Team Sharingans at SimpleText: Fine-Tuned LLM based approach to Scientific Text Simplification

Syed Muhammad Ali[1,*], Hammad Sajid[1], Owais Aijaz[1], Owais Waheed[1], Faisal Alvi[1] and Abdul Samad[1]

[1]*Computer Science Program, Dhanani School of Science and Engineering, Habib University, Karachi-75290, Pakistan.*

Abstract

This paper reports Habib University's Team Sharingans' participation in the CLEF 2024 SimpleText track, which aims to simplify scientific texts for improved readability and comprehension for non-experts. Our goal is to use state-of-the-art language models for simple yet accurate explanations of scientific texts for the general public. Our solution is based on a multi-step approach utilizing the GPT-3.5 model to solve Tasks 1, 2, and 3 i.e. passage extraction, identification and explanation of difficult concepts, and summarization. Our approach for Task 1 involved sentence embedding-based vector database for narrowing the corpus, MS-Marco for document ranking, and GPT-3.5 for selecting informative passages. For Task 2, we fine-tuned the GPT-3.5 model to identify and explain difficult terms and generate explanations. For Task 3 also, we fine-tuned the GPT-3.5 model with a specific prompt to simplify given scientific abstracts and sentences. The effectiveness of our approach was assessed based on the quality of results, demonstrating the potential of advanced language models in making scientific education more accessible to the general public. Our solution proposes using fine-tuned large language models as a reliable source for scientific education.

Keywords

Large Language Models, GPT-3.5 Turbo, Elastic Search, BERT, Text simplification, SimpleText

1. Introduction

Scientific literature often presents a formidable barrier to understanding for individuals outside specialized fields due to its complexity and technical language. Recognizing this challenge, the CLEF 2024 SimpleText Lab aims to enhance accessibility by simplifying scientific texts and producing easier comprehension for a wider audience. This pursuit is divided into three tasks, each targeting different aspects of text simplification.

- Task1: What is in (or out)? Selecting passages to include in a simplified summary [1].
- Task 2: What is unclear? Difficult concept identification and explanation (definitions, abbreviation deciphering, context, applications,..) [2].
  - Task 2.1: Extract difficult keywords from the selected paragraph.
  - Task 2.2: Provide a brief definition of the extracted keywords.
- Task 3: Rewrite this! Given a query, simplify passages from scientific abstracts [3].

2. Literature Review

We review and analyze the approaches of the teams who participated in CLEF Simple Text 2023. Specifically, the approaches of teams whose models were among the top-scoring models in their respective tasks are discussed.

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France

*Corresponding author

sn07590@st.habib.edu.pk (S. M. Ali); hs07606@st.habib.edu.pk (H. Sajid); oa07610@st.habib.edu.pk (O. Aijaz); ow07611@st.habib.edu.pk (O. Waheed); faisal.alvi@sse.habib.edu.pk (F. Alvi); abdul.samad@sse.habib.edu.pk (A. Samad)

0009-0000-2941-8927 (S. M. Ali); 0009-0003-1415-2822 (H. Sajid); 0009-0002-7752-6177 (O. Aijaz); 0009-0004-4926-6624 (O. Waheed); 0000-0003-3827-7710 (F. Alvi); 0009-0009-5166-6412 (A. Samad)

For Task 01, the **Elsevier** [4] team fine-tuned the bi-encoder and cross-encoder ranking models for ranking documents given a query in order of their relevance. Specifically, they use the Dense Passage Retrieval model. The **AIIR** and **LIAAD** Labs [5] proposed five systems for this task, including cross-encoder with and without fine-tuning, Sentence-BERT bi-encoder models, and traditional IR models like TF-IDF combined with PL2.

For Task 2.1 and Task 2.2, diverse methodologies and tools were employed. The **UBO** [6] team utilized the pke package, along with statistical and graphical approaches such as YAKE!, TextRank, and Tf-Idf, to extract keywords from the provided sentences, and subsequently extracted definitions from Wikipedia for Task 2.2. The **Sinai** [7] team used the GPT-3 auto-regressive model for lexical complexity prediction. They presented an approach for identifying the most challenging terms in the text which leveraged zero-shot and few-shot learning prompts to assess term difficulty.

For Task 03, the **UBO** [6] team employed the SimpleT5 model and trained it on the datasets. Subsequently, they utilized this trained model to generate simplified text from the test dataset. They also utilized the BLOOM model, albeit requiring sample data input due to its few-shot learning nature, and similarly applied it to generate simplified text. **AIIR** and **LIAAD** [5] team, utilized OpenAI's Davinci model with a straightforward prompt for text rewriting.

3. Approaches

3.1. Task 1

For Task 01, we had

- A Corpus of DBLP abstracts. An Elastic search index and a vector database with sentence embedding scores were provided through APIs for querying the corpus.
- An input file containing input queries and their topic texts.
- A file containing the quality relevance scores of abstracts w.r.t topics on a scale of 0-2 for 25 topics and 64 queries.
- A set of files containing the topics selected from The Guardian newspaper and Tech Xplore website along with their URLs and article content.

The approaches used for this task are given:

3.1.1. MS-Marco + GPT-3.5 based re-ranking

In this approach, we utilized the vector database for querying the top 100 relevant abstracts from the corpus. To generate the query for the API, we used the query text. If the query text was a long phrase or a sentence, then the "abstracts" parameter was used in the query to search inside abstracts. In case the query text was a short phrase the "title" parameter was used. Table 1 shows examples of phrases and the generated queries.

Then, the abstracts retrieved from the search were ranked using the "msmarco-MiniLM-L12" cross encoder w.r.t the query text as well as the topic text. The query and the topic texts were concatenated together by a period and a white space ". ". The top 10 re-ranked abstracts were provided with a fine-tuned GPT-3.5 model to select the most relevant abstract with reference to query text, and then extract the most relevant passage from the selected abstract. This two-step process in shown in Table 2.

The GPT-3.5 model was fine-tuned on manually curated training data. The hyperparameters are given in Table 3.

The training data used to fine-tune GPT-3.5 comprised several examples, each having 10 manually selected abstracts as input and a manually extracted passage as the output. Finally, the runs for this task were submitted with the run id "Sharingans_Task1_marco-GPT3".

Table 1
Examples of queries generated for vector database based on the length of query text

Sentence/Phrase	Corpus Parameter	Query
Digital Assistant	title	https://guacamole.univ-avignon.fr/stvir_test?corpus=title&phrase=Digitalassistant&length=100
how AI systems, especially virtual assistants, can perpetuate gender stereotypes	abstract	https://guacamole.univ-avignon.fr/stvir_test?corpus=abstract&phrase=howAIsystems,especiallyvirtualassistants,canperpetuategenderstereotypes&length=100

Table 2
Prompts used for the two-step process to select the most relevant passage from the re-ranked abstracts

Step	Prompt
Selecting the abstract	Select the abstract which gives the most relevant definition/explanation for the following term/phrase: *(list of 10 abstracts)*
Extracting the passage	Extract the most relevant part of abstract explaining the given term/phrase in light of the topic *(topic). (abstract)*

Table 3
Experimental setup for GPT-3.5 Turbo for Task 1

Model Name	Examples	Epochs	Batch Size	learning_rate_multiplier
GPT-3.5 Turbo	30	3	1	2

3.1.2. Keyword extraction with RAKE and ColBert+GPT-3.5 based re-ranking

For this approach, we utilized RAKE [8], a keyword extraction algorithm, to identify relevant terms for querying the corpus. We provided RAKE with the topic and query text to extract relevant keywords from them. Then we used these terms to generate a query for the Elastic Search index, which in turn narrowed down the corpus to a subset of documents. This subset was further refined using the ColBERT neural ranker [9] to choose the top 10 most relevant ones, given the topic text and the query. Finally, GPT-3.5 helped in selecting the most informative and concise passage for inclusion in the summary. We did not include runs for this approach since the MS-Marco + GPT-3.5 approach worked better which has been described above.

3.2. Task 2

For Task 02, we were provided with:

- A train file, along with some manual run files, that included the fields of the “source sentences" along with their corresponding extracted terms, definitions, difficulty, and explanations with positive and negative definitions as an indicator for what an acceptable definition should look like.
- A validation file for testing the trained model with similar entries as that in the train file.
- A test dataset, having around 500 plus entries, consisting of just the source sentences for the evaluation of the model’s output.

3.2.1. GPT-3.5 Turbo based approach

To accomplish Task 02, we fine-tuned the **GPT-3.5 Turbo** model on the train dataset. GPT-3.5 Turbo is an advanced language model developed by OpenAI, part of the broader GPT-3.5 series. Due to its enhanced Natural Language understanding and generation ability, we decided to use this model specifically for this task. Table 4 represents the details of the fine-tuning of our GPT-3.5 model.

Table 4
Experimental setup for GPT-3.5 Turbo for Task 2

Model Name	Queries	Epochs	Batch Size	learning_rate_multiplier
GPT-3.5 Turbo	501	3	1	2

The effective use of 3 epochs alongside a single batch size allowed the dataset to be passed into the model only three times, which is relatively less for such a task. However, setting a batch size of one alongside a learning rate multiplier of 2 allowed a more stable adjustment of weights. We used a unit batch size with so that it has a regularizing effect to prevent our model from overfitting on the small dataset. The idea of a small batch size was to have the model learn before having to see all the data.

For this task, we observed good performance on the test set. This indicates that the mini-batch learning approach, although unconventional with a batch size of one, was effective in optimizing the model both for term extraction and for generating definitions. The small batch size and learning rate multiplier helped achieve a better generalization over the small dataset.

We passed the training dataset as a query to the GPT model, which consisted of the keywords, difficulty scores, and their definitions respectively for each sub-task to fine-tune the model. The fine-tuned model was then used to extract keywords from the source sentences, assign them difficulty scores, generate definitions, and store them in a data frame. Finally, we converted the output into a JSON file as required for the submission with the runid “Sharingans_task2.2_GPT" for both sub-tasks.

The effectiveness of this method can be attributed to the tailored approach to the specific requirements of Task 2. The model’s performance validated our decision, demonstrating that even with small batches, careful tuning can achieve desirable outcomes.

Table 5
Sample prompt to generate definition and explanation of an extracted term

Term	Difficulty	Query
Digital Assistant	m	Generate a definition of the term: “Digital Assistant" having the difficulty score: “m" and provide an explanation.

3.2.2. KeyBert, Classification, and Prompt Engineering based approach

Our second approach for Task 02 included utilizing the “KeyBert Model" [10] for keyword extraction, Random Forest Classification for assigning difficulties, and Prompt Engineering through Mistral-7B-Instruct-v0.3 Large Language Model (LLM).

The KeyBert model leverages BERT embeddings to create/extract keywords and key phrases. We utilized it to extract keywords from the source sentences. We then used Random Forest Classification on the extracted keywords with a training and test split of 80%-20%. Through the use of Mistral-7B-Instruct-v0.3 Large Language Model (LLM), we sent requests through the Hugging Face’s API to perform prompt engineering to get the required definitions as the response.

We did not submit the runs of this approach due to a major limitation of Hugging Face API that restricts the number of requests to around 500 queries which were far less than the number of terms extracted. This would result in an extremely low score in case this run was submitted.

3.3. Task 3

3.3.1. Data Description:

For Task 03, we were provided with:

- A parallel corpora of training data comprising of source sentences/abstracts along with their query texts and simplified versions.
- Test data which included source sentences (task 3.1) and source abstracts (task 3.2) and query text for each of the sentence/abstract.

3.3.2. Fine-Tuned GPT-3.5 Turbo

In this approach, we used OpenAI's GPT-3.5 model, since it has great summarizing capabilities. We first experimented with fine-tuning the GPT-3.5 model, using the training data of task 3.1 and task 3.2 all together and shuffling the sentences and abstracts randomly. Then we experimented with fine-tuning the model for Task 3.1 and Task 3.2 separately. Utilizing the EASSE scoring [11], we found that fine-tuning the model for task 3.1 and task 3.2 separately yielded slightly better results as compared to fine-tuning the model with data for both tasks altogether, especially for task 3.2. The method to train the model for task 3.1 and task 3.2 however remained the same which is discussed below.

The fine-tuning process was similar for both of the subtasks. We provided the model with a prompt to simplify the sentences/abstracts along with the sentences/abstracts, the query text, and the reference output sentences/abstracts. The hyperparameters used for fine-tuning the model are given in Table 6 and Table 7 for tasks 3.1 and 3.2 respectively.

Table 6
Experimental setup for GPT-3.5 Turbo for Task 3.1

Model Name	Queries	Epochs	Batch Size	learning_rate_multiplier
GPT-3.5 Turbo	958	3	4	2

Table 7
Experimental setup for GPT-3.5 Turbo for Task 3.2

Model Name	Queries	Epochs	Batch Size	learning_rate_multiplier
GPT-3.5 Turbo	175	3	1	2

After training the model, we provided the same prompt with the test data (sentence/abstract and query text) to generate the simplified sentences/abstracts. These simplified sentences/abstracts were then evaluated using the EASSE score and were submitted with the runid "Sharingans_task3.1_finetuned" and "Sharingans_task3.2_finetuned" for task 3.1 and task 3.2 respectively.

3.3.3. Fine-Tuned Bart Sequence-to-Sequence Model

In this approach, we utilized Meta's BART sequence to sequence pre-trained model. BART was introduced by Meta (Facebook) as a Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension [12]. Specifically, we use the "BART-large-cnn" sequence-to-sequence model using the Hugging Face Transformer library. We first tokenized the training input sentences/abstracts and the reference outputs and used them to fine-tune the model. Then we provided the model with test data to generate simplified sentences. We observed that although the model performed well in summarizing the longer sentences and abstracts, it did not simplify them in many cases. Moreover, for shorter sentences, the model generated outputs that were very similar or even the same as the original sentence. Since this model did not perform well as compared to the GPT-3.5 model, we did not include runs for this model.

3.3.4. Fine-Tuned Pegasus Sequence-to-Sequence Model

In this approach, we utilized the PEGASUS model for text simplification. PEGASUS is a pre-trained encoder-decoder model tailored specifically for abstractive text simplification [13]. We fine-tune this model via the Hugging Face Transformer library using the same approach as for BART. This model provides slightly better results than BART but still lags behind OpenAI's GPT-3.5.

4. Results and Discussion

4.1. Task 01

Table 8 shows the score of the run submitted for task 01. The scores are fairly low for our submitted approach. Specifically, we observe that the model has a very low precision. This suggests a loophole in our MSMarco-GPT-based reranking approach. We hypothesize that this is due to the manual curation of data for fine-tuning the GPT-3.5 model. We also hypothesize that models such GPT-3.5 might be limited in their ability to extract a relevant passage from the given data.

Table 8
Run scores for Task 01

runid	MRR	Precision 10	Precision 20	NDCG10	NDCG20	Bpref	MAP
Sharingans_Task1 _marco-GPT3	0.6667	0.0667	0.0333	0.1149	0.0797	0.0107	0.0107

4.2. Task 02

Our run for Task 02 retrieved a total of 1,501 keywords, assigned them difficulty scores, and later on generated their definitions and explanations. Table 9 shows our official results for our Task 02 run.

Table 9
Run scores for Task 02

runid	recall			precision	BLEU			
	overall	average	difficult_terms		n1	n2	n3	n4
Sharingans _Task2.2_GPT	0.472222	0.530246	0.544811	0.595361	0.225719	0.103904	0.0300	0.0160

The overall recall metric indicates the proportion of terms (independently from the difficulty) that were found while the precision metric indicates how accurately were the terms labeled as difficult. The ability of GPT-3.5 Turbo to effectively comprehend Natural Language tasks can be concluded from the overall scores of recall and precision indicating that our fine-tuned model was able to extract keywords and distinguish their difficulties quite satisfactorily. The BLEU scores, on the other hand, computed with n-grams equal to 1, 2, 3, and 4 lack precision on a higher number of n-grams. This may potentially be because the words chosen by our fine-tuned model to complete the definitions were not quite in line with the actual definitions used as reference, however, the idea conveyed by the definition was correct to an extent based on manual interpretation.

4.3. Task 03

Tables 10 and 11 show the scores for the run submitted for task 3.1 and task 3.2 respectively. Since an identical approach was taken for tasks 3.1 and 3.2 for these runs, they exhibit very similar scores. We observe that the fine-tuned GPT-3.5 model scores fairly high in the scoring metrics. The FKGL, BLEU and Lexical complexity score for task 3.1 and 3.2 are similar. The SARI score and compression ratio are

slightly higher in task 3.2 which indicates that documents in task 3.2 had to be modified more than the relatively smaller sentences in task 3.1 for simplification. The FKGL scores for both sub-tasks however indicate that the text can be further simplified. But this should be done without loss of information of the original text. Overall, this suggests that our approach has fairly good potential for scientific text simplification and summarization.

Table 10
Run scores for Task 3.1

runid	Count	FKGL	SARI	BLEU	Lexical Complexity	Compression ratio	Levenshtein Similarity
Sharingans_task3.1 _finetuned	578	11.39	38.61	18.18	8.70	0.83	0.77

Table 11
Run scores for Task 3.2

runid	Count	FKGL	SARI	BLEU	Lexical Complexity	Compression ratio	Levenshtein Similarity
Sharingans_task3.2 _finetuned	103	11.53	40.96	18.29	8.80	1.2	0.65

5. Conclusion

We utilized several models and techniques to solve SimpleText tasks 1, 2 and 3. For Task 1, we resorted to extracting keywords, sorting through documents, and ranking their relevance, then finally using GPT-3.5 to pick out the most relevant passages for our summary. Task 2 mostly involved fine-tuning the GPT-3.5 Turbo model to generate complex definitions. We also experimented with the KeyBert model to extract words, Random Forest classification to assign complexities and then generating definitions via prompt engineering using the MISTRAL 7-B model. However, the GPT approach turned out to be much better. Since Task 3 was text-generation based, we utilized curated data to finetune the GPT API and generate summaries. We also experimented with the Pegasus and BART model for abstractive summarization, GPT-3.5 exhibited a better performance. Conclusively, we found that out of all approaches, Open AI's GPT 3.5 language model gave the best results for task 2 and task 3. However, the pipeline for Task 01 which utilized GPT-3.5 did not perform well. Further research can be done to investigate the cause of poor performance of the Marco-GPT pipeline as well as to further improve the approaches for Tasks 2 and 3 for better simplification of scientific texts.

Acknowledgments

We would like to acknowledge the support provided by the Office Of Research (OoR) at Habib University, Karachi, Pakistan for funding this project through the internal research grant IRG-2235. We would also like to thank SimpleText@CLEF-2024 chairs for their guidance and organization.

References

[1] E. SanJuan, et al., Overview of the CLEF 2024 SimpleText task 1: Retrieve passages to include in a simplified summary, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[2] G. M. D. Nunzio, et al., Overview of the CLEF 2024 SimpleText task 2: Identify and explain difficult concepts, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[3] L. Ermakova, et al., Overview of the CLEF 2024 SimpleText task 3: Simplify scientific text, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), CEUR Workshop Proceedings, CEUR-WS.org, 2024.

[4] A. Capari, et al., Elsevier at simpletext: Passage retrieval by fine-tuning gpl on scientific documents, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), CEUR Workshop Proceedings, CEUR-WS.org, 2023.

[5] B. Mansouri, et al., Aiir and liaad labs systems for clef 2023 simpletext, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), volume 3497 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 253–253.

[6] Q. Dubreuil, Ubo team @ clef simpletext 2023 track for task 2 and 3 - using ia models to simplify scientific texts, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), CEUR Workshop Proceedings, CEUR-WS.org, 2023.

[7] J. Ortiz-Zambrano, et al., Sinai participation in simpletext task 2 at clef 2023: Gpt-3 in lexical complexity prediction for general audience, in: G. Faggioli, et al. (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), CEUR Workshop Proceedings, CEUR-WS.org, 2023.

[8] S. Rose, D. Engel, N. Cramer, W. Cowley, Automatic Keyword Extraction from Individual Documents, 2010, pp. 1 – 20. doi:10.1002/9780470689646.ch1.

[9] O. Khattab, M. Zaharia, Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020. URL: https://arxiv.org/abs/2004.12832. arXiv:2004.12832.

[10] M. Grootendorst, Maartengr/keybert: Bibtex, 2021. URL: https://doi.org/10.5281/zenodo.4461265. doi:10.5281/zenodo.4461265.

[11] F. Alva-Manchego, L. Martin, C. Scarton, L. Specia, EASSE: Easier automatic sentence simplification evaluation, in: S. Padó, R. Huang (Eds.), Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 49–54. URL: https://aclanthology.org/D19-3009. doi:10.18653/v1/D19-3009.

[12] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Annual Meeting of the Association for Computational Linguistics, 2020, pp. 7871–7880. doi:10.18653/v1/2020.acl-main.703.

[13] J. Zhang, Y. Zhao, M. Saleh, P. J. Liu, Pegasus: Pre-training with extracted gap-sentences for abstractive summarization, ArXiv abs/1912.08777 (2019). URL: https://api.semanticscholar.org/CorpusID:209405420.