CLEF2023 SimpleText Task 2, 3: Identification and Simplification of Difficult Terms

Notebook for the SimpleText Lab at CLEF 2023

Dennis R. Davari[1], Antonela Prnjak[2], Kristina Schmitt[3]

[1] University od Kiel, Christian-Albrechts-Platz 4, Kiel, Germany
[2] University of Split, Ruđera Boškovića 31, Split, Croatia
[3] University of Malta, University of Malta, Msida MSD 2080, Malta

Abstract

Comprehending scientific texts is crucial for succeeding in education and beyond. However, individuals who are not experts in a particular scientific field often face difficulties understanding such documents. This is because scientific articles contain challenging vocabulary, complicated language, and lengthy structures, which make them difficult to comprehend without prior knowledge. This working note contains the results for SimpleText task 2 (identification and explanation of difficult concepts) and SimpleText task 3 (simplification of difficult text passages). Several statistical and AI-based models were used to solve these tasks. The results of the models were assessed according to the quality of the results.

In order to solve the tasks mostly autoregressive Large Language Models (LLMs) were used. In order for BLOOM and BLOOMZ to work effectively prompts with examples were needed while for GPT-3 simple command prompts were more effective. Apart from LLMs statistical and graph-based models were also used.

Keywords

reading comprehension, scientific texts, text simplification, CLEF 2023 SimpleText lab competition [1], natural language processing (NLP), GPT, BLOOMZ, ST5, WIKI, AI

1. Introduction

Insufficient fundamental knowledge can inhibit reading comprehension and limit access to information. Simplifying scientific texts can serve as a helpful tool as it aims to enhance the reader's comprehension by connecting complex content to fundamental vocabulary. Hence, scientific concepts become accessible by adjusting the linguistic level. However, traditional methods of text simplification may eradicate crucial aspects of concepts and structures. Simplification aims to reduce text complexity while preserving the original information and meaning.

For the CLEF 2023 SimpleText lab competition [2][3], task 2, which seeks to identify difficult concepts and subsequently explain them, and task 3, which aims to produce a simplified version of a

[1]CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece

scientific text, were dealt with. Different models were used to execute the tasks while taking their quality of the results into account as a final way of assessment.

Nowadays, the trend for sentence simplification has shifted towards utilizing deep learning techniques, which is in line with most natural language processing (NLP) tasks. One area of research involves a so-called sequence-to-sequence-based neural network to forecast explicit edit operations to create a simplified sentence taken from the original data. Dong et al. [4] proclaim a neural Programmer-Interpreter approach as an approximation to the human ability of repetitious cognitive adaption. Later, Cumbicus-Pineda et al. [5] added a graph module that incorporated the syntactic information of sentences to facilitate detecting complex phrases during the simplification process. Meanwhile, however, it is common practice to take advantage of existing pre-trained language models such as BERT [6], RoBERTa [7], or GPT-3 [8].

2. Approach

2.1.1 Task description

The goal of task 2 is to identify up to five difficult terms in a given sentence [2]. For each term an explanation needs to be provided [2]. Additionally an identified difficult term needs to be rated on a scale from 0 to 2 where 0 denotes a term where the meaning can be derived and 2 denotes a term where deep knowledge is needed to comprehend the term.

2.1.2 Data description

The train dataset used for this task consists of different sentences from abstracts from high-ranked scientific publications [2, 3]. For each sentence a difficult term along with a difficulty rating and a definition for the term are given. The total amount of rows (i. e. sentences) for the train dataset consists of 453 rows. For solving this task a subset of 203 rows were used. The subset of the train set only contains difficulty ratings of 1 (moderately difficult term) and 2 (very difficult term).

For the test set a total amount of 152,072 sentences (i. e. rows) are given. This dataset is denoted as the large dataset. The medium dataset is a subset of this dataset and consists of 4,797 sentences and the small dataset is a subset of the medium dataset and consists of 2,234 sentences. Task 2 was solved using the medium dataset.

2.2 Method description

2.2.1 Identifying difficult terms

For the identification of difficult terms four different models were used. Two AI models which were used are BLOOM and GPT-3. One statistical model, YAKE, and one graph-based model, TextRank, was used. No manual interventations were made for the results.

YAKE

YAKE is an unsupervised model which is able to automatically extract keywords in a given text based on statistical properties of the text [9]. YAKE was used through the PKE package [10]. No special parameter modification was made when using YAKE since the default parameters for the implementation via PKE were used.

TextRank

Just like YAKE TextRank also automatically extracts keywords in a given text [10]. In contrast to YAKE is TextRank a graph-based model [11]. TextRank was also accessed via the PKE package [9] and the default model parameters were used.

BLOOM

BLOOM is an LLM which is trained to continue text from a prompt [12]. The prompt which was used can be found in the appendix. The examples which were used in the prompt are a small subset of the train dataset. BLOOM was used through the Hosted Inference API from Hugging Face [13] and no changes to the default parameters were made.

GPT-3

GPT-3 is also an LLM and was introduced in 2020 [14]. The prompt which was used is in the appendix. The model which was used is text-davinci-003. The temperature was set to 0.7. The maximum amount of tokens is 256. The top probability was set to 1 and the frequence and presence penalty were set to 0.

2.2.1.1 Rating of difficult terms

For the rating of the identified difficult terms BLOOM was used for the difficult terms generated by YAKE, TextRank and BLOOM. GPT-3 was used for rating the difficult terms generated by GPT-3. The prompts for BLOOM and GPT-3 can be found in the appendix. The prompt used for BLOOM is based on examples from the train dataset. Since the rating only consists of a one-digit integer the maximum output length parameter was decreased for both GPT-3 and BLOOM to save tokens and computation time. Other than that the parameters for BLOOM and GPT-3 remained unchanged. Both prompts can be found in the appendix.

2.2.1.2 Explanation of difficult terms

The best performing dataset of the previous subtask (extraction and rating of difficult terms) was used for all models of this subtask. In our case it was the dataset where GPT-3 was used for identifying and rating difficult terms. We evaluated the quality of the results of the different models manually by looking at a subset of the results for each model. In order to explain difficult terms four different models were used: Wikipedia, simpleT5, BLOOMZ and GPT-3.

Wikipedia

Wikipedia is a library for Python which retrieves a summarized version of the Wikipedia definition for a given term [15]. If no entry exists for this term then no output is returned [15]. In our case a missing Wikipedia entry for a term is characterized by an output of -1.

simpleT5

simpleT5 is an LLM which is based on the Transformer architecture [16]. For solving this task the pretrained model t5-base was used. The model was also trained using the test dataset. Here the first 40 rows were used for evaluation and the remaining 163 rows were used for training the model. For training the maximum token length for the input is 128 and the maximum output length is 512. The batch size was set to 8 and the model was trained for 5 epochs. The precision was set to 32. Out of the 5 generated models the model with the lowest value loss was chosen. The model which was chosen had a value loss of 2.7969.

BLOOMZ

BLOOMZ is a finetuned variant of BLOOM [17]. For this task the test dataset was split in two parts. One part consists of difficult terms which are abbreviations and the other part is the remaining dataset. The regular expression which was used to extract the abbreviations can be found in the appendix. If an extracted abbreviation already contains the deciphering such as „IoT (Internet of Things)“ then there was no need to further use any model to get the deciphering of the abbreviation. For abbreviations and non-abbreviations different prompts were used which can be both found in the appendix. Just like with BLOOM the Hosted Inference API was used and except for the maximum token length which was set to 512 all parameters were set to the default values.

GPT-3
For GPT-3 the dataset was also split in abbreviations and non-abbreviations using the same regular expression. If an extracted abbreviation already contains the deciphering such as „IoT (Internet of Things)“ then there was no need to further use any model to get the deciphering of the abbreviation. The prompt for both subsets can be found in the appendix. The model text-davinci-003 was used. The temperature was set to 0.7 and the maximum token length was set to 256. The top probability was set to 1 and the frequence and presence penalty were set to 0.

2.3 SimpleText task 3
The goal of task 3 is to simplify a given scientific sentence. [1] For solving this task simpleT5, BLOOMZ and GPT-3 were used.

2.3.1 Data description
The train data consists of 648 sentences from different scientific domains where there exists a simplified sentence for a given complex sentence. [1, 2, 3] Just like for task 2 there are three differently sized datasets for the test dataset which only contain the non-simplified sentences. [1] The medium-sized dataset was chosen for this task. This dataset consists of 4,797 sentences.

2.3.2 Method description
simpleT5
The first 130 sentences of the train dataset were used to evaluate the training process and the remaining 518 sentences were used for training the simpleT5 model. For training the maximum token length of the input complex sentences was set to 512 and the maximum token length for the output simplified sentences was also set to 512. The batch size was set to 4 and amount of epochs was set to 5. The precision was set to 32. Moreover, the pretrained t5-base model was also loaded.

BLOOMZ
The prompt can be found in the appendix. The prompt constitutes a small subset of the train dataset. The Hosted Inference API was used and except for the maximum token length which was set to 512 all parameters were set to the default values.

GPT-3
The prompt can be found in the appendix. The model which was used was text-davinci-003. The temperature was set to 0.7 and the maximum amount of tokens to 512. The top probability was set to 1 and the frequency and presence penalty were set to 0.

3. Results
The performance of each model was assessed in comparison to each other. For this purpose, task 2 and 3 were ranked individually.
The performance assessment was executed manually and subjectively by analysing the data by carried out spot checks.
The following ranking was established (from highest performance to lowest performance:
Task 2:
1. GPT
2. BLOOMZ
3. ST5
4. WIKI

Task 3:
1. GPT
2. BLOOMZ

3. ST5

Regarding Task 2 of the CLEF 2023 SimpleText lab competition, the most effective models for identifying difficult concepts and explaining them were GPT, which provided the best definitions that were easy to understand at a good level, followed by BLOOMZ, which provided good definitions but were not as good as GPT. ST5 sometimes paraphrased without simplification and also defined non-complex expressions, while WIKI displayed a lot of gaps in providing definition and simplification, resulting in a score of -1.

As for Task 3, the top-performing models for producing a simplified version of a scientific text were GPT, which provided good simplifications of the text, followed by BLOOMZ, which often maintained the original wording but sometimes provided a simplified version. ST5, on the other hand, often retained the exact same expression as before and did not provide a simplified version.

4. Conclusion

The SimpleText Subtasks 2 and 3 aim to identify complex expressions and in a next step provide a simplified version of identified target word and its linguistic surrounding. However, as it became evident throughout the execution of the task the models at hand displayed a different level of performative quality. This is precipitated in the lack of precision when outlining difficult terms and subsequently simplifying them. Future research should aim to focus on the precision with which a potential AI model circles out the target expression. Along these lines, co-text related aspects that impact the specific denotation of an expression needs to be considered (e.g., acronyms that can represent multiple technical expression). Additionally, future models need to have enough explanatory data at hand that truely simplifies the target expressions without only rephrasing it.
In conclusion, future work should emphasise on precision and recall when analysing and training models.

5. Appendix

BLOOM prompt for finding difficult terms

```
prompt = "Text: The proposed algorithm and the results presented in this paper are first step towards conducting a systematic analysis of real-coded EDAs and towards developing a design theory for development of scalable and robust real-coded EDAs.\n\
Difficult terms (up to 5): real-coded EDA\n\
\n\
Text: Our results suggest that the hyperplane tiles improve the generalization capabilities of the tile coding approximator: in the hyperplane tile coding broad generalizations over the problem space result only in a soft degradation of the performance, whereas in the usual tile coding they might dramatically affect the performance.\n\
Difficult terms (up to 5): hyperplane tile coding\n\
\n\
Text: Recently, Lu and Cao proposed a new simple three-party key exchange S-3PAKE protocol and claimed that it is not only very simple and efficient, but also can survive against various known attacks.\n\
Difficult terms (up to 5): three-party key exchange, S-3PAKE\n\
\n\
Text: A proof-of-concept implementation gathers network management system data and exposes abstract maps through the Application-Layer Traffic Optimization (ALTO) protocol.\n\
Difficult terms (up to 5): Application-Layer Traffic Optimization\n\
\n\
Text: Therefore, we provide a MATLAB/Simulink benchmark suite for a ROS-based self-driving system called Autoware.\n\
Difficult terms (up to 5): MATLAB, ROS\n\
\n"

def generate_from_prompt(prompt,snt):
	try:
		p=prompt+'Text: '+snt+'\nDifficult terms (up to 5): '
		output = query({
		"inputs": p
		})
		s=re.split(r'Difficult terms \(up to 5\): ',output[0]\
['generated_text'],re.DOTALL)[-1]
		return s
	except Exception as e:
		return e
```

GPT-3 prompt for finding difficult terms

```
def gpt(input):

	openai.api_key = "..."

	response = openai.Completion.create(
	model="text-davinci-003",
	prompt="Return difficult terms from the following text:\n" + input +\	"\nDifficult terms: ",
	temperature=0.7,
	max_tokens=256,
	top_p=1,
	frequency_penalty=0,
	presence_penalty=0
)
	return response
```

BLOOMZ prompt for rating difficult terms

```
prompt = "\
Term: Diffie-Helman key exchange\n\
Difficulty: 2\n\
\n\
Term: Semantic scene classification\n\
Difficulty: 2\n\
\n\
Term: Associative Classification\n\
Difficulty: 1\n\
\n\
Term: logic qubit\n\
Difficulty: 1\n\
\n\
Term: partitioning\n\
Difficulty: 1\n\
\n\
Term: MATLAB\n\
Difficulty: 2\n\
\n\
Term: low-rate DoS attack\n\
Difficulty: 1\n\
\n\
Term: Helly property\n\
Difficulty: 2\n\
\n\
Term: tmss\n\
Difficulty: 1\n\
\n\
Term: sgRNA\n\
Difficulty: 2\n\
\n"

def generate_from_prompt(prompt,term):
        try:
                p=prompt+'Term: ' + term + "\nDifficulty: "
                output = query({
                "inputs": p,
                "parameters": {"max_new_tokens": 1}
                })
                s=re.split(r'Difficulty: ',output[0]\
        ['generated_text'],re.DOTALL)[-1]
                return s
except Exception as e:
        return e
```

GPT-3 prompt for rating difficult terms

```
def gpt(input):

        openai.api_key = "..."

        response = openai.Completion.create(
        model="text-davinci-003",
        prompt="Rate how much domain knowledge an ordinary person needs to        understand
the meaning of the following term: " + input + "\n\
        0 - The meaning can be derived or guessed.\n\
        1 - Some domain knowledge is needed.\n\
        2 - Deep domain knowledge is needed.\n\
        Return only an integer.\n\n",
        temperature=0.7,
        max_tokens=1,
        top_p=1,
        frequency_penalty=0,
        presence_penalty=0
)
        return response
```

Regular expressions to match abbreviations

```
r"^\b[\w.-]*[A-Z][\w.]*[A-Z][\w.-]*\b$"
```

BLOOMZ prompt for explaining abbreviations

```
def bloomz_abbrevs_with_snt(term, text):
        try:
                p = "The abbreviation \"" + term + "\" occurs in this text:\n\""        +    text    +
"\"\nWhat does \"" + term + "\" stand for?"
```

```
                output = query({
                "inputs": p,
                "parameters": {"max_new_tokens": 512}
                })
                s=re.split(r"\" stand for?",output[0]\
        ['generated_text'],re.DOTALL)[-1][2:]
                return s
        except Exception as e:
                return e
```

BLOOMZ prompt for explaining non-abbreviations

```
def bloomz_non_abbrevs_without_text(term):
        try:
                p = "Explain the following term in one to two sentences: \"" +            term +  "\"."
                output = query({
                "inputs": p,
                "parameters": {"max_new_tokens": 512}
                })
                s=output[0]['generated_text'][len(p):]
                return s
                return output
        except Exception as e:
                return e
```

GPT-3 prompt for explaining abbreviations

```
def gpt(input):

        openai.api_key = "..."

        response = openai.Completion.create(
        model="text-davinci-003",
        prompt="What does this abbreviation stand for?\n" + input + "\nReturn   only the result.\n\n",
        temperature=0.7,
        max_tokens=256,
        top_p=1,
        frequency_penalty=0,
        presence_penalty=0
)
        return response
```

GPT-3 prompt for explaining non-abbreviations

```
def gpt(input):

        openai.api_key = "..."

        response = openai.Completion.create(
        model="text-davinci-003",
        prompt="Return a short definition of the following term: " + input +       "\n\n",
        temperature=0.7,
        max_tokens=256,
        top_p=1,
        frequency_penalty=0,
        presence_penalty=0
)
        return response
```

BLOOMZ prompt for simplifying sentences

```
prompt = "\
original text: In an attempt to achieve the above mentioned tasks, we propose an imitation learning based, data-driven solution to UAV autonomy for navigating through city streets by learning to fly by imitating an expert pilot.\n\
simplified text: Researchers propose data-driven solutions allowing drones to autonomously navigate city streets, learning to fly by imitating an expert pilot.\n\
\n\
original text: We introduce Ignition: an end-to-end neural network architecture for training unconstrained self-driving vehicles in simulated environments.\n\
simplified text: Ignition is a neural network for training unconstrained self-driving vehicles in simulated environments.\n\
\n\
original text: It is suggested that the analysis of pharmacophore profiles could be used as an additional tool for the property-based optimization of compound selection and library design processes, thus improving the odds of success in lead discovery projects.\n\
simplified text: It is suggested that the analysis could be used as an additional tool for the optimization of compound selection and library design processes.\n\
\n\
original text: We argue that misinformation and disinformation are related yet distinct sub-categories of information.\n\
simplified text: We argue that misinformation and disinformation are related but distinct sub-categories of information.\n\
\n\
original text: Given the importance of nutrition in optimizing athletic performance, there is a concern about the effects of IFast on athletics.\n\
simplified text: Nutrition is important for the optimization of sport performance so there is a concern about the effects of intermittent fasting on performance.\n\
\n"

def bloomz(prompt, text):
        try:
                p = prompt + "original text: " + text + "\nsimplified text:"
                output = query({
                "inputs": p,
                "parameters": {"max_new_tokens": 512}
                })
                s=output[0]['generated_text'][len(p):]
                return s
        except Exception as e:
                return e
```

References

[1] Liana Ermakova, Eric SanJuan, Stéphane Huet, Olivier Augereau, Hosein Azarbonyad, and Jaap Kamps. 2023. Overview of SimpleText - CLEF-2023 track on Automatic Simplification of Scientific Texts. In Avi Arampatzis, Evangelos Kanoulas, Theodora Tsikrika, Stefanos Vrochidis, Anastasia Giachanou, Dan Li, Mohammad Aliannejadi, Michalis Vlachos, Guglielmo Faggioli, Nicola Ferro (Eds.) Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2023)

[2] Liana Ermakova, Eric SanJuan, Stéphane Huet, Olivier Augereau, Hosein Azarbonyad, and Jaap Kamps. 2023. CLEF 2023 SimpleText Track: What Happens if General Users Search Scientific Texts? In Advances in Information Retrieval: 45th European Conference on Information Retrieval, ECIR 2023, Dublin, Ireland, April 2–6, 2023, Proceedings, Part III. Springer-Verlag, Berlin, Heidelberg, 536–545. https://doi.org/10.1007/978-3-031-28241-6_62

[3] Liana Ermakova, Eric SanJuan, Jaap Kamps, Stéphane Huet, Irina Ovchinnikova, Diana Nurbakova, Sílvia Araújo, Radia Hannachi, Elise Mathurin, and Patrice Bellot. 2022. Overview of the CLEF 2022 SimpleText Lab: Automatic Simplification of Scientific Texts. In Experimental IR Meets Multilinguality, Multimodality, and Interaction: 13th International Conference of the CLEF Association, CLEF 2022, Bologna, Italy, September 5–8, 2022, Proceedings. Springer-Verlag, Berlin, Heidelberg, 470–494. https://doi.org/10.1007/978-3-031-13643-6_28

[4] Y. Dong, Z. Li, M. Rezagholizadeh, and J. C. K. Cheung, 'Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing', in ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 2020, pp. 3393–3402.

[5] O. M. Cumbicus-Pineda, I. Gonzalez-Dios, and A. Soroa, 'A Syntax-Aware Edit-based System for Text Simplification', International Conference Recent Advances in Natural Language Processing, RANLP, 2021, pp. 324–334.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, 'BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding', 2018.

[7] Y. Liu et al., 'RoBERTa: A Robustly Optimized BERT Pretraining Approach', Jul. 2019.

[8] T. B. Brown et al., 'Language models are few-shot learners', Advances in Neural Information Processing Systems, 2020.

[9] R. Campos, V. Mangaravite, A. Pasquali, A. Jorge, C Nunes, and A. Jatowt, „YAKE! Keyword extraction from single documents using multiple local features", Informaration Sciences, vol. 509, pp. 257-289, Jan. 2020.

[10] F. Boudin. (Dec. 2016). pke: an open source python-based keyphrase extraction toolkit. Presented at the 26th International Conference on Computational Linguistics: System Demonstrations, Osaka, Japan. [Online]. Available: http://aclweb.org/anthology/C16-2015

[11] R. Mihalcea, and P. Tarau, „TextRank: Bringing Order into Text," in Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing, Jul. 2004, pp. 404-411, [Online]. Available: https://aclanthology.org/W04-3252

[12] T. Le Scao et al., „BLOOM: A 176B-Parameter Open-Access Multilingual Language Model"

[13] „Hosted Inference API" Hugging Face. https://huggingface.co/docs/api-inference/index (accessed Mai 14, 2022)

[14] T. B. Brown et al., „Language Models are Few-Shot Learners“

[15] „wikipedia 1.4.0“ PyPI. https://pypi.org/project/wikipedia/ (accessed Mai 14, 2022)

[16] X. Linting et al., „mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer,“ in Proc. of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Jun. 2021, pp. 483-498, [Online]. Available: https://aclanthology.org/2021.naacl-main.41

[17] N. Muennighoff et al., „Crosslingual Generalization through Multitask Finetuning“