create and maintain OWL [177] ontologies. However, its comprehensive features, OWL support and complex GUI, result in a steep learning curve for beginners. Once ontology engineers would like to quickly model less expressive RDFS [184] ontologies, learning to use a feature-rich tool often does not pay off. Therefore, the tool Simple RDFS Editor is proposed which provides a simplified feature set to create and maintain RDFS ontologies.

A screenshot from the desktop application of Simple RDFS Editor is presented in Figure 22. On the left side, the GUI lets users create classes (C), properties (P) and instances (I) which can be labeled and commented in different languages. On the right side, the ontologies meta data like namespaces and prefixes can be configured. Below, class and property hierarchies are modeled and at the bottom class instances and their property objects are listed. The interface makes heavily use of Drag&Drop operations, since it proved to be a convenient way to relate elements. In the following, the tool's features are discussed.



☑ Sync → Property

Name	Animal Example
Localname	likes
Label	likes
Comment	
Onto. URI	http://example.com/ex /
Onto. Prefix	ex
Inst. NS	http://example.com/inst#
Inst. Prefix	inst

Domain	Properties	Range
ex	ex	ex
Animal	likes : Animal - Animal	Animal
Cat (1)		Cat (1)
Dog		Dog

Domain Instances	Objects	Range Instances
Filou		

Figure 22: Screenshot from Simple RDFS Editor showing a small example.

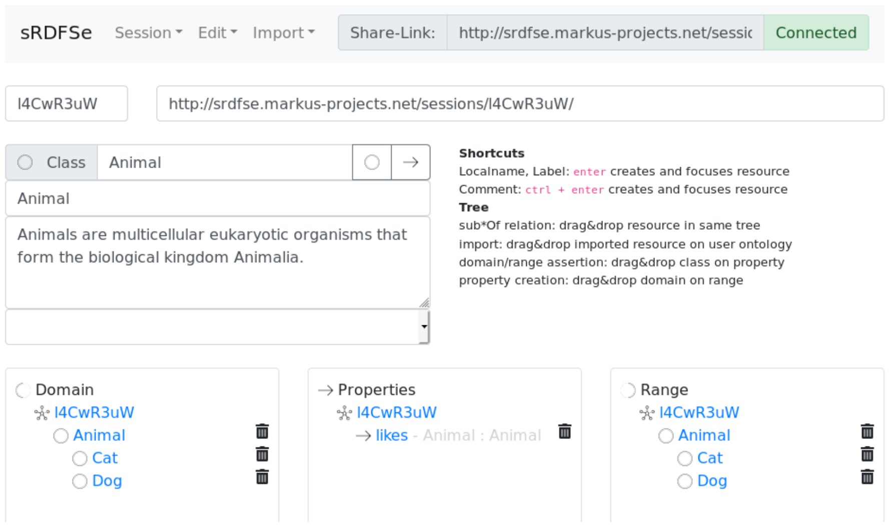
Class and Property Taxonomies. When creating classes and properties by giving them a label, the *Sync* feature automatically provides a local name for the resource by applying camel case and URL encoding. The created top level concept can be inserted into the taxonomy via a Drag&Drop operation. A convenient way to create a property is also done with a Drag&Drop operation from one domain class to a range class which results in a property named `has<RangeClassName>`. Similarly, to assign domain and range information to a property, Drag&Drop from the domain class tree (left) or the range class tree (right) to the property can be performed. A selected property's domain and range classes are highlighted in green.

Class Instances. By selecting a class, an instance of it can be created with a label and comment. For the instance's local name a UUID is automatically suggested. If both a domain instance (subject) and a property (predicate) are selected, the *Objects*-list shows all corresponding RDF objects. Using the range instances, an object can be added by Drag&Drop operation on the list.

Importing Ontologies. Once an external ontology (like FOAF [28]) is imported, their classes and properties can be reused. By a Drag&Drop move to the own ontology, external classes and properties are integrated. For defining literal properties, the XSD [179] ontology is provided in the tool. As usual, users can preview, save and load the created ontology.

Online Version. Protégé is also implemented as a Web-based application called WebProtégé[21] [108]. Inspired by this, Simple RDFS Editor is also implemented as a Web application to allow simultaneous collaborative editing on one ontology. Its GUI which is similar to the desktop version is depicted in Figure 23. On the server, users

21 `https://webprotege.stanford.edu/`



http://srdfse.markus-projects.net/sessic

l4CwR3uW

http://srdfse.markus-projects.net/sessions/l4CwR3uW/

Animal

Animal

Animals are multicellular eukaryotic organisms that form the biological kingdom Animalia.

Shortcuts
Localname, Label: enter creates and focuses resource
Comment: ctrl + enter creates and focuses resource
Tree
sub*Of relation: drag&drop resource in same tree
import: drag&drop imported resource on user ontology
domain/range assertion: drag&drop class on property
property creation: drag&drop domain on range

l4CwR3uW
Animal
Cat
Dog

l4CwR3uW
likes - Animal : Animal

l4CwR3uW
Animal
Cat
Dog

Figure 23: Screenshot from the online version of Simple RDFS Editor.

can create private sessions with a share link to invite collaborators. Although, the Web app offers not all features presented in the desktop version, it lets multiple editors work on one ontology by simply using Web browsers.

Limitations

To facilitate its usage, Simple RDFS Editor has some accepted limitations. It does not support multi-inheritance in class and property taxonomies to keep their tree structures simple. For similar reasons, properties cannot have multiple domains and ranges. Regarding naming, for each language tag only one label and comment can be set. In case of instances, there is currently no support to have objects being literals.

3.5.2 *Linked Data Application Framework*

During knowledge service implementations, Semantic Web practitioners rely on participation and contributions from non-technical users and software developers. A shared Linked Data (LD) application on top of an evolving KG could enable rapid prototyping and may push early discussions about the KG's content. However, implementing such an application can be time-consuming when typical requirements for these user groups need to be met. To reduce initial hurdles for Semantic Web practitioners, a Linked Data Application Framework (LDAF) is proposed that provides guidelines and useful implementations which already recognizes some demands. It is designed to enable project partners who are not acquainted with Semantic Web standards to participate in a LD application. This way, integration of users from all user groups is facilitated.

The general architecture of LDAF is presented by Figure 24 which illustrates how a LDAF-based application (left) is deployed on top of a KG (right). In a typical REST architecture the LDAF server lets users browse linked data resources. A special feature of this approach is that content negotiation (middle) is utilized to serve a linked data resource in a form a user prefers: HTML (non-technical users), JSON (developers) or RDF (Semantic Web practitioners). Web developers become able to implement services using Create, Read, Update, and Delete (CRUD) operations with HTTP `GET`, `POST`, `PUT`, `PATCH` and `DELETE` methods. In the following, major features are discussed in more detail.

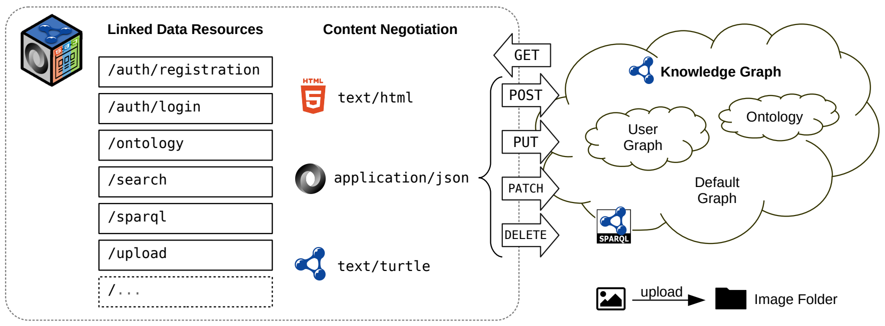


Figure 24: Architecture of LDAF: linked data resources support content negotiation and CRUD operations to maintain a KG.

Authentication. Since a KG project can involve multiple independent users, basic registration and login methods already exist in the framework. For authentication it utilizes JSON Web Tokens[22] which is a Internet standard for encrypted JSON data containing claims. To store a user's personal information (e.g. credentials), a dedicated user graph is associated with the person. Only agents who are logged in have access to other linked data resources.

Linked Data Resources. In a RESTful server, resources are typically identified with URLs which is usually a well-known concept for all participants. Even non-technical users are familiar with them due to their browsing activities in the World Wide Web. Essential for LD is that hyperlinks (URLs) are always resolvable which is ensured by LDAF. Besides authentication, other resources are already provided by the framework which are usually required. The agreed terminology is served in form of a read-only ontology via the `/ontology` entry point. This way, participants can look up classes and properties with their URIs which are also managed by the server. Simple search capabilities with SPARQL and regular expressions are provided by the `/search` resource. Similarly, `/sparql` lets users perform more expressive SPARQL queries. Because resources are often associated with depictions, `/upload` allows to send images to the server which can be linked. Of cause, Semantic Web practitioners can easily extend the server with further LD resources suitable for their use cases. Impor-

22 https://datatracker.ietf.org/doc/html/rfc7519

tant is that all resources are able to return an HTML, JSON and RDF representation to be consumable for the respective user group. The framework provides means to easily configure resources to support CRUD operations and paginated results.

RDF/JSON Conversion. For developers, an RDF resource is bidirectionally converted to a nested JSON document [39] by traversing the RDF graph for a given depth. Resulting JSON objects are always identifiable with `uri`, `path` and `localname`. They list other RDF properties as object keys, while incoming edges are listed in a special `_incoming` object. The converter aims for a good trade-off between lightweight JSON representation and the expressiveness of RDF. This way, Web developers who are not familiar with Semantic Web concepts get a more intuitive object view on RDF resources, similar to the SPARQL REST API approach (Section 3.4.1). The related RDF serialization JSON-LD [186] is not considered, since it potentially introduces 23 `@`-notations (e.g. `@context`) and an sometimes unintuitive nesting which would require additional training to understand.

HTML Rendering. The framework makes use of the template engine Apache FreeMarker[23] to render HTML pages for non-technical users. They are designed with HTML, CSS and JavaScript code and reflect the information of the resource's JSON representation. In the Web page, users are supported with dynamic forms to change linked data conveniently.

LDAF has been successfully applied in a side project which manages Linked Data (LD) about video games. The so-called Linked Open Game Data[24] (LOGD) platform lets users collaboratively create and browse a linked dataset of games, releases, franchises and platforms. By using private user graphs, members keep track of their gaming progress, maintain their digital inventory and compose wish lists. Using a dedicated ontology, video games can be classified by linking to game-related concepts.

Limitations

During practical applications of LDAF, some design decisions show limitations. Changing the ontology requires to adapt HTML representations which can cause considerable effort. A suitable GUI generation approach for Web pages could reduce manual implementation time. The avoidance of JSON-LD results in JSON messages having a proprietary format without proper context to RDF concepts. Currently only local triplestores are supported, however in some use cases it is useful to build LD applications on top of externally hosted KGs.

23 `https://freemarker.apache.org`
24 `http://logd.markus-projects.net/`

3.6 CONCLUSION

In this chapter various Human-in-the-Loop (HumL) approaches were presented to involve users in the KG construction process. Proposed approaches were characterized by their intended user group and applied construction phase. Investigations in related work showed that comparable methods rely on crowdsourcing, games with a purpose and interface mediation. Since crowdsourcing and games do not really fit in the given enterprise scenario, more office-related applications for small user groups were introduced. To involve domain experts having a non-technical background in particular, GUIs were proposed which exploit well-known metaphors on personal data. To initialize KGs, Concept Miner lets users select promising terms from their personal information sphere. KECS applies an improved version of that on file names and shows in an interview setting that AI support can reduce manual modeling effort. Using DeepLinker, users are able to associate their personal desktop files on a fine-grained level with KG resources. RDF Spreadsheet Editor uses the spreadsheet metaphor which allows users to quickly enter RDF statements in a KG. After KG deployment, software developers get involved with interfaces to acquainted protocols and conversions to familiar data structures. Here, SPARQL REST API provides them a RESTful and JSON-based CRUD interface to a SPARQL endpoint. Similarly, RDF2RDB-REST-API lets them convert whole KGs to relational databases together with suitable REST APIs. The group of Semantic Web practitioners take part with ontology modeling and implementations of knowledge services. While Simple RDFS Editor lets them collaboratively model ordinary RDFS ontologies, LDAF provides a framework for implementing LD applications.

After investigating various HumL approaches, concepts and tools, an answer is formulated for the second research question (Section 1.2): **How can domain experts be integrated in the construction process?**

To integrate domain experts, it is helpful to classify them in different user groups first. This way, HumL approaches can be designed for a certain audience regarding its prior knowledge, skills and requirements. Additionally, it is useful to become aware in which construction phase the approach is applied. While acquisition and curation tasks focus on getting expert feedback to establish a KG mainly in a start-up phase, after a KG's deployment, the attention shifts towards maintaining its content through services.

Users may come in contact with Semantic Web related topics (see Section 1.3) when feedback is required during construction of KGs. Especially for non-technical users this can be an overwhelming experience which reduces their ability to give proper feedback. Therefore, HumL approaches should hide such complexity and communicate semantic data in a form users can comprehend. To accomplish this, it has been proven (a) to exploit metaphors which are familiar to users and (b) to apply approaches close to their personal environment. The spreadsheet metaphor was successfully applied in RDF Spreadsheet Editor to let users formulate RDF statements. A user study with 17 participants showed that they were able to create more statements in less time compared to baselines. Moreover, in a User Experience Questionnaire (UEQ) very positive feedback were given to its perspicuity and efficiency. The hyperlink metaphor used in DeepLinker lets them interlink personal information with concepts on their (semantic) desktops. Similarly, the path metaphor in SPARQL REST API allowed developers to browse a KG. If there is no appropriate one and users find it difficult to operate GUIs, an interview setting in KECS has shown to be a suitable solution to still collect feedback. Regarding personal environments, Concept Miner, KECS and DeepLinker also pointed out that domain experts should be integrated in the construction process by asking them about their personal concepts in their familiar environments from their own data. This bottom-up approach ensures that the most relevant concepts in an enterprise are considered in the KG, while at the same time they are described by experts who know them well. KECS showed in a case study with four expert interviews that personal concepts indeed hide in file names and that they can be modeled in a KG with moderate effort. Additionally, it demonstrated that such modeling can be complemented with all kinds of AI support.

To involve developers in maintaining a deployed KG, reading and especially writing (i.e. create, update and delete) operations should be provided with familiar interfaces. SPARQL REST API provides such an interface to consume and manipulate graph data via well-known JSON objects. With RDF2RDB-REST-API RDF graphs can be converted to widely known Relational Databases (RDBs) with reasonable tables. Applications in industry projects showed that these methods increased the willingness of developers to work with KGs and related technologies, although they were not familiar with them. Once the group is actively involved, they become qualified to give valuable feedback about the modeled knowledge.

Semantic Web practitioners should be integrated early with ontology modeling tasks to become aware of the terminology in the domain. Collaborative modeling allows to integrate several participants in this process. Early developed LD applications additionally enable rapid prototyping and push discussions between project members about the evolving KG.

4 DATASET GENERATION

This chapter is about the generation of datasets which are used for the evaluation of Knowledge Graph (KG) construction methods. Parts of this chapter have already been published in [134, 144, 152].

4.1 OPEN DATASET PROBLEM

Evaluations are conducted in computer science research to assess the performance of approaches. Using a quantitative method, data is collected and analyzed to draw conclusions from evidences. To give an example, the performance of machine learning models are typically tested on labeled datasets [169, Evaluation phase]. To receive evaluation results, a task (i.e. challenge to be solved) is represented by annotated datasets which contain inputs associated with their expected outputs (i.e. annotations or labels). For the challenge of constructing KGs from messy enterprise data, such a dataset could also be compiled: input would be data in practice with a certain degree of messiness, while output would be KG statements expected to be constructed from that. Such a dataset could be shared in communities to let other researches reproduce results and compare with them. However, data assets from companies are almost always confidential, since they can contain, for example, private data about products or services and personal information about customers or employees. Justifiably, stakeholders are not interested or allowed to share industry or desktop data for research purpose. Even in certain cases where a selected fraction can be published, considerable annotation efforts remain for adding ground truth data.

Related to enterprise data, some works in literature consider the collection of data about Personal Information Management (PIM). Abela et al. [1] collected in a controlled experiment browsing activities using an RDF model. However, this dataset which was promised in the future to be available, has unfortunately never been published. Moreover, activities solely on the Internet do only reflect a small fraction of PIM. Kim and Croft [95] suggest to create a pseudo-desktop collection by crawling publicly available documents about people mentioned in email threads using a search engine. Although, desktop-related documents of different types can be gathered, the authors admit that typical elements are missing, like user activities, folder hierarchies and file meta data. Gonçalves [67] discusses the representativeness of such datasets and identifies three missing links: autobiographic information, meaning and ground truth. For these reasons, there seems to be no available and suitable PIM datasets that could be utilized to evaluate KG construction from enterprise data.

Still, there might be some possibilities to compile such a dataset. A workaround for data confidentiality could be **obfuscation** which would make its content unclear, obscure, and unintelligible. For example, words could be consistently swapped with synonym, related or nonsensical ones, depending on the degree of obfuscation. This method would lead to authentic datasets, however it requires high effort to correctly carry out the obfuscation for several data types. The additional and remaining fear of possible de-obfuscation (like for graph de-anonymization [84]) does not make this approach very attractive. Another idea could be the **compilation** of a dataset **from public sources**, like for the pseudo-desktop collection in [95]. Especially open government data portals (e.g. from U.S.[1] or Germany[2]) or company data which has been made public [96] provide real data from various domains. However, such data is usually not related to PIM, is not cohesive enough and again requires considerable labeling effort. Another option might be the **generation** of synthetic datasets inspired by real ones. This way, data and its labels could be generated, while software agents could simulate user behavior. An algorithm would be able to produce large and diverse datasets, especially with labels (ground truth). Yet, an artificial dataset might miss certain aspects a natural one has and might be not be authentic.

Still, to overcome the open dataset problem, a generation approach seems to be the most promising option for evaluating KG construction. Therefore, methods are proposed in this PhD thesis which generate datasets, after related work is discussed in the next section.

4.2 RELATED WORK

Data generation is a common way to acquire datasets for the purpose of testing systems (for a survey see [117]). Approaches in literature often propose a language to let users define a data model and configure the generation process. Bruno and Chaudhuri [30] specify a Data Generation Language (DGL) which provides iterators, distributions, expressions and functions to generate Relational Databases (RDBs). Similarly, Hoag and Thompson [78] propose a Synthetic Data Description Language (SDDL) and use parallelism to generate large RDBs in a short amount of time. Their XML-based language let users define database elements, constraints and iterations. Jeske et al. [83] utilize semantic graphs to express data relationships in their Information Discovery and Analysis Systems Data Set Generator (IDSG). This way, different kinds of conditional distributions can be defined. Rabl and Poess [119] present a Parallel Data Generation Framework (PDGF)

1 https://data.gov/
2 https://www.govdata.de/

which is able to generate datasets to benchmark cloud computing. Its XML syntax let users specify an RDB schema, generators and cluster settings for parallelization. A follow-up work [120] also considers complex RDBs with poor or odd schema design. They allow to generate intra-row, intra-table and inter-table dependencies which aligns to the messy data challenges discussed in Section 2.1.

The mentioned generators are designed to produce RDBs, however enterprise data also consists of office documents and PIM structures. For example, a company specialized in Purchase-to-Pay (P2P) processes typically work with invoice documents, thus for such a special domain dedicated generators are required. To acquire such a corpus for analysis and recognition, Blanchard et al. [24] propose an invoice generator. The authors define an invoice model schema which is used to generate invoice meta data. Different layout strategies arrange elements in the final documents which are outputted as PDFs and images. In collaboration with Schulze et al. [155] a similar generator called Purchase-To-Pay Dataset Generator (ptpDG) is proposed which also considers the process context of invoices with a Multi-Agent System (MAS) simulation approach. Using the Purchase-to-Pay Ontology (P2P-O) [154], a ground truth KG of invoices is generated first which is used to produce invoice XML documents. During document generation, noise patterns are applied to introduce a certain degree of messiness found in real-world documents.

The generation of KGs is considered by communities which would like to benchmark SPARQL [181] queries. For a comprehensive survey on graph generators the reader may consult the work by Bonifati et al. [26]. In literature the following notable generators have been proposed: Lehigh University Benchmark (LUBM) [69], Berlin SPARQL Benchmark (BSBM) [21] and The SP^{2}Bench SPARQL Performance Benchmark (SP2B) [132]. Although, they generate arbitrary large KGs, they miss a mechanism to produce messy data from it to receive a useful dataset for evaluating KG construction. For benchmarking virtual KG access, GTFS-Madrid-Bench [34] also considers the generation of data in various formats (CSV, JSON, SQL and XML), however in a consistent and clean way to not compromise the benchmark.

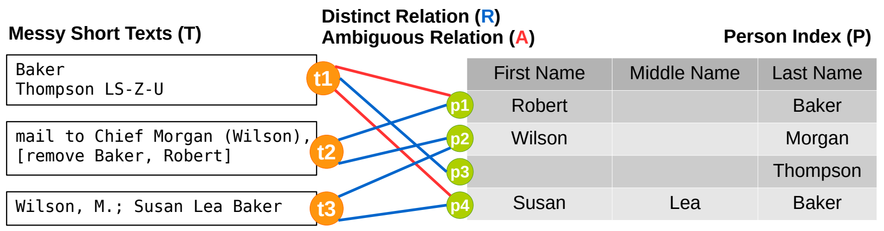
To conclude, the discussed generation approaches still do not consider or reach the degree of messiness which has been shown in Section 2.1. For evaluating KG construction, they also lack in sufficient annotations of ground truth, for example in form of provenance information and semantic meaning. Generators are usually designed to produce datasets for benchmarking the performance of various methods. This is the reason why generated data is diverse, consistent and comprehensive, but not intentionally messy.

In comparison to RDBs, spreadsheets have less constraints and offer more freedom to introduce messiness in data. Resulting challenges for KG construction have been discussed in Section 2.1. Like in the example in Table 2, similar data could be automatically produced to obtain messy datasets. For example, once entities are mentioned in cells multiple surface forms can be purposely used. This particular idea is discussed in the next section where a proposed generation approach focus on person mentions.

4.3 PERSON INDEX GENERATOR

Persons can be mentioned in unstructured (sometimes short) texts (T) in various ways, since their first names (fn), middle names (mn) and last names (ln) can be arbitrarily selected, ordered and abbreviated. In cases where persons share some of their names with others, ambiguities can occur, especially in short texts [80]. During Knowledge Graph (KG) construction from such data, one reasonable task is the building of a person index (P) which distinctly catalogs persons by their names as resources in the graph. This is particularly useful when no such knowledge base about persons exists in advance. However, if texts are rather messy and person names do not occur in a certain pattern, the recognition becomes a challenging task. Formally defined, given some short texts $t_i \in T$, the task is to extract persons $p_j = (fn, mn, ln) \in P$ from them with their full length names as far as possible. An additional relation $(t_i, p_j) \in R \subseteq T \times P$ associates both sides, however in case of ambiguities, a separate relation $(t_i, P_A) \in A$ lists ambiguous mentions of persons in P_A instead.

An example is presented in Figure 25 where a person index (right) with four individuals is derived from three messy short texts (left). Five distinct relations exist (blue lines), namely $R = \{(t_1, p_3), (t_2, p_2), (t_2, p_1), (t_3, p_2), (t_3, p_4)\}$. The ambiguous relation (red lines) is $A = \{(t_1, \{p_1, p_4\})\}$, since "Baker" cannot be clearly assigned to one person.



	First Name	Middle Name	Last Name
p1	Robert		Baker
p2	Wilson		Morgan
p3			Thompson
p4	Susan	Lea	Baker

Figure 25: Exemplary output of Person Index Generator: messy short texts (left) are associated with indexed persons (right) via distinct and ambiguous relations (middle).

To evaluate KG construction for this particular challenge, a ground truth generator is proposed which is able to generate the relations P, T, R and A in form of CSV files. As input, the generator expects a list of first names and last names, a seed for randomness and quantity settings. First, the person index (P) is formed with the given name lists. To introduce ambiguity here, the size and number of groups where persons share a name can be configured. These individuals are generated first, while the persons' names will not be used again to avoid unintentional ambiguities later. After that, fully distinct persons are added to the index without reusing any name. Some of them receive a middle name (configurable) by giving them another first name.

From the finalized person index, all messy short text snippets are written (T). At random, each text snippet mentions a single person or a group of them. To produce especially messy texts, various naming variations are randomly picked for the persons. A list of naming patterns is presented by Table 16. Patterns are a combination of first names (fn), middle names (mn) and last names (ln). To add noise, the functions *department*(), *note*() and *role*() return abbreviated department names, short notes and role descriptions, respectively. The function *rnd*(n) produces a random string of length n to generate pseudo emails. Given a name, *lc* turns it to lower case, while *letter* returns its first letter. A new line is indicated by the ↵ symbol. In the examples the full name "John Fitzgerald Kennedy" is used to demonstrate the pattern. To ensure that a middle name is mentioned at least once, the 11th pattern is always picked first. The check marks in Table 16 help to keep track if names are fully mentioned at least once. Multiple mentions are delimited with randomly chosen separators or are surrounded with quotes and brackets to avoid that they can be trivially separated. Once a text snippet is produced, the relation R is filled whenever persons are unambiguously mentioned. In case of ambiguity, the relation A is used instead.

Limitations

The generator's usage is limited by some design decisions. It is solely made for Western world names with exactly one optional middle name which restricts its usage to some languages and use cases. Besides persons, other entity types could be similarly generated, but this is not supported by the generator. Regarding the identified challenges with messy data (Section 2.1), the generator covers *Multiple Surface Forms* and *Multiple Entities in a Cell*. Since noisy departments, notes and roles are also added to the text, it also includes the challenge of *Free Comments*. However, not all possibilities are exhausted to produce messy data, since the remaining challenges are not considered. To better understand what other circumstances make spreadsheets messy, the next section covers a dedicated pattern language.

Table 16: Naming patterns in Person Index Generator with examples (has been published in [152]). A check mark (✓) indicates that a certain name is fully mentioned. Abbreviations: First Name (FN), Middle Name (MN) and Last Name (LN).

No.	Pattern	Example	FN	MN	LN
1	fn	John	✓		
2	ln	Kennedy			✓
3	fn ln	John Kennedy	✓		✓
4	ln fn	Kennedy John	✓		✓
5	ln, fn	Kennedy, John	✓		✓
6	ln, *letter*(fn).	Kennedy, J.			✓
7	ln *department*()	Kennedy US-Z-G			✓
8	*department*()↵ln fn	US-Z-G↵Kennedy John	✓		✓
9	ln fn <*lc*(ln)@*rnd*(5).*rnd*(2)>	Kennedy John <kennedy@xraok.nc>	✓		✓
10	*note*() *role*() ln fn	new Admin Kennedy John	✓		✓
11	fn mn ln	John Fitzgerald Kennedy	✓	✓	✓
12	fn *letter*(mn). ln	John F. Kennedy	✓		✓
13	*letter*(fn). *letter*(mn). ln	J. F. Kennedy			✓
14	ln, *letter*(fn). *letter*(mn).	Kennedy, J. F.			✓
Sum	-	-	9	1	13

4.4 A PATTERN LANGUAGE FOR SPREADSHEETS

Particular behaviors can be observed when domain experts work with their data assets: once they enter or modify data, they tend to carry out operations in an accustomed way. However, when they do not follow a data management strategy, their familiar workarounds and habits can make data messy. These behaviors on data can be interpreted as *patterns*, because people recurrently show particular ways how they work with datasets. This approach is inspired by Alexander's pattern language in the architectural domain [6], however instead of describing how houses can be designed, the proposed pattern language for spreadsheets explain for a given circumstance (i.e. situation, issue) a possible modeling decision of users in workbooks. Because decisions tend to make data messier, they can be seen as a form of anti-patterns (i.e. unfavorable for data quality). Such (anti-)patterns can be collected and cataloged by interviewing users and examining their files in practice. After investing data in industry projects, twelve reoccurring patterns could be identified which are listed in Table 17. Each pattern has a unique name and is assigned to a category whether it describes how . . .

- information is modeled (Modeling),
- formatting conveys meaning (Formatting),
- additional information is added (Extension) or
- information is positioned (Layout).

An explanation of a circumstance is followed by a description how information is modeled in spreadsheets. Check marks in the last

Table 17: A pattern language for spreadsheets which describes circumstances and resulting modeling decisions. Entities can be things (resources) or values (literals). A check mark indicates a pattern's implementation in Data Sprout (see Section 4.5). Parts have been published in [144].

Category	Pattern	Circumstance	Modeling Decision	Data Sprout
Modeling	Numeric Information as Text	There is numeric information.	It can also be represented as text.	✓
Modeling	Acronyms or Symbols	There is a reference to an entity.	To save time a rather short acronym or symbol string is used to refer to it.	✓
Modeling	Multiple Surface Forms	Entities can be mentioned in various ways.	Different cells contain distinct surface forms of equal entities.	✓
Formatting	Property Value as Color	Some entities have different properties.	Different colors encode properties which are chosen to color entities' cell ranges.	✓
Formatting	Partial Formatting Indicates Relations	Multiple entities in one cell have different relationships with another entity.	Partial formatting is used to indicate their relations.	✓
Formatting	Outdated is Formatted	Information is not valid anymore, but must not be removed completely.	Outdated information is formatted.	✓
Extension	Multiple Entities in one Cell	Multiple entities have to be referred to.	Entities are listed in the same cell.	✓
Extension	Unordered Entities	Multiple entities have to be referred to.	Entities are listed in the same cell, but in different orders.	
Extension	Intra-Cell Additional Information	Additional information is related to information that is already recorded in a cell.	The additional information is recorded in the same cell.	✓
Extension	Distant Additional Information	Additional information is related to information that is already recorded in a cell.	Via a reference, the additional information is recorded in a distant cell.	
Extension	Cell Comment Explains Content	A cell's content is vague.	A cell comment is attached to the cell to explain its content in more detail.	
Layout	Multiple Types in a Table	Some entities of different types share some properties.	Entities are recorded in the same table.	✓

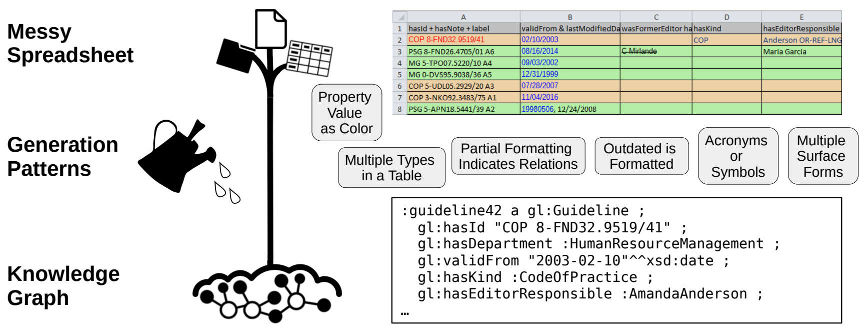
column mean that patterns have been implemented in the Data Sprout generator which is discussed in the next section. An online version of Table 17 has been published on a website with additional information, examples and images [134].

The twelve patterns were derived from the dataset which is mentioned in Table 3. Two authors were interviewed for four hours in total to intensively discuss around 90 columns from their spreadsheets. Insights in the way they modeled information in their sheets lead to the first patterns in Table 17. To verify that they also occur in other datasets, the U.S. Government's open data platform[3] was consulted and 3692 spreadsheets were downloaded. After 200 manually inspected sheets, all patterns could be found, yet they occurred very infrequently.

A dataset generator which utilizes the proposed pattern language is presented in the next section.

4.5 DATA SPROUT SPREADSHEET GENERATOR

Using the pattern language from the previous section, a dataset generator called Data Sprout is presented. The patterns are utilized to imitate the completion of spreadsheets. If this can be executed authentic enough, it would appear as if some persons made the files themselves. Patterns have also the advantage that they can be arbitrarily mixed to produce datasets which have not been seen yet. Such combinations can introduce ambiguities and make possible pattern recognition solutions non-trivial. Moreover, users can decide which patterns they would like to apply to best mimic their confidential datasets. Since the patterns imitate real-world datasets, evaluation results on synthetic ones are expected to be similar.



	A	B	C	D	E
1	hasId + hasNote + label	validFrom & lastModifiedDa	wasFormerEditor ha	hasKind	hasEditorResponsible
2	COP 8-FND32.9519/41	02/10/2003		COP	Anderson OR-REF-LNG
3	PSG 8-FND26.4705/01 A6	08/16/2014	~~C Mirlande~~		Maria Garcia
4	MG 5-TPO07.5220/10 A4	09/03/2002			
5	MG 0-DVS95.9038/36 A5	12/31/1999			
6	COP 5-UDL05.2929/20 A3	07/28/2007			
7	COP 3-NKO92.3483/75 A1	11/04/2016			
8	PSG 5-APN18.5441/39 A2	19980506, 12/24/2008			

```
:guideline42 a gl:Guideline ;
  gl:hasId "COP 8-FND32.9519/41" ;
  gl:hasDepartment :HumanResourceManagement ;
  gl:validFrom "2003-02-10"^^xsd:date ;
  gl:hasKind :CodeOfPractice ;
  gl:hasEditorResponsible :AmandaAnderson ;
…
```

Figure 26: Illustration of Data Sprout with an example (from bottom to top): based on a given knowledge graph various generation patterns are randomly applied to produce a messy spreadsheet.

An illustrative example of the approach is shown in Figure 26 which is read from bottom to top. Data Sprout expects two inputs: an existing

3 https://www.data.gov/

Knowledge Graph (KG) and generation patterns as in Table 17. At the bottom, an excerpt of an RDF graph shows meta data about a fictional guideline document using a suitable ontology [135] (prefixed `gl`). Gray boxes in the middle indicate used patterns located near columns they were applied to. A generated messy spreadsheet is depicted at the top. The information shown in the excerpt is used to complete the sheet's second line. How patterns are implemented in Data Sprout is discussed next. The sheet in Figure 26 serves as an example for some of the patterns.

Layout. The sheets' layout are derived from the KG's terminology while its content is filled with the KG's assertions. Similar to RDF Spreadsheet Editor (Section 3.3.4), the default layout has a *class per sheet*, *entity per row*, *property per column* and *object per cell* analogy. This behavior can be changed with the *Multiple Types in a Table* pattern: it makes sure that two classes are associated with one sheet, once they share a certain amount of properties (i.e. columns). As a result, instances of different classes can be found in the same table. Since guidelines and attachments have several properties in common, they are mixed in the exemplary table.

Extension. Similarly, *Intra-Cell Additional Information* does also change the default table layout, since one column gets associated with two or three properties, which happens in Column A and Column B, for example. Consequently, multiple RDF nodes from different properties are present in one cell, which automatically requires the *Multiple Entities in one Cell* pattern, as shown in Cell B8.

Formatting. To recognize which property corresponds to a value, the *Partial Formatting Indicates Relations* pattern consistently styles or colors text. For instance, in Column B all "valid from" dates are colored blue, while last modified dates stay black (Cell B8). Moreover, a cell's foreground and background color can encode property values. To apply the *Property Value as Color* pattern, Data Sprout finds suitable property-value pairs and associates them with colors. As an example, depending on the documents' associated departments, rows are differently background colored: *Human Resource Management* department means orange, while *Research and Development* department is green. Since the background color alone provides insight into a document's associated department, the column that explicitly lists them is removed from the table. Regarding the *Outdated is Formatted* pattern, Data Sprout expects a list of properties that refers to outdated information. Once such a property's value is mentioned in a cell, it is automatically struck out. An example for this can be seen in Cell C3: since Mirlande Johnson is a *former* editor, the person's name is crossed out in the cell (the preceding 'C' indicates a department).

Modeling. Due to the *Numeric Information as Text* pattern, some literal values will be stored using their string representations. This way, numbers, dates and Boolean values occur as texts in potentially different formats. Dates can be seen in Column B where the two formats `MM/dd/yyyy` and `yyyyMMdd` are used. Besides literals, resources are identified with their labels by default. However, the *Multiple Surface Forms* pattern will consider a random mix of several properties of a resource to mention it. In the example, persons are mentioned with a combination of first name, last name and department (Column E). The *Acronyms or Symbols* pattern tries to use special acronym labels (if available) and represents Boolean values as symbols (e.g. *true* is represented as '✓'). In Cell D2, the acronym "COP" is intentionally chosen, although the resource's label is "Code of Practice".

Provenance. During generation, Data Sprout keeps track of the statements used to complete cells. Therefore, this provenance information can tell evaluators for each spreadsheet cell what RDF statements are expected. This ground truth information is essential to measure KG construction performance. For instance, ground truth data has been used to evaluate Spread2RML's prediction performance (Section 2.5).

Generated Datasets. To test Data Sprout's capabilities and to provide some generated spreadsheets, it was applied on RDF datasets acquired from the following benchmarks: Lehigh University Benchmark (LUBM) [69], Berlin SPARQL Benchmark (BSBM) [21] and The SP^{2}Bench SPARQL Performance Benchmark (SP2B) [132]. Additionally, a KG generation procedure is provided which produces assertions using terminology from the guideline ontology [135]. From these KG sources, spreadsheets have been generated with one activated pattern each, with all of them (very messy) and without any of them (clean).

Limitations

The produced datasets from Data Sprout also reveals some of its limitations. Since randomness is applied for each cell and particular decision inside a pattern, variety gets very high. Although this leads to very messy spreadsheets, datasets lose authenticity very much, since people usually do not change their behaviors this frequently. The distribution of random choices in patterns is currently not configurable.

Generated synthetic datasets with ground truth are shareable and enable external evaluations for other researchers. In contrast to that, in the next section a dedicated tool is covered which collects users' personal information meant for internal evaluations only.

4.6 PERSONAL INFORMATION MANAGEMENT CRAWLER

Data about Personal Information Management (PIM) is usually distributed in isolated data stores on a user's desktop. To gather various information in a single place for later analyses, a crawler for data about PIM has been designed in collaboration with Jilek [86]. The idea has been realized in a tool called PIM Crawler. It is able to collect PIM data in form of files, calendars, emails and bookmarks to one Relational Database (RDB). The compilation of one dataset enables approaches to easily retrieve and analyze personal data, like for example Concept Miner in Section 3.3.1. A bachelor thesis about context mining [74] and a related student project [36] successfully utilized the tool.

The database's schema in Figure 27 shows what meta data can be collected by PIM Crawler. In the following the crawling of diverse data sources is discussed.

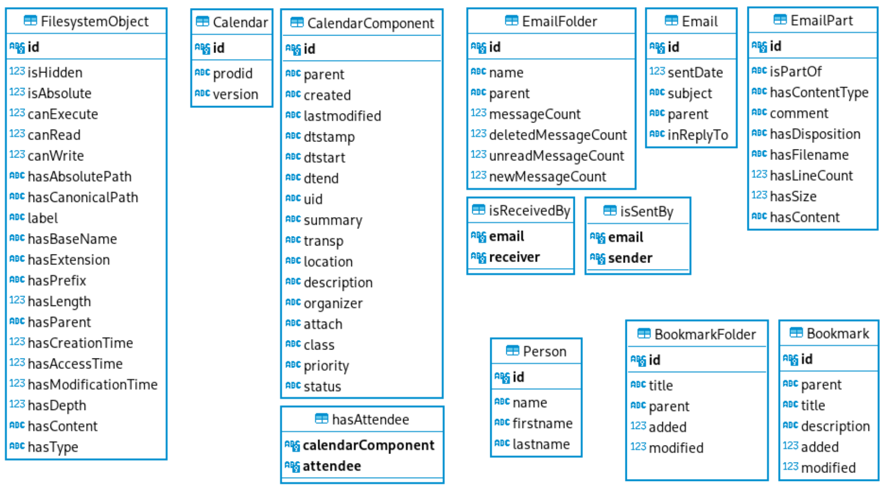


Figure 27: Personal information collected by PIM Crawler in form of a database schema diagram.

Files. Starting from a given folder, PIM Crawler recursively traverses a local file tree and captures various meta data of visited files and folders in the *FilesystemObject* table. Besides visibility and access permissions, a file's path, name, extension and size is stored. Recorded timestamps provide points in time when files are created, accessed and modified. The hierarchical structure can be reproduced with references to parent folders and depths in the tree.

Calendars. Common calendar applications allow to serialize their data in files as defined in the Internet Calendaring and Scheduling Core Object Specification[4] (iCalendar). PIM Crawler utilizes the iCal4j Library[5] to read and store relevant meta data in the *Calendar* and *CalendarComponent* table. The latter records calendar entries' textual information (*description*, *summary*), time information (*dtstart*, *dtend*), spatial information (*location*), and personal related information (*organizer*, *hasAttendee*). The separate *hasAttendee* table associates an entry to one or many persons in the *Person* table using their email addresses.

4 https://datatracker.ietf.org/doc/html/rfc2445
5 https://ical4j.github.io/

Emails. The JavaMail API[6] allows PIM Crawler to obtain meta data about emails. While the Internet Message Access Protocol[7] (IMAP) is typically used to access emails remotely, clients also tend to store (or export) their data in the mbox[8] (Mailbox) format. Independent which source is chosen, the API allows to record hierarchically structured email folders with their message statistics in the *EmailFolder* table. Containing emails are stored with their usual subject and time information (*Email* table). The actual (possibly hierarchically structured) content is captured by the *EmailPart* table including meta data about its name, type and size. Many-to-many relationships between email and receiver/sender (*Person* table) is modeled with the *isReceivedBy*/*isSentBy* relations.

Bookmarks. Users typically store bookmarks in Web browsers to later retrieve certain websites on the World Wide Web. PIM Crawler is able to access bookmark data from the Mozilla Firefox[9] browser which is stored in a local database called `places.sqlite`[10]. This way, meta data about folders and entries are captured in corresponding tables with title, description and time information.

4.7 CONCLUSION

This chapter shed light on the open dataset problem which arises when evaluations are planned with confidential datasets from industry including Personal Information Management (PIM) data. Since literature does not already provide suitable datasets, a possible solution to generate artificial ones was proposed. An initial step was the generation of an index of persons together with short messy texts mentioning them. Because this approach does not cover all identified challenges with messy data, a pattern language for spreadsheets was introduced. The catalog of (anti-)patterns explain how users enter data in spreadsheets in particular ways on given circumstances. With the help of these descriptions, a dataset generator was implemented called Data Sprout. By reproducing the patterns to generate spreadsheets, the approach tries to imitate real-world data as authentic as possible. Since RDF graphs are the sources for completing spreadsheet cells, its provenance information serves as ground truth in evaluations of KG construction approaches. For internal investigations, a crawler for PIM data was presented which is able to store parts of a user's personal information sphere in a single database.

The conducted investigations lead to an answer for the third research question (Section 1.2): **How can datasets be generated for evaluation purpose?**

6 https://javaee.github.io/javamail/
7 https://datatracker.ietf.org/doc/html/rfc9051
8 https://datatracker.ietf.org/doc/html/rfc4155
9 https://www.mozilla.org/en-US/firefox/
10 https://github.com/mozilla/firefox-data-store-docs#placessqlite

Generated synthetic datasets have to be close to reality in order to acquire very similar evaluation measures. To capture characteristics of datasets in practice, recognized patterns in data turned out to be a useful instrument. Person Index Generator uses naming patterns to consider many realistic person name variations. A pattern language for spreadsheets record behaviors when domain experts complete their sheets. Once patterns are reproduced by a generator like Data Sprout, real-world data is mimicked in an authentic way. The flexibility of patterns allows the creation of more complex datasets providing greater challenges. Applied randomness and mixtures of patterns can introduce such a high ambiguity in data that trivial inverse methods of its generation might not be sufficient solutions. Still, the collection of real data should always be considered to be able to recognize true challenges and discover patterns in practice.

Regarding evaluation, data generation from KG facts and the record of provenance information proved to be important design decisions in Data Sprout. Evaluations can then be performed by inverting the dataflow and declaring the generated data to be the challenge. This way, provenance information and the KG become the ground truth necessary to perform evaluations of KG construction approaches (like for Spread2RML in Section 2.5).

5 KNOWLEDGE GRAPH APPLICATIONS

This chapter covers contributions related to this PhD thesis in diverse domains. Parts of it have been published in [76, 89, 90, 97, 148].

5.1 PERSONAL KNOWLEDGE ASSISTANTS

In the Semantic Desktop [129] (see Section 1.3.3) users can be supported with assistants which are aware of users' personal knowledge. The research project SensAI[1] pursues several goals to enable this technology in daily knowledge work. Since KGs are a key technology to represent a user's mental model (in form of PIMOs) [130] and a company's corporate memory[2], one major goal is the construction of them from personal and enterprise data. An important contribution towards archiving this goal has been the research already mentioned in this PhD thesis (see Chapter 2 and Chapter 3). The project's second goal is the collection of evidences in user activities with software sensors to model contexts which are designed to be self-organizing. Important foundations for reaching this goal can be found in the research by Jilek [86]. Based on that, a third goal is the application of personal knowledge services which are embedded in work environments [49].

Several related topics have been pursued in collaboration with peers to contribute to these goals. They are briefly presented in the following sections.

5.1.1 *Text Analysis*

Enterprise and personal data usually contains unstructured texts in form of office documents, texts in emails, file names, titles and descriptions of calendar entries, bookmarks, etc. Analysis of these texts give important indications for users' PIMOs, as shown in Concept Miner (Section 3.3.1) and KECS (Section 3.3.2). Named Entity Recognition (NER) is a useful approach to better understand the content of texts. Personal knowledge assistants could apply NER to texts from observed user activities to instantaneously provide support measures. Together with Jilek et al. [89] a NER approach is proposed which considers such real-time applications. Additionally, it takes language into account that are highly inflectional such as German, Spanish, etc. A conceptional overview of the approach is given in Figure 28 where arbitrary text (left) is annotated with recognized named entities (right). To

1 `https://comem.ai/SensAI`
2 `https://comem.ai/`

do so, the method combines *trie*-based string matching [5], finite state cascades [2] and exhaustive inflection listing [131]. KGs, like PIMOs or the DBpedia [10], are used to exploit background information about the entities to be recognized. Additional language resources provide access to information about word types and flections. Compared to available high-speed methods, the proposed approach is still fast and outperforms them once terms vary slightly, for example because of inflections. With this NER method, personal knowledge assistants are able to instantly find named entities in user activity streams and act upon them.

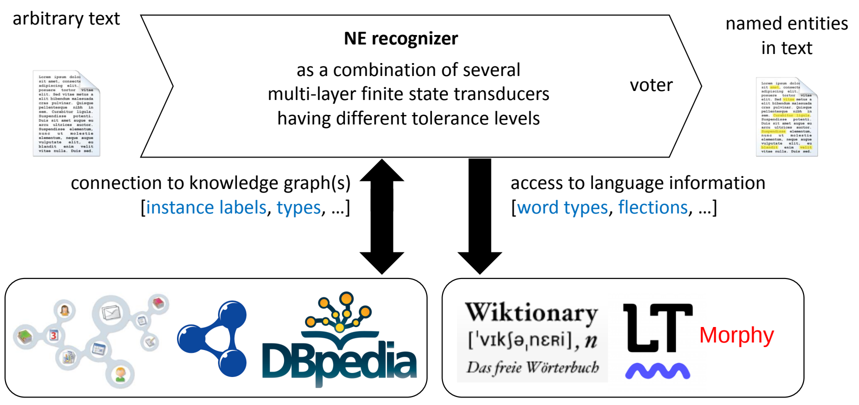

arbitrary text
NE recognizer
as a combination of several multi-layer finite state transducers having different tolerance levels
voter
named entities in text
connection to knowledge graph(s) [instance labels, types, ...]
access to language information [word types, flections, ...]
DBpedia
Wiktionary ['vɪkʃəˌnɛʀi], n Das freie Wörterbuch
LT Morphy

Figure 28: Conceptional overview of inflection-tolerant ontology-based NER for real-time applications (has been published in [89]).

An implementation of this NER approach has been integrated in a component called **Texana**[3] (an abbreviation of Text Analysis). To demonstrate its capabilities, it is loaded with the NECKAr dataset [65] which contains named entities in form of persons, locations, and organizations extracted from Wikidata [165]. Besides NER, the component also provides functionality to tokenize, stem, lemmatize, decompound words and detect a text's language. An additional server implementation with a RESTful API allows other researchers to use the service remotely.

Another similar library has been implemented independently which is called **String Analyzer**. It is designed to analyze and process especially short and messy character strings (text snippets) in various ways. For an overview of a set of strings, it is able to calculate several statistics on them like average length, distinctness and character distribution. Strings can also be clustered in groups to find similar looking ones. For machine learning tasks an extraction of string-features can be applied to acquire training vectors. Similar to the previous NER approach [89], *trie* structures can be formed from strings, but with the motivation to analyze their prefixes and postfixes, for example to find longest common ones. Regular expressions are used to search,

3 `https://www.dfki.uni-kl.de/kwt/texana`

filter and annotate texts. Conversely, such expressions can also be inferred from a list of strings, similar to the method from Bartoli et al. [13]. Splitting procedures are provided to separate strings by n-grams or tokens. Common term extraction algorithms are implemented to extract keywords from texts (for a list see [189]). The library's features have been used in a bachelor thesis about a search engine in forgetful information systems [110] and on several occasions in KG construction tasks.

5.1.2 *Context Spaces*

As already stated, PIMOs try to represent users' mental models and interlink their information items [130]. In the PhD thesis of Jilek [86], this concept is extended with explicit contexts called Context Spaces (cSpaces). They provide users additional contextual information and let them associate items in contexts while working in them. A visualization of this approach is given at the bottom of Figure 29. By exploiting

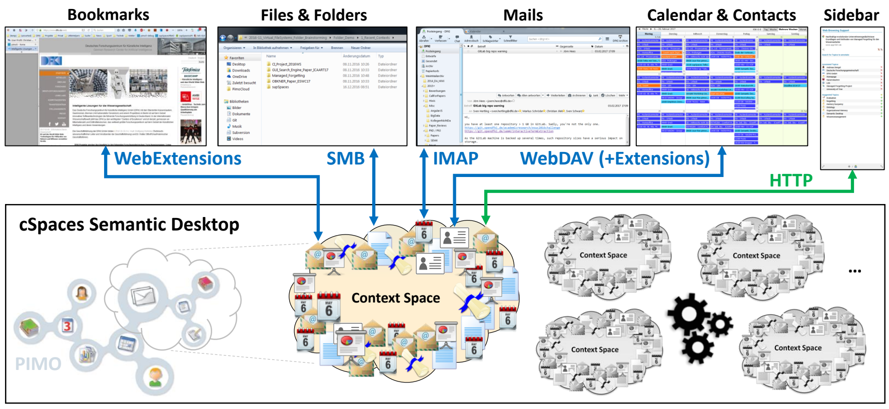


Figure 29: Context Spaces (cSpaces) overview with transparent integrations in usual office applications and an additional sidebar (has been published in [86]).

standard protocols, cSpaces are transparently integrated in usual applications to let users work in the same context regardless of technical barriers. An additional sidebar is able to provide advanced features no common application offers out of the box. Once users interact with explicit contexts, novel support measures for self-reorganization are possible such as condensation, summarization, temporal hiding or permanent reorganization. In [90] an early technical prototype of this concept is presented.

Mentioned support measures are based on Memory Buoyancy, a variant of Information Value Assessment and a cornerstone of Managed Forgetting [88]. Envisioned and developed with colleagues of cognitive science, it is inspired by human memory and cognition. In its latest version it also considers context-sensitivity [87].

5.1.3 *GUI Assistance*

Besides support for usual information management tasks on the Semantic Desktop, operations on its Graphical User Interface (GUI) can also be supported by the system. In a project called SuGraBo[4] such assistance for the GUI on desktops was investigated. Related to this project and in collaboration with Hertling et al. [76] a search engine for the GUI for the Windows[5] operating system is proposed. By utilizing the accessibility interface [106] which is typically used by screen readers, the hierarchical structure and meta data of GUI elements can be obtained. From this data, a directed graph of GUI elements is formed which expresses what elements make other elements visible. From these ingredients, a GUI search engine can be created which is demonstrated in Figure 30. On the left, an information retrieval

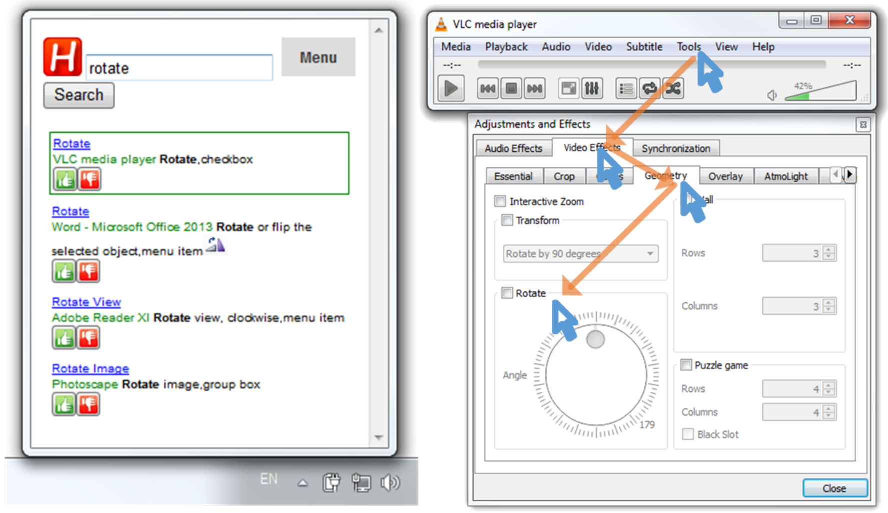

rotate
Search
Menu
Rotate
VLC media player Rotate,checkbox
Rotate
Word - Microsoft Office 2013 Rotate or flip the selected object,menu item
Rotate View
Adobe Reader XI Rotate view, clockwise,menu item
Rotate Image
Photoscape Rotate image,group box
VLC media player
Media Playback Audio Video Subtitle Tools View Help
Adjustments and Effects
Audio Effects Video Effects Synchronization
Essential Crop Colors Geometry Overlay AtmoLight
Interactive Zoom
Transform
Rotate by 90 degrees
Rotate
Angle
Wall
Rows
Columns
Puzzle game
Black Slot
Close

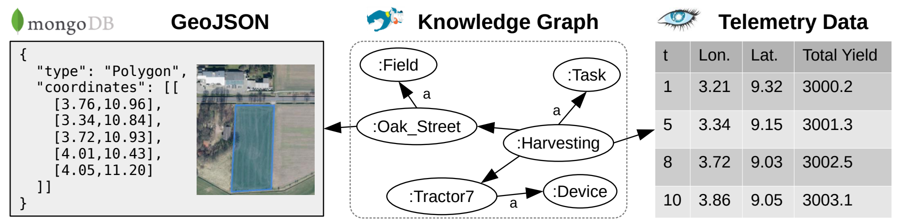
Figure 30: Demonstration of a GUI search engine where the query "rotate" leads to a list of possible elements (left). By selecting the first one, mouse operations are automatically preformed to make the element visible (right). This figure has been published in [76].

system interprets user queries and lists matching GUI elements by using their meta data. Once an element is selected, the graph is used to automatically traverse the interface until the desired element is visible. This way, textual queries retrieve GUI elements of applications. An evaluation showed that in particular cases the search engine found elements even faster than users or helped those who gave up manual search.

4 https://comem.ai/SuGraBo
5 https://www.microsoft.com/en-US/windows

5.2 SMART AGRICULTURE

In the agricultural domain typical technical challenges are the data exchange across manufacturers and interpretation of diverse data formats [54]. For data integration of different agricultural sources the application of KGs can be beneficial in this field. This was also a goal in the research project Smarte Daten Smarte Dienste[6] (abbrev.: SDSD, Engl.: Smart Data Smart Services). In context of the project and in collaboration with Klose et al. [97] preparation of data in agriculture is investigated. To archive this, an open collaborative platform called Wikinormia is proposed to describe data formats with Semantic Web standards. Together with parser implementations, raw data is transformed to RDF statements and integrated into a KG. Besides linked data, two more data models are considered for agricultural data in a polyglot store [127] as depicted in Figure 31. Field boundaries and



```
{
  "type": "Polygon",
  "coordinates": [[
    [3.76,10.96],
    [3.34,10.84],
    [3.72,10.93],
    [4.01,10.43],
    [4.05,11.20]
  ]]
}
```

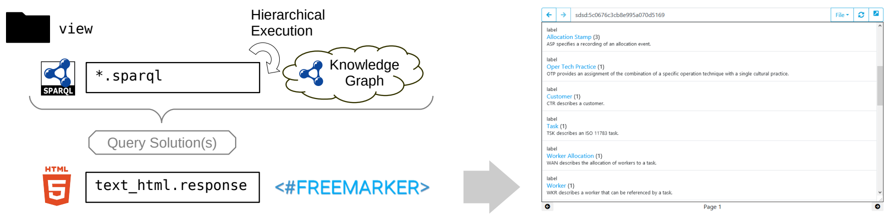
t	Lon.	Lat.	Total Yield
1	3.21	9.32	3000.2
5	3.34	9.15	3001.3
8	3.72	9.03	3002.5
10	3.86	9.05	3003.1

Figure 31: Polyglot storage in SDSD to manage spatial data (GeoJSON), linked data (KG) and time series (telemetry data).

lanes are serialized in the GeoJSON[7] format and stored with MongoDB[8] using spatial indexing. Sensor data from machines (like total yield) with time and position information (longitude and latitude) are recorded in the Apache Cassandra[9] database due to the data its quantity. Entities such as fields, tasks and machines are semantically described and interlinked in a KG using the Stardog Triplestore[10]. The graph additionally links associated geodata and time series to their corresponding instances.

To visualize the KG's content for farmers, an approach is proposed called **Smart Data Browser**. A conceptional view with an example of it is provided by Figure 32. Views on the KG are defined by two parts: SPARQL queries [181] for knowledge retrieval and response templates using mime types (usually HTML). Both utilize the template engine Apache FreeMarker[11] to dynamically generate their definitions. For a single view more than one SPARQL query can be hierarchically executed to collect multiple query solutions. In a final rendering step

6 https://sdsd-projekt.de/
7 https://datatracker.ietf.org/doc/html/rfc7946
8 https://www.mongodb.com/
9 https://cassandra.apache.org/
10 https://www.stardog.com/
11 https://freemarker.apache.org/



sdsd:5c0676c3cb8e995a070d5169

label
Allocation Stamp (3)
ASP specifies a recording of an allocation event.

label
Oper Tech Practice (1)
OTP provides an assignment of the combination of a specific operation technique with a single cultural practice.

label
Customer (1)
CTR describes a customer.

label
Task (1)
TSK describes an ISO 11783 task.

label
Worker Allocation (1)
WAN describes the allocation of workers to a task.

label
Worker (1)
WKR describes a worker that can be referenced by a task.

Figure 32: Conceptual presentation of Smart Data Browser which utilizes SPARQL and response templates together with the template language FreeMarker to render views on KGs.

a view is presented with all gathered information, for example as a Web page. Typical browsing is enabled by providing parameterized links to further views.

5.3 DATA SCIENCE

In the data science field, formalized knowledge has proven to be useful, in particular when data mining tasks and domain knowledge come together [53]. At the beginning of a data mining project this usually happens when data scientists and data providers have an active exchange of information about data sources. During this *Data Understanding* phase they collect, describe and explore datasets as specified by the Cross-Industry Standard Process for Data Mining (CRISP-DM) [169]. Together with project partners this phase was also conducted in the research project Big Data Production Optimization in Smart Ecosystems[12] (PRO-OPT) which is described in detail in the following. Two data sources were selected for the analysis of manufacturing processes: a Relational Database (RDB) with 84 relevant tables and a dataset of 2186 log files. Initially, the data provider explained the relationships of 25 tables with a diagram. Additionally, about 30 emails were exchanged to clarify technical terms and abbreviations, five SQL queries and their reformulations, several data mining use cases, comprehension questions, feedback to analysis results and additional information in form of four Excel sheets and four PDF documents. In order to cope with the diverse information, the data analysts compiled a document which covered domain knowledge, relevant SQL tables, meanings of technical codes and parsing rules for complex fields found in the database. A glossary listed 30 concepts which are relevant for understanding the domain and its data.

Several challenges could be identified during this venture. Due to a first normal form violation in the database additional parsing of cells was required. The database's schema was incomprehensibly labeled with various naming conventions, unfamiliar abbreviations, technical terms and concatenated lower case words without delimiters. As a

12 http://www.pro-opt.org/

result, between data scientist and data provider there was an ongoing dialog via emails, telephone calls and face-to-face meetings. Each time new unstructured information were exchanged which resulted in high recording and documentation efforts to summarize findings. Unfortunately, provided explanatory documents were separated, heterogeneous, unstructured and were loosely related to the entities they describe.

A potential solution could be the maintenance of a data dictionary [163] which is able to serve as a central repository of information about data. However, to address identified challenges, a more sophisticated solution is required which is able to semantically store and semi-automatically extract insights about data. To archive this, an early concept of a ***semantic* data dictionary** is proposed (see Figure 33). It consists of three layers: in the bottom layer, various data sources can be considered for the semantic data dictionary (an example is given with a database). The middle layer explicitly models how data elements and values relate to each other, while provenance information keeps track from which source the elements come from. To introduce semantic, the top layer let users define a conceptualization with terminologies and assertions. This way, data elements (e.g. tables) can be associated with concepts or, in a more fine-grained way, text spans of technical labels can be annotated with concepts (e.g. `vblk` stands for "Verification Block"). During PRO-OPT and in a follow-up project contributions to this concept were made.

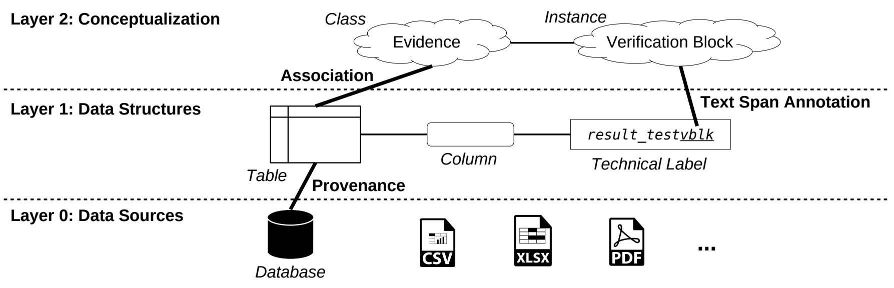
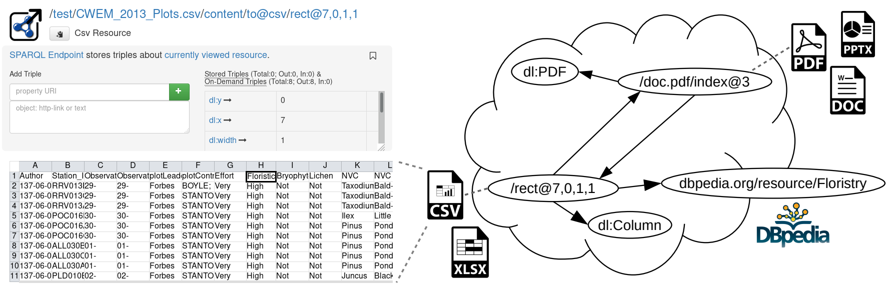


Figure 33: Conceptual view of a semantic data dictionary with a database example.

In a feasibility study with an industry partner the application of KGs for data lakes was investigated. Especially in data lakes, data scientists can lose overview of relevant data due to its size and complexity [109]. As a solution, a Knowledge Graph (KG) is able to interlink data and enrich it with meaningful concepts. To construct such a graph, several analyses were performed on database schema and content. In order to obtain a summarizing and understandable format for column values, inference of regular expressions (regex) was applied. For example, an inferred regex pattern like "`\d+:\d+:\d+`" indicates that the column solely consists of timestamp values. The proposed inference could produce more suitable expressions compared to a similar approach from Bartoli et al. [13]. String matching [5] was applied to table and column names to annotate them with named entities. This way technical terms and abbreviations could be unambiguously associated to comprehensible entities. Additional compound splitting on names was performed to find overlapping compounds which indicated relatedness. Similarly, grouping of column names and values depending on similar prefixes or postfixes suggested their cohesiveness. In total 155,085 triples were created which describe 24 tables, 1,717 columns and 13,101 values. There were 1,216 named entity annotations, 1,126 inferred regex patterns and 379 extracted terms stored in the graph. The KG was mainly used to discover column matching candidates and to perform various queries with SPARQL. Project results have been published in a patent about a method and system for determining a pair of table columns for linking [139].

Another related approach investigates how to provide support in the data understanding phase. If data analysts are not familiar with datasets, they need to study (possibly heterogeneous) documentations about it and consult data providers for questions. However, when insights are collected to understand the data, the following main problem remains: data and its documentations are still not logically associated to each other. Therefore, an approach is proposed to semantically enhance the data understanding phase by utilizing DeepLinker (see Section 3.3.3). Figure 34 demonstrates its application for an examined CSV file (left). With deep links that refer to data (certain value in


/test/CWEM_2013_Plots.csv/content/to@csv/rect@7,0,1,1

Csv Resource

SPARQL Endpoint stores triples about currently viewed resource.

Add Triple

property URI

object: http-link or text

Stored Triples (Total:0; Out:0, In:0) &
On-Demand Triples (Total:8; Out:8, In:0)

dl:y →	0
dl:x →	7
dl:width →	1

	A	B	C	D	E	F	G	H	I	J	K	L
1	Author	Station_I	Observat	Observat	plotLead	plotContr	Effort	Floristic	Bryophyt	Lichen	NVC	NVC
2	137-06-0	RRV013I	29-	29-	Forbes	BOYLE;	Very	High	Not	Not	Taxodiun	Bald-
3	137-06-0	RRV013(	29-	29-	Forbes	STANTO	Very	High	Not	Not	Taxodiun	Bald-
4	137-06-0	RRV013/	29-	29-	Forbes	STANTO	Very	High	Not	Not	Taxodiun	Bald-
5	137-06-0	POC016I	30-	30-	Forbes	STANTO	Very	High	Not	Not	Ilex	Little
6	137-06-0	POC016.	30-	30-	Forbes	STANTO	Very	High	Not	Not	Pinus	Pond
7	137-06-0	POC016	30-	30-	Forbes	STANTO	Very	High	Not	Not	Pinus	Pond
8	137-06-0	ALL030E	01-	01-	Forbes	STANTO	Very	High	Not	Not	Pinus	Pond
9	137-06-0	ALL030C	01-	01-	Forbes	STANTO	Very	High	Not	Not	Pinus	Pond
10	137-06-0	ALL030A	01-	01-	Forbes	STANTO	Very	High	Not	Not	Pinus	Pond
11	137-06-0	PLD010E	02-	02-	Forbes	STANTO	Very	High	Not	Not	Juncus	Black

Figure 34: Association of CSV data (left) with documentation in a PDF (right) using DeepLinker and a KG (parts have been published in [148]).

a CSV file) and corresponding documentation (a PDF page), a KG can bidirectionally link both resources. Similarly, data can be further annotated with external KG resources to clarify its content, for example by using DBpedia [10]. With this approach the KG ensures that data and its documentation is logically associated on a fine-grained level.

5.4 CONCLUSION

This chapter covered further contributions by this PhD thesis in diverse domains. Research for personal knowledge assistants was discussed which includes approaches for text analysis, user contexts and interface assistance. For a fast analysis of texts during user activities, an approach for Named Entity Recognition (NER) was presented which is inflection-tolerant and real-time capable. In this regard, libraries related to this topic were mentioned, namely Texana and String Analyzer. Further research was conducted which contributes to the idea of a personal knowledge assistant: Context Spaces enable the contextual association of items in Personal Information Models (PIMOs), while a search engine for the Graphical User Interface (GUI) supports users in finding elements in applications. In a project about the agricultural domain Knowledge Graphs (KGs) were successfully applied to interlink concepts with domain-specific data. Additionally, a viewer was implemented to let farmers interactively browse the graph. In the data science domain KGs were also utilized to support data understanding in data mining projects. In this context, the concept of a semantic data dictionary was pursued in follow-up projects with different focal points. In a feasibility study such a structure was prototypically deployed for databases taken from a data lake to get an overview. Similarly, this concept was used to associate data with documentation for supporting data understanding. In conclusion, once concepts can be interlinked meaningfully the construction and application of KGs shows advantages in diverse domains.

6 CONCLUSION

This chapter concludes the PhD thesis with a summary, lessons learned and an outlook on future work.

6.1 SUMMARY

The absence of data management strategies can reduce data quality and lead to messy enterprise data. Such a condition makes it difficult for companies to use datasets to its full potential. As a solution, the meaning of data was semantically formalized in this work with Knowledge Graphs (KGs), since they model knowledge as entities and their interrelations in a graph. This structure served as a semantic bridge between raw data and domain conceptualization. An illustrative scenario in Figure 2 (Section 1.4) gave an overview about relevant datasets and actors in a KG construction project. As prominent representatives for enterprise data, spreadsheets and Personal Information Management (PIM) data from desktop environments were considered. Domain experts from different user groups were identified as participants in such a project. The construction and maintenance of KGs poses several challenges which were discussed and tackled in this PhD thesis.

Chapter 2 and Chapter 3 covered various approaches which provided solutions for different parts of the problem. They are visually summarized in Figure 35 based on the scenario overview in Figure 2. In the top left corner, approaches for constructing KGs from spreadsheets are arranged. AnnoSpreadKGC followed an interactive approach by extracting data from cells and at the same time annotat-

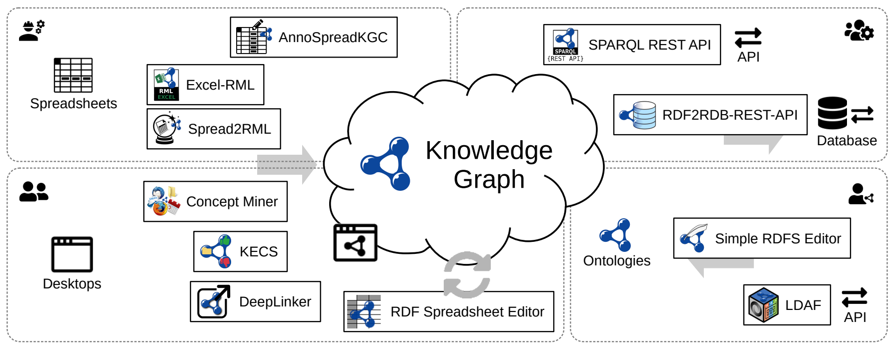


Figure 35: An overview of approaches covered by Chapter 2 and Chapter 3. They revolve around a KG in thematically separated areas. Gray arrows indicate data flow directions.

ing them. The actual mapping of spreadsheet data to KG was enabled with the RDF Mapping Language (RML) and an extension for Excel spreadsheets (Excel-RML). Based on that, RML rules were predicted by Spread2RML to reduce efforts in defining them manually.

Approaches which include experts in the KG construction process were proposed in Chapter 3. In the bottom left corner, users with non-technical backgrounds were involved in knowledge acquisition activities with appropriate interfaces. To acquire relevant personal concepts from them, Concept Miner suggested promising candidates from their personal information sphere. Similarly, such concepts could be discovered in file names by performing knowledge extraction on them (KECS) in an interview setting. As intended in the Semantic Desktop, DeepLinker allowed them to semantically annotate their desktop resources on a fine-grained level. Independent of given data, RDF Spreadsheet Editor let them enter their expertise in form of RDF assertions. With suitable Application Programming Interfaces (APIs), shown in the top right corner, software developers are enabled to take part in KG maintenance. Without learning a query language, SPARQL REST API enabled them to query and manipulate a KG with path-based requests and JSON payloads. RDF2RDB-REST-API allowed them to transform the whole KG into a database with an additionally generated API. In the bottom right corner the group of Semantic Web practitioners are considered. To let them model RDF Schema (RDFS) ontologies for the KG, a simple editor was provided. A Linked Data Application Framework (LDAF) supported them in building LD applications for rapid prototyping. All in all, various approaches were investigated which let Knowledge Engineers (KEs) construct KGs from messy spreadsheets and PIM data on desktops, while experts from different user groups are enabled to take part in the overall process.

Having such methods, the topic switched in Chapter 4 to the challenge of evaluating them. Since open datasets are practically non-existent when it comes to confidential data from Personal Information Management (PIM), the generation of synthetic datasets was proposed as a solution. The start was made by Person Index Generator which produces messy text snippets mentioning persons. With the help of a pattern language, the more versatile Data Sprout generator was presented which is able to produce messy spreadsheets with ground truth data. The last section covered a crawler for PIM data for internal evaluations. To sum up, generators were proposed which are able to produce suitable datasets for conducting KG construction experiments.

Last but not least, Chapter 5 gave an overview of further contributions of this PhD thesis in diverse domains. Among them was the research field about personal knowledge assistants helping users in their daily knowledge work. To do so, methods for text analysis were presented which find named entities and analyze short texts.

Second, the concept of Context Spaces was discussed that let users work in explicit contexts. Third, assistance for the Graphical User Interface (GUI) in form of a search engine was demonstrated. In the domain of smart agriculture KGs were applied to interlink geographical data and telemetry data. Concerning data science, the concept of a semantic data dictionary was described and differently realized in two projects.

Over time mentioned approaches in this PhD thesis have been collected in a toolkit which is presented on the next page.

Hephaistos Toolkit

During research for this PhD thesis many approaches, tools and demos have been implemented which are summarized in a toolkit called Hephaistos. Its loosely coupled tools which are listed in Table 18 find applications in Knowledge Graph Construction (KGC), Human-in-the-Loop (HumL) integration and evaluation (Eval). They are designed for a particular user group, namely Knowledge Engineer (KE), Non-Technical User (NTU), Developer (Dev) and Semantic Web practitioner (SW). An online version of Table 18 is available on a website with additional information and screenshots [137].

Table 18: Hephaistos toolkit with related section, year, category (Cat.), intended user group and references to demo, code and paper.

Tool	Section	Year	Cat.	Group	Demo	Code	Paper
AnnoSpreadKGC	2.3	2021	KGC	KE			[151]
Excel-RML	2.4	2021	KGC	KE	Demo[1]	Code[2]	[145]
Spread2RML	2.5	2021	KGC	KE	Demo[3]	Code[4]	[146]
String Analyzer	5.1.1	2019	KGC	KE	Demo[5]		
Concept Miner	3.3.1	2018	HumL	NTU	Demo[6]		[142]
KECS	3.3.2	2022	HumL	NTU	Demo[7]	Code[8]	[147]
DeepLinker	3.3.3	2018	HumL	NTU	Demo[9]		[141]
RDF Spreadsheet Editor	3.3.4	2017	HumL	NTU	Demo[10]		[150]
Smart Data Browser	5.2	2020	HumL	NTU		Code[11]	
SPARQL REST API	3.4.1	2018	HumL	Dev	Demo[12]		[140]
RDF2RDB-REST-API	3.4.2	2021	HumL	Dev		Code[13]	[153]
Simple RDFS Editor	3.5.1	2021	HumL	SW		Code[14]	
LDAF	3.5.2	2021	HumL	SW	Demo[15]	Code[16]	[143]
Person Index Generator	4.3	2021	Eval	KE		Code[17]	[152]
Data Sprout	4.5	2021	Eval	KE	Demo[18]	Code[19]	[144]

1 https://www.dfki.uni-kl.de/~mschroeder/demo/excel-rml
2 https://github.com/mschroeder-github/rmlmapper-java/tree/mschroeder-features
3 https://www.dfki.uni-kl.de/~mschroeder/demo/spread2rml
4 https://github.com/mschroeder-github/spread2rml
5 https://www.dfki.uni-kl.de/~mschroeder/demo/string-analyzer
6 https://www.dfki.uni-kl.de/~mschroeder/demo/pim-semantifier
7 https://www.dfki.uni-kl.de/~mschroeder/demo/kecs
8 https://github.com/mschroeder-github/kecs
9 https://www.dfki.uni-kl.de/~mschroeder/demo/deeplinker
10 https://www.dfki.uni-kl.de/~mschroeder/demo/rdf-spreadsheet-editor
11 https://github.com/julianklose/sdsd/tree/master/website/src/main/java/de/sdsd/projekt/prototype/smartdatabrowser
12 https://www.dfki.uni-kl.de/~mschroeder/demo/sparql-rest-api
13 https://github.com/mschroeder-github/rdf-to-rdb-rest-api
14 https://github.com/mschroeder-github/simple-rdfs-editor
15 https://www.dfki.uni-kl.de/~mschroeder/demo/ldaf
16 https://github.com/mschroeder-github/ldaf
17 https://github.com/mschroeder-github/person-index
18 http://www.dfki.uni-kl.de/~mschroeder/demo/datasprout
19 https://github.com/mschroeder-github/datasprout

6.2 lessons learned & limitations

All presented approaches showed strengths and weaknesses in their applications. This section summarizes lessons learned and identified limitations.

Concerning Chapter 2, AnnoSpreadKGC demonstrated that in an interactive process Knowledge Engineers (KEs) can react to unexpected situations in messy data with configurations of global extraction methods and incremental annotations. This flexibility comes with the high price of time-consuming manual work and non-reproducible steps. On the contrary, the application of RML has the benefit that its well-defined rules can be reused, shared, communicated and suggested. Yet, mapping of messy data can only work when sources are sufficiently accessible, a local element-by-element mapping is sufficient and functions can be applied to extract or transform values. Especially said functions turned out to be a useful instrument to cope with messy data. Spread2RML showed that even on messy data, heuristics can be enough to predict meaningful mapping rules and that it reduces manual work defining them. However, its evaluation resulted in comparably low scores which reveals that the approach has much room for improvements. In conclusion, investigations pointed out that the construction of KGs from messy data should be performed with a combination of global extraction & annotation techniques and local mapping procedures. Moreover, it has proven to be useful to complement the latter with predictions of mapping rules to reduce manual effort.

In Chapter 3 various lessons could be learned from the ways users were integrated in the construction process. The usage of Concept Miner pointed out that personal and relevant terms can be found in PIM data by ranking them on multiple criteria. However, this is only an initial step towards the modeling of a sophisticated Personal Information Model (PIMO) or more general Personal Knowledge Graph (PKG). The more mature KECS approach showed in expert interviews that already file names are promising sources for construction. Experimental results indicate that with moderate effort a KE is able to construct PKGs from them, while AI is able to predict useful statements. Still, results also reveal that there is room for improvements in AI models as well as in the extraction and management of terminology from file names. The usage of metaphors have proven to be useful in working with KGs. DeepLinker applied the well-known hyperlink metaphor, a familiar concept for people browsing in the Web. As intended in the Semantic Desktop, it lets users annotate their documents with semantic concepts. The ability to formulate deep links has shown to be a crucial step to associate content with them. Yet, its rather unfamiliar annotation process with RDF and the requirement for a separate application are downsides of the approach. RDF Spreadsheet Editor demonstrated that input of expertise as linked data can be successfully supported with the right metaphor, in this case a spreadsheet. Still, such an interface can have its limits, for example, when RDF graphs become large and complex. Both SPARQL REST API and RDF2RDB-REST-API demonstrated in industry projects that familiar interfaces and data structures increase the willingness of developers to deal with semantic technologies. However, these simplified interfaces and structures come with the drawback of reduced functionality and lower expressiveness. This trade-off can also be seen in Simple RDFS Editor which provides easy handling for collaborative ontology editing, but does not fully support all aspects of RDFS. Design decisions made for LDAF are also in line with already mentioned aspects: known interfaces let users collaboratively work in LD applications. In conclusion, for integrating domain experts in the KG construction process a key concept is familiarity, be it familiar metaphors, familiar environments or familiar interfaces. Only then participants are able to comprehend formalized knowledge and be able to give appropriate feedback based on their knowledge.

In Chapter 4 generation patterns has proven to be useful to produce datasets for evaluation. Regarding Person Index Generator, naming patterns introduced variety, ambiguity and messiness in short texts, but did not cover all identified challenges. Data Sprout demonstrated with a larger set that generation patterns are useful to mimic messy spreadsheets in practice. Originating from a KG, the resulting provenance information has proven to be important for obtaining ground truth data. However, used patterns need the right combination, randomness and distribution to produce synthetic datasets which are as authentic as the real ones. In conclusion, the discovery and cataloging of generation patterns as well as the reproduction of them from KG statements enables the production of (messy) synthetic datasets with ground truth annotations.

6.3 FUTURE WORK

In the future, approaches from Chapter 2 and Chapter 3 (see Figure 35) can be further integrated to a single methodology. For now, the PhD thesis investigated various local solutions which are loosely coupled but in fact combinable since they all respect Semantic Web standards. A single consolidated method for KG construction from messy enterprise data could consist of approaches which will be utilized on a central KG as soon as they are required by a use case. As concluded in Chapter 2, this integration also includes the combination of extraction and mapping techniques to a single method. Further related KG approaches could be added for this integration, such as entity alignment [188] or refinement methods in general [114]. Especially emerging KG embeddings [167] could be applied in the context of KG construction.

During this PhD thesis enterprise data was investigated in form of spreadsheets (Chapter 2) and PIM data (Section 3.3.1) including file names (Section 3.3.2). These are prominent examples of enterprise data which were observed in attended research and industry projects. However, in the future more file types and more personal information sources than usual mails, bookmarks and calendars could be considered in companies. One interesting source for future KG construction is user activity locally observed on desktop systems. This includes, for instance, visible texts from a desktop's GUI, clipboard content or low-level events from office applications. Such recorded work evidences could be analyzed to fill or update a KG or learn new relations between its resources. A permanent observation could also notice how vocabulary evolves over time, for example, that new concepts emerge, become irrelevant or change their meaning.

The construction of KGs from data is usually performed in a single mapping step comparable with an Extract-Transform-Load (ETL) approach [8]. However, in this PhD thesis it has already been shown that construction is an iterative process, especially when various approaches are used and users take part in the process. Construction might begin with an existing KG where some parts of a domain is already modeled. From this starting point, several resulting challenges needs to be considered by future work. Since KGs are not initially created, already existing ones need to be updated properly. Mechanisms such as RDF Patch[20] could be used to change (i.e. add or remove) only specific statements. For each new iteration in the construction process, other data sources and human participants could be considered. To reuse modeled knowledge and avoid redundancies, iterations will need to keep track of past modeling attempts.

While performing several iterations, how do Knowledge Engineers (KEs) know that formalized knowledge in the KG goes in the right direction or is sufficient? First answers towards this question were found in KECS (Section 3.3.2) where a Status view (Figure 12d) led the KE with calculated metrics. This concept may be worth further investigating in the future to establish a goal-oriented KG construction process. New metrics might be able to identify the necessity of domain experts and how long or often they should give feedback. Moreover, it could give hints when the coverage of enterprise data with KGs is sufficient.

6.4 CLOSING REMARKS

Especially messy data in enterprises poses new challenges in the construction of KGs. The fact that such data is not comprehensible with common knowledge alone underlines the importance to include domain experts in the process. This PhD thesis considered this scenario

20 https://afs.github.io/rdf-delta/rdf-patch.html

and proposed several solutions which may have an impact on follow-up works about KG construction from messy data.

BIBLIOGRAPHY

[1] Charlie Abela, Chris Staff, and Siegfried Handschuh. "Collecting and Analysing Personal Information Management Data." In: *Proceedings of the First DIACHRON Workshop on Managing the Evolution and Preservation of the Data Web co-located with 12th European Semantic Web Conference (ESWC 2015), Portorož, Slovenia, May 31, 2015*. Ed. by Jeremy Debattista, Mathieu d'Aquin, and Christoph Lange. Vol. 1377. CEUR Workshop Proceedings. CEUR-WS.org, 2015, pp. 22–27. URL: http://ceur-ws.org/Vol-1377/paper3.pdf.

[2] Steven P. Abney. "Partial parsing via finite-state cascades." In: *Nat. Lang. Eng.* 2.4 (1996), pp. 337–344. DOI: 10.1017/S1351324997001599.

[3] Maribel Acosta, Elena Simperl, Fabian Flöck, and Maria-Esther Vidal. "Enhancing answer completeness of SPARQL queries via crowdsourcing." In: *J. Web Semant.* 45 (2017), pp. 41–62. DOI: 10.1016/j.websem.2017.07.001.

[4] Luis von Ahn and Laura Dabbish. "Designing games with a purpose." In: *Commun. ACM* 51.8 (2008), pp. 58–67. DOI: 10.1145/1378704.1378719.

[5] Alfred V. Aho and Margaret J. Corasick. "Efficient String Matching: An Aid to Bibliographic Search." In: *Commun. ACM* 18.6 (1975), pp. 333–340. DOI: 10.1145/360825.360855.

[6] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language - Towns, Buildings, Construction*. Oxford Univ. Press, 1977. ISBN: 978-0-19-501919-3.

[7] Francesco Antoniazzi and Fabio Viola. "RDF Graph Visualization Tools: a Survey." In: *23rd Conference of Open Innovations Association, FRUCT 2018, Bologna, Italy, November 13-16, 2018*. IEEE, 2018, pp. 25–36. DOI: 10.23919/FRUCT.2018.8588069.

[8] Julián Arenas-Guerrero, Mario Scrocca, Ana Iglesias-Molina, Jhon Toledo, Luis Pozo-Gilo, Daniel Doña, Óscar Corcho, and David Chaves-Fraga. "Knowledge Graph Construction with R2RML and RML: An ETL System-based Overview." In: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*. Ed. by David Chaves-Fraga, Anastasia Dimou, Pieter Heyvaert, Freddy Priyatna, and Juan F. Sequeda. Vol. 2873. CEUR Workshop Proceed-

ings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2873/paper11.pdf.

[9] Mark van Assem, Hajo Rijgersberg, Mari Wigham, and Jan L. Top. "Converting and Annotating Quantitative Data Tables." In: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010*. Vol. 6496. Lecture Notes in Computer Science. Springer, 2010, pp. 16–31. DOI: 10.1007/978-3-642-17746-0_2.

[10] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. "DBpedia: A Nucleus for a Web of Open Data." In: *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*. Ed. by Karl Aberer, Key-Sun Choi, Natasha Fridman Noy, Dean Allemang, Kyung-Il Lee, Lyndon J. B. Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux. Vol. 4825. Lecture Notes in Computer Science. Springer, 2007, pp. 722–735. DOI: 10.1007/978-3-540-76298-0_52.

[11] Eirik Bakke, David R. Karger, and Rob Miller. "A spreadsheet-based user interface for managing plural relationships in structured data." In: *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Vancouver, BC, Canada, May 7-12, 2011*. Ed. by Desney S. Tan, Saleema Amershi, Bo Begole, Wendy A. Kellogg, and Manas Tungare. ACM, 2011, pp. 2541–2550. DOI: 10.1145/1978942.1979313.

[12] Krisztian Balog and Tom Kenter. "Personal Knowledge Graphs: A Research Agenda." In: *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019*. Ed. by Yi Fang, Yi Zhang, James Allan, Krisztian Balog, Ben Carterette, and Jiafeng Guo. ACM, 2019, pp. 217–220. DOI: 10.1145/3341981.3344241.

[13] Alberto Bartoli, Andrea De Lorenzo, Eric Medvet, and Fabiano Tarlao. "Inference of Regular Expressions for Text Extraction from Examples." In: *IEEE Trans. Knowl. Data Eng.* 28.5 (2016), pp. 1217–1230. DOI: 10.1109/TKDE.2016.2515587.

[14] Robert Battle and Edward Benson. "Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST)." In: *J. Web Semant.* 6.1 (2008), pp. 61–69. DOI: 10.1016/j.websem.2007.11.002.

[15] Kristi L. Berg, Tom Seymour, and Richa Goel. "History Of Databases." In: *International Journal of Management & Information Systems (IJMIS)* 17.1 (2012), pp. 29–36. DOI: 10.19030/ijmis.v17i1.7587.

[16] Ivelize Rocha Bernardo, Matheus Silva Mota, and André Santanché. "Extracting and Semantically Integrating Implicit Schemas from Multiple Spreadsheets of Biology based on the Recognition of their Nature." In: *JIDM* 4.2 (2013), pp. 104–113.

[17] Tim Berners-Lee. *Information management: A proposal*. Tech. rep. CERN Document, Mar. 1989. URL: https://cds.cern.ch/record/369245/files/dd-89-001.pdf (visited on Feb. 1, 2022).

[18] Tim Berners-Lee. *The Original HTTP as defined in 1991*. 1991. URL: https://www.w3.org/Protocols/HTTP/AsImplemented.html (visited on Feb. 1, 2022).

[19] Tim Berners-Lee. *Weaving the Web: The original design and ultimate destiny of the World Wide Web*. Harper, 1999. ISBN: 978-0-06-251586-5.

[20] Tim Berners-Lee, James Hendler, and Ora Lassila. "The semantic web." In: *Scientific american* 284.5 (2001), pp. 34–43.

[21] Christian Bizer and Andreas Schultz. "The Berlin SPARQL Benchmark." In: *Int. J. Semantic Web Inf. Syst.* 5.2 (2009), pp. 1–24. DOI: 10.4018/jswis.2009040101.

[22] Christian Bizer and Andreas Schultz. "The Berlin SPARQL Benchmark." In: *Semantic Services, Interoperability and Web Applications - Emerging Concepts*. CRC Press, 2011, pp. 81–103. DOI: 10.4018/978-1-60960-593-3.ch004.

[23] Christian Bizer and Andy Seaborne. "D2RQ - Treating Non-RDF Databases as Virtual RDF Graphs." In: *Proceedings of the 3rd international semantic web conference (ISWC 2004)*. 2004.

[24] Jérôme Blanchard, Yolande Belaïd, and Abdel Belaïd. "Automatic Generation of a Custom Corpora for Invoice Analysis and Recognition." In: *Workshop on Industrial Applications of Document Analysis and Recognition, WIADAR@ICDAR 2019, Sydney, Australia, September 22-25, 2019*. IEEE, 2019, p. 1. DOI: 10.1109/ICDARW.2019.60121.

[25] Sara Bonfitto, Elena Casiraghi, and Marco Mesiti. "Table understanding approaches for extracting knowledge from heterogeneous tables." In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 11.4 (2021). DOI: 10.1002/widm.1407.

[26] Angela Bonifati, Irena Holubová, Arnau Prat-Pérez, and Sherif Sakr. "Graph Generators: State of the Art and Open Challenges." In: *ACM Comput. Surv.* 53.2 (2020), 36:1–36:30. DOI: 10.1145/3379445.

[27] Leo Breiman. "Random Forests." In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[28] Dan Brickley and Libby Miller. *FOAF Vocabulary Specification 0.9*. May 24, 2007. URL: xmlns.com/foaf/spec/ (visited on Feb. 1, 2022).

[29] Polly S. Brown and John D. Gould. "An Experimental Study of People Creating Spreadsheets." In: *ACM Trans. Inf. Syst.* 5.3 (1987), pp. 258–272. DOI: 10.1145/27641.28058.

[30] Nicolas Bruno and Surajit Chaudhuri. "Flexible Database Generators." In: *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*. Ed. by Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi. ACM, 2005, pp. 1097–1107. URL: http://www.vldb.org/archives/website/2005/program/paper/wed/p1097-bruno.pdf.

[31] Helun Bu and Kazuhiro Kuwabara. "Toward Blockchain-Assisted Gamified Crowdsourcing for Knowledge Refinement." In: *Intelligent Information and Database Systems - 12th Asian Conference, ACIIDS 2020, Phuket, Thailand, March 23-26, 2020, Proceedings, Part I*. Ed. by Ngoc Thanh Nguyen, Kietikul Jearanaitanakij, Ali Selamat, Bogdan Trawinski, and Suphamit Chittayasothorn. Vol. 12033. Lecture Notes in Computer Science. Springer, 2020, pp. 3–14. DOI: 10.1007/978-3-030-41964-6_1.

[32] Vannevar Bush. "As We May Think." In: *Atl. Mon.* 176.1 (1945), pp. 101–108. URL: http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm.

[33] John M Carroll. "Creative names for personal files in an interactive computing environment." In: *International Journal of Man-Machine Studies* 16.4 (1982), pp. 405–438.

[34] David Chaves-Fraga, Freddy Priyatna, Andrea Cimmino, Jhon Toledo, Edna Ruckhaus, and Óscar Corcho. "GTFS-Madrid-Bench: A benchmark for virtual knowledge graph access in the transport domain." In: *J. Web Semant.* 65 (2020), p. 100596. DOI: 10.1016/j.websem.2020.100596.

[35] Alexandros Chortaras and Giorgos Stamou. "D2RML: Integrating Heterogeneous Data and Web Services into Custom RDF Graphs." In: *Workshop on Linked Data on the Web co-located with The Web Conference 2018, LDOW@WWW 2018, Lyon, France April 23rd, 2018*. Ed. by Tim Berners-Lee, Sarven Capadisli, Stefan Dietze, Aidan Hogan, Krzysztof Janowicz, and Jens Lehmann. Vol. 2073. CEUR Workshop Proceedings. CEUR-WS.org, 2018. URL: http://ceur-ws.org/Vol-2073/article-07.pdf.

[36] Jessica Chwalek. "Context Mining." Student research project complemented by seminar paper. TU Kaiserslautern, 2019.

[37] Kenneth Clarkson, Anna Lisa Gentile, Daniel Gruhl, Petar Ristoski, Joseph Terdiman, and Steve Welch. "User-Centric Ontology Population." In: *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Ed. by Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam. Vol. 10843. Lecture Notes in Computer Science. Springer, 2018, pp. 112–127. DOI: 10.1007/978-3-319-93417-4_8.

[38] Marco Cremaschi, Anisa Rula, Alessandra Siano, and Flavio De Paoli. "MantisTable: A Tool for Creating Semantic Annotations on Tabular Data." In: *The Semantic Web: ESWC 2019 Satellite Events - ESWC 2019 Satellite Events, Portorož, Slovenia, June 2-6, 2019, Revised Selected Papers*. Vol. 11762. Lecture Notes in Computer Science. Springer, 2019, pp. 18–23. DOI: 10.1007/978-3-030-32327-1_4.

[39] Douglas Crockford. *Introducing JSON*. 2002. URL: https://www.json.org/ (visited on Feb. 1, 2022).

[40] Jerome W Crowder, Jonathan S Marion, and Michele Reilly. "File naming in digital media research: Examples from the humanities and social sciences." In: *Journal of Librarianship and Scholarly Communication* 3.3 (2015).

[41] CrowedFlower. *Data Scientist Report 2015*. 2015. URL: https://visit.figure-eight.com/rs/416-ZBE-142/images/Crowdflower_Data_Scientist_Survey2015.pdf (visited on Feb. 1, 2022).

[42] Enrico Daga, Luca Panziera, and Carlos Pedrinaci. "A BASILar Approach for Building Web APIs on Top of SPARQL Endpoints." In: *Proceedings of the Third Workshop on Services and Applications over Linked APIs and Data co-located with the 12th Extended Semantic Web Conference (ESWC 2015), Portorož, Slovenia, June 1, 2015*. Ed. by Maria Maleshkova, Ruben Verborgh, and Steffen Stadtmüller. Vol. 1359. CEUR Workshop Proceedings. CEUR-WS.org, 2015, pp. 22–32. URL: http://ceur-ws.org/Vol-1359/paper4.pdf.

[43] Stefan Decker and Martin R. Frank. "The Networked Semantic Desktop." In: *Proceedings of the WWW2004 Workshop on Application Design, Development and Implementation Issues in the Semantic Web, New York, NY, USA, May 18, 2004*. Ed. by Christoph Bussler, Stefan Decker, Daniel Schwabe, and Oscar Pastor. Vol. 105. CEUR Workshop Proceedings. CEUR-WS.org, 2004. URL: http://ceur-ws.org/Vol-105/DeckerFrank.pdf.

[44] United States Department of Defense. *DoD Modeling and Simulation (M&S) Glossary*. 5000.59-M. 1998. ISBN: 978-1-4820-9539-5. URL: https://www.acqnotes.com/Attachments/DoD%205000.59-M%20DoD%20Modeling%20and%20Simulation%20Glossary.pdf (visited on Feb. 1, 2022).

[45] Thomas Delva, Julián Arenas-Guerrero, Ana Iglesias-Molina, Óscar Corcho, David Chaves-Fraga, and Anastasia Dimou. "RML-star: A Declarative Mapping Language for RDF-star Generation." In: *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021*. Ed. by Oshani Seneviratne, Catia Pesquita, Juan Sequeda, and Lorena Etcheverry. Vol. 2980. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2980/paper374.pdf.

[46] Thomas Delva, Dylan Van Assche, Pieter Heyvaert, Ben De Meester, and Anastasia Dimou. "Integrating Nested Data into Knowledge Graphs with RML Fields." In: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*. Ed. by David Chaves-Fraga, Anastasia Dimou, Pieter Heyvaert, Freddy Priyatna, and Juan F. Sequeda. Vol. 2873. CEUR Workshop Proceedings. CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2873/paper9.pdf.

[47] Andreas Dengel. "Six Thousand Words about Multi-Perspective Personal Document Management." In: *Tenth IEEE International Enterprise Distributed Object Computing Conference (EDOC 2006), 16-20 October 2006, Hong Kong, China, Workshops*. IEEE Computer Society, 2006, p. 62. DOI: 10.1109/EDOCW.2006.63.

[48] Andreas Dengel. *Semantische Technologien – Grundlagen. Konzepte. Technologien.* Jan. 2012. ISBN: 978-3-82742663-5. DOI: 10.1007/978-3-8274-2664-2.

[49] Andreas Dengel and Heiko Maus. "Personalisierte Wissensdienste: Das Unternehmen denkt mit." In: *IM+io* 3 (Sept. 2018), pp. 46–49. URL: https://www.im-io.de/kuenstliche-intelligenz/personalisierte-wissensdienste-das-unternehmen-denkt-mit/ (visited on Feb. 1, 2022).

[50] Anastasia Dimou and Miel Vander Sande. *RDF Mapping Language (RML)*. Jan. 21, 2020. URL: https://rml.io/specs/rml/ (visited on Feb. 1, 2022).

[51] Anastasia Dimou, Miel Vander Sande, Pieter Colpaert, Ruben Verborgh, Erik Mannens, and Rik Van de Walle. "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data." In: *Proceedings of the Workshop on Linked Data on*

the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014. Ed. by Christian Bizer, Tom Heath, Sören Auer, and Tim Berners-Lee. Vol. 1184. CEUR Workshop Proceedings. CEUR-WS.org, 2014. URL: http://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf.

[52] Alan Dix, Janet Finlay, Gregory D Abowd, and Russell Beale. *Human-computer interaction*. Pearson Education, 2004. ISBN: 978-0-13-046109-4.

[53] Dejing Dou, Hao Wang, and Haishan Liu. “Semantic data mining: A survey of ontology-based approaches.” In: *Proceedings of the 9th IEEE International Conference on Semantic Computing, ICSC 2015, Anaheim, CA, USA, February 7-9, 2015*. Ed. by Mohan S. Kankanhalli, Tao Li, and Wei Wang. IEEE Computer Society, 2015, pp. 244–251. DOI: 10.1109/ICOSC.2015.7050814.

[54] Olakunle Elijah, Tharek Abdul Rahman, Igbafe Orikumhi, Chee Yen Leow, and Mohammad Nour Hindia. “An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges.” In: *IEEE Internet Things J.* 5.5 (2018), pp. 3758–3773. DOI: 10.1109/JIOT.2018.2844296.

[55] Ludger van Elst and Malte Kiesel. “Generating and Integrating Evidence for Ontology Mappings.” In: *Engineering Knowledge in the Age of the Semantic Web, 14th International Conference, EKAW 2004, Whittlebury Hall, UK, October 5-8, 2004, Proceedings*. Ed. by Enrico Motta, Nigel Shadbolt, Arthur Stutt, and Nicholas Gibbins. Vol. 3257. Lecture Notes in Computer Science. Springer, 2004, pp. 15–29. DOI: 10.1007/978-3-540-30202-5_2.

[56] David Eppstein. “Subgraph Isomorphism in Planar Graphs and Related Problems.” In: *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1995. San Francisco, California, USA*. Ed. by Kenneth L. Clarkson. ACM/SIAM, 1995, pp. 632–640. URL: http://dl.acm.org/citation.cfm?id=313651.313830.

[57] Christina Feilmayr and Wolfram Wöß. “An analysis of ontologies and their success factors for application to business.” In: *Data Knowl. Eng.* 101 (2016), pp. 1–23. DOI: 10.1016/j.datak.2015.11.003.

[58] Dieter Fensel, Umutcan Simsek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. *Knowledge Graphs - Methodology, Tools and Selected Use Cases*. Springer, 2020. ISBN: 978-3-03037438-9. DOI: 10.1007/978-3-030-37439-6.

[59] Roy Thomas Fielding. “Architectural styles and the design of network-based software architectures.” PhD thesis. Dept. Inf. Comput. Sci. Univ. California, Irvine, 2000. URL: http:

//www.ics.uci.edu/~fielding/pubs/dissertation/top.htm (visited on Feb. 1, 2022).

[60] Manuel Fiorelli, Tiziano Lorenzetti, Maria Teresa Pazienza, Armando Stellato, and Andrea Turbati. "Sheet2RDF: a Flexible and Dynamic Spreadsheet Import&Lifting Framework for RDF." In: *Current Approaches in Applied Artificial Intelligence - 28th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2015, Seoul, South Korea, June 10-12, 2015, Proceedings*. Ed. by Moonis Ali, Young Sig Kwon, Chang-Hwan Lee, Juntae Kim, and Yongdai Kim. Vol. 9101. Lecture Notes in Computer Science. Springer, 2015, pp. 131–140. DOI: 10.1007/978-3-319-19066-2_13.

[61] Manuel Fiorelli and Armando Stellato. "Lifting Tabular Data to RDF: A Survey." In: *Metadata and Semantic Research - 14th International Conference, MTSR 2020, Madrid, Spain, December 2-4, 2020, Revised Selected Papers*. Ed. by Emmanouel Garoufallou and María Antonia Ovalle-Perandones. Vol. 1355. Communications in Computer and Information Science. Springer, 2020, pp. 85–96. DOI: 10.1007/978-3-030-71903-6_9.

[62] Martin Fowler. *POJO*. Dec. 8, 2003. URL: https://www.martinfowler.com/bliki/POJO.html (visited on Feb. 1, 2022).

[63] Michael J. Franklin, Alon Y. Halevy, and David Maier. "From databases to dataspaces: a new abstraction for information management." In: *SIGMOD Rec.* 34.4 (2005), pp. 27–33. DOI: 10.1145/1107499.1107502.

[64] Mikhail Galkin, Sören Auer, Maria-Esther Vidal, and Simon Scerri. "Enterprise Knowledge Graphs: A Semantic Approach for Knowledge Management in the Next Generation of Enterprise Information Systems." In: *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 2, Porto, Portugal, April 26-29, 2017*. Ed. by Slimane Hammoudi, Michal Smialek, Olivier Camp, and Joaquim Filipe. SciTePress, 2017, pp. 88–98. DOI: 10.5220/0006325200880098.

[65] Johanna Geiß, Andreas Spitz, and Michael Gertz. "NECKAr: A Named Entity Classifier for Wikidata." In: *Language Technologies for the Challenges of the Digital Age - 27th International Conference, GSCL 2017, Berlin, Germany, September 13-14, 2017, Proceedings*. Ed. by Georg Rehm and Thierry Declerck. Vol. 10713. Lecture Notes in Computer Science. Springer, 2017, pp. 115–129. DOI: 10.1007/978-3-319-73706-5_10. URL: https://event.ifi.uni-heidelberg.de/?page_id=532 (visited on Feb. 1, 2022).

[66] Anna Lisa Gentile, Daniel Gruhl, Petar Ristoski, and Steve Welch. "Personalized Knowledge Graphs for the Pharmaceutical Domain." In: *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, Oc-*

tober 26-30, 2019, Proceedings, Part II. Ed. by Chiara Ghidini, Olaf Hartig, Maria Maleshkova, Vojtech Svátek, Isabel F. Cruz, Aidan Hogan, Jie Song, Maxime Lefrançois, and Fabien Gandon. Vol. 11779. Lecture Notes in Computer Science. Springer, 2019, pp. 400–417. DOI: 10.1007/978-3-030-30796-7_25.

[67] David Gonçalves. "Pseudo-desktop collections and PIM: The missing link." In: *ECIR 2011 Workshop on Evaluating Personal Search*. 2011, pp. 3–4.

[68] Thomas Gruber. "A translation approach to portable ontology specifications." In: *Knowledge acquisition* 5.2 (1993), pp. 199–220.

[69] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. "LUBM: A benchmark for OWL knowledge base systems." In: *J. Web Semant.* 3.2-3 (2005), pp. 158–182. DOI: 10.1016/j.websem.2005.06.005.

[70] Karl Hammar. "Linked Data Creation with ExcelRDF." In: *The Semantic Web: ESWC 2020 Satellite Events - ESWC 2020 Satellite Events, Heraklion, Crete, Greece, May 31 - June 4, 2020, Revised Selected Papers*. Ed. by Andreas Harth, Valentina Presutti, Raphaël Troncy, Maribel Acosta, Axel Polleres, Javier D. Fernández, Josiane Xavier Parreira, Olaf Hartig, Katja Hose, and Michael Cochez. Vol. 12124. Lecture Notes in Computer Science. Springer, 2020, pp. 104–109. DOI: 10.1007/978-3-030-62327-2_18.

[71] Lushan Han, Tim Finin, Cynthia Sims Parr, Joel Sachs, and Anupam Joshi. "RDF123: From Spreadsheets to RDF." In: *The Semantic Web - ISWC 2008, 7th International Semantic Web Conference, ISWC 2008, Karlsruhe, Germany, October 26-30, 2008. Proceedings*. Ed. by Amit P. Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy W. Finin, and Krishnaprasad Thirunarayan. Vol. 5318. Lecture Notes in Computer Science. Springer, 2008, pp. 451–466. DOI: 10.1007/978-3-540-88564-1_29.

[72] Olaf Hartig. "Foundations of RDF⋆ and SPARQL⋆ (An Alternative Approach to Statement-Level Metadata in RDF)." In: *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017*. Ed. by Juan L. Reutter and Divesh Srivastava. Vol. 1912. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1912/paper12.pdf.

[73] Jörn Hees, Thomas Roth-Berghofer, Ralf Biedert, Benjamin Adrian, and Andreas Dengel. "BetterRelations: Using a Game to Rate Linked Data Triples." In: *KI 2011: Advances in Artificial Intelligence, 34th Annual German Conference on AI, Berlin, Germany, October 4-7,2011. Proceedings*. Ed. by Joscha Bach and

Stefan Edelkamp. Vol. 7006. Lecture Notes in Computer Science. Springer, 2011, pp. 134–138. DOI: 10.1007/978-3-642-24455-1_12.

[74] Desiree Heim. "Contextifier: Retrospective Context Mining on (Big) Personal Data." Bachelor Thesis. TU Kaiserslautern, June 1, 2021.

[75] Sven Hertling, Markus Schröder, Christian Jilek, and Andreas Dengel. "Top-k Shortest Paths in Directed Labeled Multigraphs." In: *Semantic Web Challenges - Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected Papers*. Vol. 641. Communications in Computer and Information Science. Springer, May 2016, pp. 200–212. DOI: 10.1007/978-3-319-46565-4_16.

[76] Sven Hertling, Markus Schröder, Christian Jilek, and Andreas Dengel. "Where is that Button Again?! - Towards a Universal GUI Search Engine." In: *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017*. SciTePress, Feb. 2017, pp. 217–227. DOI: 10.5220/0006201402170227.

[77] Ben J. Hicks, Andy Dong, R. Palmer, and Hamish C. McAlpine. "Organizing and managing personal electronic files: A mechanical engineer's perspective." In: *ACM Trans. Inf. Syst.* 26.4 (2008), 23:1–23:40. DOI: 10.1145/1402256.1402262.

[78] Joseph E. Hoag and Craig W. Thompson. "A parallel general-purpose synthetic data generator." In: *SIGMOD Rec.* 36.1 (2007), pp. 19–24. DOI: 10.1145/1276301.1276305.

[79] Andreas Holzinger. "Interactive machine learning for health informatics: when do we need the human-in-the-loop?" In: *Brain Informatics* 3.2 (2016), pp. 119–131. DOI: 10.1007/s40708-016-0042-6.

[80] Wen Hua, Zhongyuan Wang, Haixun Wang, Kai Zheng, and Xiaofang Zhou. "Short text understanding through lexical-semantic analysis." In: *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*. Ed. by Johannes Gehrke, Wolfgang Lehner, Kyuseok Shim, Sang Kyun Cha, and Guy M. Lohman. IEEE Computer Society, 2015, pp. 495–506. DOI: 10.1109/ICDE.2015.7113309.

[81] Enrique Iglesias, Samaneh Jozashoori, David Chaves-Fraga, Diego Collarana, and Maria-Esther Vidal. "SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs." In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Ireland, October 19-23, 2020*. ACM, 2020, pp. 3039–3046.

[82] Paul Jaccard. “Lois de distribution florale dans la zone alpine.” In: *Bull. Soc. Vaudoise Sci. Nat.* 38 (1902), pp. 69–130.

[83] Daniel R. Jeske, Behrokh Samadi, Pengyue J. Lin, Lan Ye, Sean Cox, Rui Xiao, Ted Younglove, Minh Ly, Douglas Holt, and Ryan Rich. “Generation of synthetic data sets for evaluating the accuracy of knowledge discovery systems.” In: *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA, August 21-24, 2005*. Ed. by Robert Grossman, Roberto J. Bayardo, and Kristin P. Bennett. ACM, 2005, pp. 756–762. DOI: 10.1145/1081870.1081969.

[84] Shouling Ji, Prateek Mittal, and Raheem A. Beyah. “Graph Data Anonymization, De-Anonymization Attacks, and De-Anonymizability Quantification: A Survey.” In: *IEEE Commun. Surv. Tutorials* 19.2 (2017), pp. 1305–1326. DOI: 10.1109/COMST.2016.2633620.

[85] Valentin Jijkoun, Mahboob Alam Khalid, Maarten Marx, and Maarten de Rijke. “Named entity normalization in user generated content.” In: *Proceedings of the Second Workshop on Analytics for Noisy Unstructured Text Data, AND 2008, Singapore, July 24, 2008*. Ed. by Daniel P. Lopresti, Shourya Roy, Klaus U. Schulz, and L. Venkata Subramaniam. Vol. 303. ACM International Conference Proceeding Series. ACM, 2008, pp. 23–30. DOI: 10.1145/1390749.1390755.

[86] Christian Jilek. “Self-organizing Context Spaces to Support Information Management and Knowledge Work.” Under review. PhD thesis. TU Kaiserslautern, 2022.

[87] Christian Jilek, Jessica Chwalek, Sven Schwarz, Markus Schröder, Heiko Maus, and Andreas Dengel. “Advanced Memory Buoyancy for Forgetful Information Systems.” In: *AIS Transactions on Enterprise Systems* 4.1 (May 2019). DOI: 10.30844/ais-tes.v4i1.11. arXiv: 1811.12177.

[88] Christian Jilek, Yannick Runge, Claudia Niederée, Heiko Maus, Tobias Tempel, Andreas Dengel, and Christian Frings. “Managed Forgetting to Support Information Management and Knowledge Work.” In: *Künstliche Intell.* 33.1 (2019), pp. 45–55. DOI: 10.1007/s13218-018-00568-9.

[89] Christian Jilek, Markus Schröder, Rudolf Novik, Sven Schwarz, Heiko Maus, and Andreas Dengel. “Inflection-Tolerant Ontology-Based Named Entity Recognition for Real-Time Applications.” In: *2nd Conference on Language, Data and Knowledge, LDK 2019, May 20-23, 2019, Leipzig, Germany*. Vol. 70. OASICS. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, May 2019, 11:1–11:14. DOI: 10.4230/OASIcs.LDK.2019.11.

[90] Christian Jilek, Markus Schröder, Sven Schwarz, Heiko Maus, and Andreas Dengel. "Context Spaces as the Cornerstone of a Near-Transparent and Self-reorganizing Semantic Desktop." In: *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*. Vol. 11155. Lecture Notes in Computer Science. Springer, June 2018, pp. 89–94. DOI: 10.1007/978-3-319-98192-5_17.

[91] Wolfgang Jochum. "Requirements of Fragment Identification." In: *International Conferences on New Media Technology and Semantic Systems*. 2007, pp. 172–179.

[92] Mladjan Jovanovic. "Gamifying knowledge maintenance." In: 2015. URL: https://www.essence-network.com/wp-content/uploads/2015/08/GamifyingKnowledgeMaintenance_Jovanovic.pdf (visited on Feb. 1, 2022).

[93] JSON-RPC Working Group. *JSON-RPC 2.0 Specification*. Jan. 4, 2013. URL: https://www.jsonrpc.org/specification (visited on Feb. 1, 2022).

[94] William Kent. "A Simple Guide to Five Normal Forms in Relational Database Theory." In: *Commun. ACM* 26.2 (1983), pp. 120–125. DOI: 10.1145/358024.358054.

[95] Jinyoung Kim and W. Bruce Croft. "Retrieval experiments using pseudo-desktop collections." In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009*. Ed. by David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, and Jimmy Lin. ACM, 2009, pp. 1297–1306. DOI: 10.1145/1645953.1646117.

[96] Bryan Klimt and Yiming Yang. "The Enron Corpus: A New Dataset for Email Classification Research." In: *Machine Learning: ECML 2004, 15th European Conference on Machine Learning, Pisa, Italy, September 20-24, 2004, Proceedings*. Ed. by Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi. Vol. 3201. Lecture Notes in Computer Science. Springer, 2004, pp. 217–226. DOI: 10.1007/978-3-540-30115-8_22.

[97] Julian Klose, Markus Schröder, Silke Becker, Ansgar Bernardi, and Arno Ruckelshausen. "Datenaufbereitung in der Landwirtschaft durch automatisierte semantische Annotation." In: *40. GIL - Jahrestagung, Informatik in der Land-, Forst- und Ernährungswirtschaft, Fokus: Digitalisierung für Mensch, Umwelt und Tier, 17. - 18. Februar 2020, Campus Weihenstephan, Freising, Germany*. Vol. P-299. LNI. Gesellschaft für Informatik e.V., Feb. 2020, pp. 133–138. arXiv: 1911.06606. URL: https://dl.gi.de/20.500.12116/31882.

[98] Steffen Lamparter, Marc Ehrig, and Christoph Tempich. "Knowledge Extraction from Classification Schemas." In: *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, OTM Conf. Int'l. Conf., Agia Napa, Cyprus, October 25-29, 2004, Proceedings, Part I*. Ed. by Robert Meersman and Zahir Tari. Vol. 3290. LNCS. Springer, 2004, pp. 618–636. DOI: 10.1007/978-3-540-30468-5_40.

[99] Andreas Langegger and Wolfram Wöß. "XLWrap - Querying and Integrating Arbitrary Spreadsheets with SPARQL." In: *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*. Vol. 5823. Lecture Notes in Computer Science. Springer, 2009, pp. 359–374.

[100] Bettina Laugwitz, Theo Held, and Martin Schrepp. "Construction and Evaluation of a User Experience Questionnaire." In: *HCI and Usability for Education and Work, 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society, USAB 2008, Graz, Austria, November 20-21, 2008. Proceedings*. Ed. by Andreas Holzinger. Vol. 5298. Lecture Notes in Computer Science. Springer, 2008, pp. 63–76. DOI: 10.1007/978-3-540-89350-9_6.

[101] V. I. Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions and Reversals." In: *Soviet Physics Doklady* 10 (Feb. 1966).

[102] Zongmin Ma, Miriam A. M. Capretz, and Li Yan. "Storing massive Resource Description Framework (RDF) data: a survey." In: *Knowl. Eng. Rev.* 31.4 (2016), pp. 391–413. DOI: 10.1017/S0269888916000217.

[103] Alejandro Mallea, Marcelo Arenas, Aidan Hogan, and Axel Polleres. "On Blank Nodes." In: *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Bonn, Germany, October 23-27, 2011, Proceedings, Part I*. Ed. by Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Fridman Noy, and Eva Blomqvist. Vol. 7031. Lecture Notes in Computer Science. Springer, 2011, pp. 421–437. DOI: 10.1007/978-3-642-25073-6_27.

[104] H. Maus, S. Schwarz, and A. Dengel. "Weaving Personal Knowledge Spaces into Office Applications." In: *Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives*. Springer, 2013, pp. 71–82.

[105] Ben De Meester, Anastasia Dimou, Ruben Verborgh, and Erik Mannens. "An Ontology to Semantically Declare and Describe Functions." In: *The Semantic Web - ESWC 2016 Satellite Events, Heraklion, Crete, Greece, May 29 - June 2, 2016, Revised Selected*

Papers. Vol. 9989. Lecture Notes in Computer Science. 2016, pp. 46–49. DOI: 10.1007/978-3-319-47602-5_10.

[106] Microsoft Corporation. *Microsoft Active Accessibility: Architecture*. Aug. 1, 2001. URL: https://docs.microsoft.com/de-de/previous-versions/windows/desktop/dnacc/microsoft-active-accessibility (visited on Feb. 1, 2022).

[107] Varish Mulwad, Tim Finin, and Anupam Joshi. "A Domain Independent Framework for Extracting Linked Semantic Data from Tables." In: *Search Computing - Broadening Web Search*. Vol. 7538. Lecture Notes in Computer Science. Springer, 2012, pp. 16–33. DOI: 10.1007/978-3-642-34213-4_2.

[108] Mark A. Musen. "The protégé project: a look back and a look forward." In: *AI Matters* 1.4 (2015), pp. 4–12. DOI: 10.1145/2757001.2757003. URL: https://protege.stanford.edu/ (visited on Feb. 1, 2022).

[109] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. "Data Lake Management: Challenges and Opportunities." In: *Proc. VLDB Endow.* 12.12 (2019), pp. 1986–1989. DOI: 10.14778/3352063.3352116.

[110] Rudolf Novik. "Searching Forgetful Information Systems." Bachelor thesis. TU Kaiserslautern, 2019.

[111] Martin J. O'Connor, Christian Halaschek-Wiener, and Mark A. Musen. "Mapping Master: A Flexible Approach for Mapping Spreadsheets to OWL." In: *The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part II*. Ed. by Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm. Vol. 6497. Lecture Notes in Computer Science. Springer, 2010, pp. 194–208. DOI: 10.1007/978-3-642-17749-1_13.

[112] Allard Oelen, Markus Stocker, and Sören Auer. "Creating a Scholarly Knowledge Graph from Survey Article Tables." In: *Digital Libraries at Times of Massive Societal Transition - 22nd International Conference on Asia-Pacific Digital Libraries, ICADL 2020, Kyoto, Japan, November 30 - December 1, 2020, Proceedings*. Ed. by Emi Ishita, Natalie Lee-San Pang, and Lihong Zhou. Vol. 12504. Lecture Notes in Computer Science. Springer, 2020, pp. 373–389. DOI: 10.1007/978-3-030-64452-9_35.

[113] Judith Reitman Olson and Henry H Rueter. "Extracting expertise from experts: Methods for knowledge acquisition." In: *Expert systems* 4.3 (1987), pp. 152–168.

[114] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods." In: *Semantic Web* 8.3 (2017), pp. 489–508. DOI: 10.3233/SW-160218.

[115] Georgios Petasis, Vangelis Karkaletsis, Georgios Paliouras, Anastasia Krithara, and Elias Zavitsanos. "Ontology Population and Enrichment: State of the Art." In: *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution - Bridging the Semantic Gap*. Ed. by Georgios Paliouras, Constantine D. Spyropoulos, and George Tsatsaronis. Vol. 6050. Lecture Notes in Computer Science. Springer, 2011, pp. 134–166. DOI: 10.1007/978-3-642-20795-2_6.

[116] Minh Pham, Suresh Alse, Craig A. Knoblock, and Pedro A. Szekely. "Semantic Labeling: A Domain-Independent Approach." In: *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part I*. Ed. by Paul Groth, Elena Simperl, Alasdair J. G. Gray, Marta Sabou, Markus Krötzsch, Freddy Lécué, Fabian Flöck, and Yolanda Gil. Vol. 9981. Lecture Notes in Computer Science. 2016, pp. 446–462. DOI: 10.1007/978-3-319-46523-4_27.

[117] Srdan Popic, Bogdan Pavkovic, Ivan Velikic, and Nikola Teslic. "Data generators: a short survey of techniques and use cases with focus on testing." In: *9th IEEE Int'l. Conf. on Consumer Electronics, ICCE, Berlin, Germany*. IEEE, 2019, pp. 189–194. DOI: 10.1109/ICCE-Berlin47944.2019.8966202.

[118] Gianluca Quercini and Chantal Reynaud. "Entity discovery and annotation in tables." In: *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*. ACM, 2013, pp. 693–704. DOI: 10.1145/2452376.2452457.

[119] Tilmann Rabl, Michael Frank, Hatem Mousselly Sergieh, and Harald Kosch. "A Data Generator for Cloud-Scale Benchmarking." In: *Performance Evaluation, Measurement and Characterization of Complex Systems - Second TPC Technology Conference, TPCTC 2010, Singapore, September 13-17, 2010. Revised Selected Papers*. Ed. by Raghunath Othayoth Nambiar and Meikel Poess. Vol. 6417. Lecture Notes in Computer Science. Springer, 2010, pp. 41–56. DOI: 10.1007/978-3-642-18206-8_4.

[120] Tilmann Rabl and Meikel Poess. "Parallel data generation for performance analysis of large, complex RDBMS." In: *Proceedings of the Fourth International Workshop on Testing Database Systems, DBTest 2011, Athens, Greece, June 13, 2011*. Ed. by Goetz Graefe and Kenneth Salem. ACM, 2011, p. 5. DOI: 10.1145/1988842.1988847.

[121] Jyothsna Rachapalli, Vaibhav Khadilkar, Murat Kantarcioglu, and Bhavani Thuraisingham. "RETRO: a framework for semantics preserving SQL-to-SPARQL translation." In: (2011).

[122] Sunitha Ramanujam, Anubha Gupta, Latifur Khan, Steven Seida, and Bhavani M. Thuraisingham. "R2D: A Bridge between the Semantic Web and Relational Visualization Tools." In: *Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009), 14-16 September 2009, Berkeley, CA, USA*. IEEE Computer Society, 2009, pp. 303–311. DOI: 10.1109/ICSC.2009.29.

[123] Shreyas Suresh Rao and Ashalatha Nayak. "LinkED: A Novel Methodology for Publishing Linked Enterprise Data." In: *J. Comput. Inf. Technol.* 25.3 (2017), pp. 191–209. URL: http://cit.fer.hr/index.php/CIT/article/view/3477.

[124] Mohammad Rifat Ahmmad Rashid, Giuseppe Rizzo, Marco Torchiano, Nandana Mihindukulasooriya, Óscar Corcho, and Raúl García-Castro. "Completeness and consistency analysis for evolving knowledge bases." In: *J. Web Semant.* 54 (2019), pp. 48–71. DOI: 10.1016/j.websem.2018.11.004.

[125] Thomas C. Redman. *Data quality for the information age*. Artech House, 1996. ISBN: 978-0-89006-883-0.

[126] Marina Riga, Panagiotis Mitzias, Efstratios Kontopoulos, and Ioannis Kompatsiaris. "PROPheT - Ontology Population and Semantic Enrichment from Linked Data Sources." In: *Data Analytics and Management in Data Intensive Domains - XIX International Conference, DAMDID/RCDL 2017, Moscow, Russia, October 10-13, 2017, Revised Selected Papers*. Ed. by Leonid A. Kalinichenko, Yannis Manolopoulos, Oleg Malkov, Nikolay A. Skvortsov, Sergey A. Stupnikov, and Vladimir Sukhomlin. Vol. 822. Communications in Computer and Information Science. Springer, 2017, pp. 157–168. DOI: 10.1007/978-3-319-96553-6_12.

[127] Pramod J. Sadalage and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Addison-Wesley Professional, 2012. ISBN: 978-0-321-82662-6.

[128] Abdul Samad, Mamoona Qadir, Ishrat Nawaz, Muhammad Arshad Islam, and Muhammad Aleem. "A Comprehensive Survey of Link Prediction Techniques for Social Network." In: *EAI Endorsed Trans. Ind. Networks Intell. Syst.* 7.23 (2020), e3. DOI: 10.4108/eai.13-7-2018.163988.

[129] Leo Sauermann, Ansgar Bernardi, and Andreas Dengel. "Overview and Outlook on the Semantic Desktop." In: *Proceedings of the ISWC 2005 Workshop on The Semantic Desktop - Next Generation Information Management & Collaboration Infrastructure. Galway, Ireland, November 6, 2005*. Ed. by Stefan Decker, Jack Park, Dennis Quan, and Leo Sauermann. Vol. 175. CEUR Workshop Proceedings. CEUR-WS.org, 2005. URL: http://ceur-ws.org/Vol-175/18_sauermann_overviewsemdesk_final.pdf.

[130] Leo Sauermann, Ludger van Elst, and Andreas Dengel. “PIMO - A Framework for Representing Personal Information Models.” In: *Proceedings of I-MEDIA '07 and I-SEMANTICS '07 International Conferences on New Media Technology and Semantic Systems as part of (TRIPLE-I 2007), September 5-7, Graz, Austria. International Conferences on Knowledge Management and New Media Technology (I-MEDIA)*. Ed. by T. Pellegrini and S. Schaffert. J.UCS. Know-Center, Austria. Online-Proceedings, Sept. 2007, pp. 270–277.

[131] Agata Savary and Jakub Piskorski. “Lexicons and grammars for named entity annotation in the National corpus of Polish.” In: *Intelligent Information Systems* (2010), pp. 141–154.

[132] Michael Schmidt, Thomas Hornung, Michael Meier, Christoph Pinkel, and Georg Lausen. “SP^{2}Bench: A SPARQL Performance Benchmark.” In: *Semantic Web Information Management - A Model-Based Perspective*. Ed. by Roberto De Virgilio, Fausto Giunchiglia, and Letizia Tanca. Springer, 2009, pp. 371–393. DOI: 10.1007/978-3-642-04329-1_16.

[133] Markus Schröder. “Efficient High-Level Semantic Enrichment of Undocumented Enterprise Data.” In: *The Semantic Web: ESWC 2019 Satellite Events - ESWC 2019 Satellite Events, Portorož, Slovenia, June 2-6, 2019, Revised Selected Papers*. Vol. 11762. Lecture Notes in Computer Science. Springer, June 2019, pp. 220–230. DOI: 10.1007/978-3-030-32327-1_41.

[134] Markus Schröder. *A Pattern Language for Spreadsheets*. Mar. 1, 2021. URL: https://www.dfki.uni-kl.de/~mschroeder/pattern-language-spreadsheets/.

[135] Markus Schröder. *Guideline Ontology*. Mar. 4, 2021. URL: https://www.dfki.uni-kl.de/~mschroeder/ld/gl.

[136] Markus Schröder. *Spreadsheet Ontology*. Feb. 10, 2021. URL: https://www.dfki.uni-kl.de/~mschroeder/ld/ss.

[137] Markus Schröder. *Hephaistos Toolkit*. Apr. 14, 2022. URL: https://www.dfki.uni-kl.de/~mschroeder/hephaistos/.

[138] Markus Schröder. *Semantic Technologies for Busy People*. Jan. 24, 2022. URL: https://www.dfki.uni-kl.de/~mschroeder/tutorial/semtec/.

[139] Markus Schröder, Sven Böhmert, Heiko Maus, Geraldine Bous, Andy Bruntsch, and Benny Kneissl. “Verfahren und System zur Bestimmung eines Paars von Tabellenspalten zur Verknüpfung.” Pat. 10 2018 129 138.8. Bayerische Motoren Werke AG. May 20, 2020. URL: https://register.dpma.de/DPMAregister/pat/register?AKZ=1020181291388 (visited on Feb. 1, 2022).

[140] Markus Schröder, Jörn Hees, Ansgar Bernardi, Daniel Ewert, Peter Klotz, and Steffen Stadtmüller. “Simplified SPARQL REST API - CRUD on JSON Object Graphs via URI Paths.” In: *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*. Vol. 11155. Lecture Notes in Computer Science. Springer, June 2018, pp. 40–45. DOI: 10.1007/978-3-319-98192-5_8. arXiv: 1805.01825.

[141] Markus Schröder, Christian Jilek, and Andreas Dengel. “Deep Linking Desktop Resources.” In: *The Semantic Web: ESWC 2018 Satellite Events - ESWC 2018 Satellite Events, Heraklion, Crete, Greece, June 3-7, 2018, Revised Selected Papers*. Vol. 11155. Lecture Notes in Computer Science. Springer, June 2018, pp. 202–207. DOI: 10.1007/978-3-319-98192-5_38. arXiv: 1805.03491.

[142] Markus Schröder, Christian Jilek, and Andreas Dengel. “Interactive Concept Mining on Personal Data - Bootstrapping Semantic Services.” In: *CoRR* abs/1903.05872 (2019). arXiv: 1903.05872.

[143] Markus Schröder, Christian Jilek, and Andreas Dengel. “A Linked Data Application Framework to Enable Rapid Prototyping.” In: *CoRR* abs/2104.13605 (2021). arXiv: 2104.13605.

[144] Markus Schröder, Christian Jilek, and Andreas Dengel. “Dataset Generation Patterns for Evaluating Knowledge Graph Construction.” In: *The Semantic Web: ESWC 2021 Satellite Events - Virtual Event, June 6-10, 2021, Revised Selected Papers*. Vol. 12739. Lecture Notes in Computer Science. Springer, June 2021, pp. 27–32. DOI: 10.1007/978-3-030-80418-3_5. arXiv: 2104.13576.

[145] Markus Schröder, Christian Jilek, and Andreas Dengel. “Mapping Spreadsheets to RDF: Supporting Excel in RML.” In: *Proceedings of the 2nd International Workshop on Knowledge Graph Construction co-located with 18th Extended Semantic Web Conference (ESWC 2021), Online, June 6, 2021*. Vol. 2873. CEUR Workshop Proceedings. CEUR-WS.org, June 2021. arXiv: 2104.13600. URL: http://ceur-ws.org/Vol-2873/paper3.pdf.

[146] Markus Schröder, Christian Jilek, and Andreas Dengel. “Spread2RML: Constructing Knowledge Graphs by Predicting RML Mappings on Messy Spreadsheets.” In: *K-CAP ’21: Knowledge Capture Conference, Virtual Event, USA, December 2-3, 2021*. Best Research Paper. ACM, Dec. 2021, pp. 145–152. DOI: 10.1145/3460210.3493544. arXiv: 2110.12829.

[147] Markus Schröder, Christian Jilek, and Andreas Dengel. “A Human-in-the-Loop Approach for Personal Knowledge Graph Construction from File Names.” In: *Proceedings of the 3rd International Workshop on Knowledge Graph Construction (KGCW 2022) co-located with 19th Extended Semantic Web Conference (ESWC*

2022), Hersonissos, Greek, May 30, 2022. Ed. by David Chaves-Fraga, Anastasia Dimou, Pieter Heyvaert, Freddy Priyatna, and Juan Sequeda. Vol. 3141. CEUR Workshop Proceedings. CEUR-WS.org, 2022. URL: http://ceur-ws.org/Vol-3141/paper2.pdf.

[148] Markus Schröder, Christian Jilek, Jörn Hees, and Andreas Dengel. "Towards Semantically Enhanced Data Understanding." In: *CoRR* abs/1806.04952 (2018). arXiv: 1806.04952.

[149] Markus Schröder, Christian Jilek, Jörn Hees, Sven Hertling, and Andreas Dengel. "RDF Spreadsheet Editor: Get (G)rid of Your RDF Data Entry Problems." In: *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*. Vol. 1963. CEUR Workshop Proceedings. CEUR-WS.org, Oct. 2017. URL: http://ceur-ws.org/Vol-1963/paper635.pdf.

[150] Markus Schröder, Christian Jilek, Jörn Hees, Sven Hertling, and Andreas Dengel. "An Easy & Collaborative RDF Data Entry Method using the Spreadsheet Metaphor." In: *CoRR* abs/1804.04175 (2018). arXiv: 1804.04175.

[151] Markus Schröder, Christian Jilek, Michael Schulze, and Andreas Dengel. "Interactively Constructing Knowledge Graphs from Messy User-Generated Spreadsheets." In: *CoRR* abs/2103.03537 (2021). arXiv: 2103.03537.

[152] Markus Schröder, Christian Jilek, Michael Schulze, and Andreas Dengel. "The Person Index Challenge: Extraction of Persons from Messy, Short Texts." In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 2, Online Streaming, February 4-6, 2021*. SCITEPRESS, Feb. 2021, pp. 531–537. DOI: 10.5220/0010188405310537. arXiv: 2011.07990.

[153] Markus Schröder, Michael Schulze, Christian Jilek, and Andreas Dengel. "Bridging the Technology Gap between Industry and Semantic Web: Generating Databases and Server Code from RDF." In: *Proceedings of the 13th International Conference on Agents and Artificial Intelligence, ICAART 2021, Volume 2, Online Streaming, February 4-6, 2021*. SCITEPRESS, Feb. 2021, pp. 507–514. DOI: 10.5220/0010186005070514. arXiv: 2011.07957.

[154] Michael Schulze, Markus Schröder, Christian Jilek, Torsten Albers, Heiko Maus, and Andreas Dengel. "P2P-O: A Purchase-To-Pay Ontology for Enabling Semantic Invoices." In: *The Semantic Web - 18th International Conference, ESWC 2021, Virtual Event, June 6-10, 2021, Proceedings*. Vol. 12731. Lecture Notes in Computer Science. Springer, June 2021, pp. 647–663. DOI: 10.1007/978-3-030-77385-4_39.

[155] Michael Schulze, Markus Schröder, Christian Jilek, and Andreas Dengel. "ptpDG: A Purchase-To-Pay Dataset Generator for Evaluating Knowledge-Graph-Based Services." In: *Proceedings of the ISWC 2021 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 20th International Semantic Web Conference (ISWC 2021), Virtual Conference, October 24-28, 2021*. Vol. 2980. CEUR Workshop Proceedings. CEUR-WS.org, Oct. 2021. URL: http://ceur-ws.org/Vol-2980/paper386.pdf.

[156] Sven Schwarz and Thomas Roth-Berghofer. "Towards Goal Elicitation by User Observation." In: *Proceedings of the LLWA 2003. LLWA, Karlsruhe*. Ed. by Andreas Hotho and Gerd Stumme. AIFB Karlsruhe. Karlsruhe: GI, Oct. 2003, pp. 224–228.

[157] Ming Sheng, Jingwen Wang, Yong Zhang, Xin Li, Chao Li, Chunxiao Xing, Qiang Li, Yuyao Shao, and Han Zhang. "DocKG: A Knowledge Graph Framework for Health with Doctor-in-the-Loop." In: *Health Information Science - 8th International Conference, HIS 2019, Xi'an, China, October 18-20, 2019, Proceedings*. Ed. by Hua Wang, Siuly Siuly, Rui Zhou, Fernando Martín-Sánchez, Yanchun Zhang, and Zhisheng Huang. Vol. 11837. Lecture Notes in Computer Science. Springer, 2019, pp. 3–14. DOI: 10.1007/978-3-030-32962-4_1.

[158] Umutcan Simsek, Elias Kärle, and Dieter Fensel. "RocketRML - A NodeJS Implementation of a Use Case Specific RML Mapper." In: *Joint Proceedings of the 1st International Workshop on Knowledge Graph Building and 1st International Workshop on Large Scale RDF Analytics (ESWC 2019), Portorož, Slovenia, June 3, 2019*. Vol. 2489. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 46–53.

[159] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge." In: *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*. Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy. ACM, 2007, pp. 697–706. DOI: 10.1145/1242572.1242667.

[160] Zareen Saba Syed, Tim Finin, and Anupam Joshi. "Wikitology: Using Wikipedia as an Ontology." In: *Proceedings of the Grace Hopper Celebration of Women in Computing Conference*. Oct. 2008.

[161] Ignacio G. Terrizzano, Peter M. Schwarz, Mary Roth, and John E. Colino. "Data Wrangling: The Challenging Yourney from the Wild to the Lake." In: *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org, 2015. URL: http://cidrdb.org/cidr2015/Papers/CIDR15_Paper2.pdf.

[162] *The Linked Open Data Cloud*. May 5, 2021. URL: https://lod-cloud.net/ (visited on Feb. 1, 2022).

[163] Peter P. Uhrowczik. “Data Dictionary/Directories.” In: *IBM Syst. J.* 12.4 (1973), pp. 332–350. DOI: 10.1147/sj.124.0332.

[164] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. “Silk - A Link Discovery Framework for the Web of Data.” In: *Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009*. Ed. by Christian Bizer, Tom Heath, Tim Berners-Lee, and Kingsley Idehen. Vol. 538. CEUR Workshop Proceedings. CEUR-WS.org, 2009. URL: http://ceur-ws.org/Vol-538/ldow2009_paper13.pdf.

[165] Denny Vrandecic. “Wikidata: a new platform for collaborative data collection.” In: *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*. Ed. by Alain Mille, Fabien Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab. ACM, 2012, pp. 1063–1064. DOI: 10.1145/2187980.2188242.

[166] Jingjing Wang, Haixun Wang, Zhongyuan Wang, and Kenny Qili Zhu. “Understanding Tables on the Web.” In: *Conceptual Modeling - 31st International Conference ER 2012, Florence, Italy, October 15-18, 2012. Proceedings*. Ed. by Paolo Atzeni, David W. Cheung, and Sudha Ram. Vol. 7532. Lecture Notes in Computer Science. Springer, 2012, pp. 141–155. DOI: 10.1007/978-3-642-34002-4_11.

[167] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. “Knowledge Graph Embedding: A Survey of Approaches and Applications.” In: *IEEE Trans. Knowl. Data Eng.* 29.12 (2017), pp. 2724–2743. DOI: 10.1109/TKDE.2017.2754499.

[168] Susan C Weller and A Kimball Romney. *Systematic data collection*. Sage publications, 1988. ISBN: 978-0-8039-3073-5.

[169] Rüdiger Wirth and Jochen Hipp. “CRISP-DM: Towards a standard process model for data mining.” In: *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*. 2000, pp. 29–39.

[170] World Wide Web Consortium. *HTML Tags*. Nov. 3, 1992. URL: https://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html (visited on Feb. 1, 2022).

[171] World Wide Web Consortium. *"Deep Linking" in the World Wide Web*. Sept. 11, 2003. URL: https://www.w3.org/2001/tag/doc/deeplinking.html (visited on Feb. 1, 2022).

[172] World Wide Web Consortium. *SKOS Simple Knowledge Organization System Primer*. Aug. 18, 2009. URL: https://www.w3.org/TR/skos-primer/ (visited on Feb. 1, 2022).

[173] World Wide Web Consortium. *CURIE Syntax 1.0*. Dec. 16, 2010. URL: https://www.w3.org/TR/2010/NOTE-curie-20101216/ (visited on Feb. 1, 2022).

[174] World Wide Web Consortium. *RdfSyntax*. Jan. 27, 2011. URL: https://www.w3.org/wiki/RdfSyntax (visited on Feb. 1, 2022).

[175] World Wide Web Consortium. *A Direct Mapping of Relational Data to RDF*. Sept. 27, 2012. URL: https://www.w3.org/TR/rdb-direct-mapping/ (visited on Feb. 1, 2022).

[176] World Wide Web Consortium. *Media Fragments URI 1.0 (basic)*. Sept. 25, 2012. URL: https://www.w3.org/TR/media-frags/ (visited on Feb. 1, 2022).

[177] World Wide Web Consortium. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. Dec. 11, 2012. URL: https://www.w3.org/TR/owl2-overview/ (visited on Feb. 1, 2022).

[178] World Wide Web Consortium. *R2RML: RDB to RDF Mapping Language*. Sept. 27, 2012. URL: https://www.w3.org/TR/r2rml/ (visited on Feb. 1, 2022).

[179] World Wide Web Consortium. *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes*. Apr. 5, 2012. URL: https://www.w3.org/TR/xmlschema11-2/ (visited on Feb. 1, 2022).

[180] World Wide Web Consortium. *Property Paths*. Mar. 21, 2013. URL: https://www.w3.org/TR/sparql11-query/#propertypaths (visited on Feb. 1, 2022).

[181] World Wide Web Consortium. *SPARQL 1.1 Query Language*. Mar. 21, 2013. URL: https://www.w3.org/TR/sparql11-query/ (visited on Feb. 1, 2022).

[182] World Wide Web Consortium. *RDF 1.1 Primer*. June 24, 2014. URL: https://www.w3.org/TR/rdf11-primer/ (visited on Feb. 1, 2022).

[183] World Wide Web Consortium. *RDF 1.1 Turtle*. Feb. 25, 2014. URL: https://www.w3.org/TR/turtle/ (visited on Feb. 1, 2022).

[184] World Wide Web Consortium. *RDF Schema 1.1*. Feb. 25, 2014. URL: https://www.w3.org/TR/rdf-schema/ (visited on Feb. 1, 2022).

[185] World Wide Web Consortium. *Generating RDF from Tabular Data on the Web*. Dec. 17, 2015. URL: https://www.w3.org/TR/csv2rdf/ (visited on Feb. 1, 2022).

[186] World Wide Web Consortium. *JSON-LD 1.1 - A JSON-based Serialization for Linked Data*. July 16, 2020. URL: https://www.w3.org/TR/json-ld11/ (visited on Feb. 1, 2022).

[187] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Qili Zhu. “Probase: a probabilistic taxonomy for text understanding.” In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2012, Scottsdale, AZ, USA, May 20-24, 2012*. Ed. by K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, and Ariel Fuxman. ACM, 2012, pp. 481–492. DOI: 10.1145/2213836.2213891.

[188] Kaisheng Zeng, Chengjiang Li, Lei Hou, Juanzi Li, and Ling Feng. “A comprehensive survey of entity alignment for knowledge graphs.” In: *AI Open* 2 (2021), pp. 1–13. ISSN: 2666-6510. DOI: 10.1016/j.aiopen.2021.02.002. URL: https://www.sciencedirect.com/science/article/pii/S2666651021000036.

[189] Ziqi Zhang, Jie Gao, and Fabio Ciravegna. “JATE 2.0: Java Automatic Term Extraction with Apache Solr.” In: *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. Ed. by Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asunción Moreno, Jan Odijk, and Stelios Piperidis. European Language Resources Association (ELRA), 2016. URL: http://www.lrec-conf.org/proceedings/lrec2016/summaries/211.html (visited on Feb. 1, 2022).

CURRICULUM VITAE: MARKUS SCHRÖDER

CONTACT INFORMATION

Website:	`https://www.dfki.uni-kl.de/~mschroeder/`
Email:	markus.schroeder@dfki.de

EDUCATION

2006–2009	Karl Kübel Schule, Bensheim (KKS) General higher education entrance qualification A-level subjects: Computer Science, Mathematics
2009–2012	Technische Universität Darmstadt (TU Darmstadt) Bachelor of Science in Computer Science Thesis: *Enabling Semantic Web Services for Desktop Environment*
2012–2015	Technische Universität Darmstadt (TU Darmstadt) Master of Science in Computer Science Thesis: *Discovery of Interaction Patterns with Graphical User Interface Usage Mining*

EXPERIENCE

since 2015 German Research Center for Artificial Intelligence (DFKI), Smart Data & Knowledge Services Dept. (SDS), Prof. Dr. Andreas Dengel

Full-time Researcher, selected projects:

2015–2018	PRO-OPT: Big Data Production Optimization in Smart Ecosystems
2016–2017	SuGraBo: Search Engine for Graphical User Interfaces
2017–2020	SDSD: Smarte Daten & Smarte Dienste
2018–2020	Geobox: Standardisierung der Geobox-Infrastruktur
since 2020	SensAI: Self-organizing Personal Knowledge Assistants in Evolving Corporate Memories

SELECTED PUBLICATIONS

ICAART 2017	Where is that Button again?! – Towards a Universal GUI Search Engine [76] S. Hertling, **M. Schröder**, C. Jilek, A. Dengel
ISWC 2017	RDF Spreadsheet Editor: Get (G)rid of Your RDF Data Entry Problems [149] **M. Schröder**, C. Jilek, J. Hees, S. Hertling, A. Dengel
ESWC 2021	Dataset Generation Patterns for Evaluating Knowledge Graph Construction [144] **M. Schröder**, C. Jilek, A. Dengel
K-CAP 2021	Spread2RML: Constructing Knowledge Graphs by Predicting RML Mappings on Messy Spreadsheets [146] **M. Schröder**, C. Jilek, A. Dengel
ESWC 2022	A Human-in-the-Loop Approach for Personal Knowledge Graph Construction from File Names [147] **M. Schröder**, C. Jilek, A. Dengel

AWARDS AND NOMINATIONS

2016	Winner of Top-k Shortest Paths in Large Typed RDF Graphs Challenge [75] S. Hertling, M. Schröder, C. Jilek, A. Dengel at Extended Semantic Web Conference (ESWC)
2017	Best Paper Award for "Where is that Button Again?! - Towards a Universal GUI Search Engine" [76] S. Hertling, M. Schröder, C. Jilek, A. Dengel at International Conference on Agents and Artificial Intelligence (ICCART)
2019	Best Research Paper Award for "Inflection-Tolerant Ontology-Based Named Entity Recognition for Real-Time Applications" [89] C. Jilek, M. Schröder, R. Novik, S. Schwarz, H. Maus, A. Dengel at Conference on Language, Data and Knowledge (LDK)
2021	Best Poster Award Nomination for "The Person Index Challenge: Extraction of Persons from Messy, Short Texts" [152] M. Schröder, C. Jilek, M. Schulze, A. Dengel at International Conference on Agents and Artificial Intelligence (ICCART)

	Best Paper Award for “Spread2RML: Constructing Knowledge Graphs by Predicting RML Mappings on Messy Spreadsheets” [146] M. Schröder, C. Jilek, A. Dengel at International Conference on Knowledge Capture (K-CAP)

TALKS AND PRESENTATIONS

2018	Smart Data Analytics - Optimierung in Automobilproduktion Invited Talk at Big-Data.AI Summit in Hanau
	Knowledge Graph Feasibility Study Final Presentation
2019	Efficient High-Level Semantic Enrichment of Undocumented Enterprise Data Presentation at PhD Symposium, Extended Semantic Web Conference (ESWC)
2021	Building Knowledge Graphs from Messy Enterprise Data Talk at Department of Computer Science, TU Kaiserslautern
	Building Knowledge Graphs from Messy Enterprise Data Invited Talk at Leibniz Universität Hannover (LUH)
2022	Knowledge Graph Construction Tutorial (KGCT) Presentation at Extended Semantic Web Conference (ESWC)

PEER REVIEWING

2021	Workshop on Intelligent Decision Support Systems in Fintech (WIDSSF) at Iberian Conference on Information Systems and Technologies (CISTI)
2022	International Conference on Semantics Systems (SEMANTiCS)

SUMMER SCHOOLS & COMMUNITY GROUPS

2019	European Summer School on Explainable Data Science, European Association for Data Science (EuADS)
since 2020	W3C Knowledge Graph Construction Community Group (KG-Construct)

TEACHING

since 2015	Artificial Intelligence Group, TU Kaiserslautern, Prof. Dr. Andreas Dengel (AGD) Supervision of 4 master theses 5 seminars 6 student projects 12 student research assistants
2015	Tutoring for Einführung in die symbolische Künstliche Intelligenz (ESKI)
2021	Lecture about Knowledge Graph Construction in Applications of Machine Learning and Data Science (AMLDS)