University *of Ljubljana*
Faculty *of Computer and Information Science*

'; DROP TABLE TEAMS; –

Henri Sellis, Igor Sitek

Abstract

For our Natural Language Processing course group project, we are developing a conversational agent that retrieves additional information from Google Scholar documents, to increase the quality of answering questions, ensuring up-to-date outputs. To accomplish this, first a number of most relevant keywords are extracted from the user input, then corresponding queries are made to retrieve documents from Google Scholar, and finally the documents, alongside with the original user input, are passed to a Large Language Model, to generate a paraphrased response.

Keywords
Retrieval-Augmented Generation, RAG, web scraping, rank algorithms, Python, keyword extraction

Advisors: Aleš Žagar

Introduction

Large Language Models have proven to be very successful at general-knowledge topics, but a trained model cannot continuously stay up to date with the most recent academic topics. To solve this problem, we are using Retrieval-Augmented Generation to provide an LLM with the most relevant academic papers to allow the LLM to generate more relevant and accurate responses. The documents are retrieved from Google Scholar, as it is an easily accessible repository with lots of academic papers.

With the increasing number of scientific papers published across the world, it is difficult to analyze the most relevant data. Automated literature lookup based on relevant datasets might highly improve the process of conducting scientific experiments, mitigating the risk of work duplication. However, it is crucial to ensure up-to-date answers, thus relying on static datasets is not the most optimal approach. RAG-augmented query analyses alongside with web crawling and web scraping techniques seem to be promising to resolve that matter.

Methodology

The proposed methodology scheme is presented in figure 1. Each vital step is described in the following subsections of this article.

Corpus analysis

To specify our domain, we will limit our research to English papers only, eliminating the necessity of language-independent vector transitions and specific rules in different languages. The English language was chosen as it is the most popular language with the highest amount of data available, and it is characterized by low sentence sensitivity (in opposition to romance languages). That characteristic enables low-effort word stemming in text processing [1].

Keyword extraction

In order to be able to query for relevant literature, keywords need to be extracted from user input. These keywords can be used as a Google Scholar query.

Work [2] is a review of existing methods for keyword extraction, containing summaries of different supervised and unsupervised algorithms. Upon these, Neural Networks seem to be used mainly. On the other hand, ready-to-use implementations of RAKE and RANK algorithms (like TextRank [3] - https://github.com/davidadamojr/TextRank or TopicRank [4] - https://github.com/Aayushpatel007/topicrankpy) seem to be useful enough for our case. Due to simplicity of use, the KeyBERT model [5] was chosen. It uses BERT-embeddings and simple cosine similarity to find the most central sub-phrases in a text.

Web scraping

Keywords extracted from user input are combined into a suitable Google Scholar query, in order to find the most relevant academic documents which might include useful context for the chatbot for answering the user's question. Querying Google Scholar was implemented using the Scholarly [6]

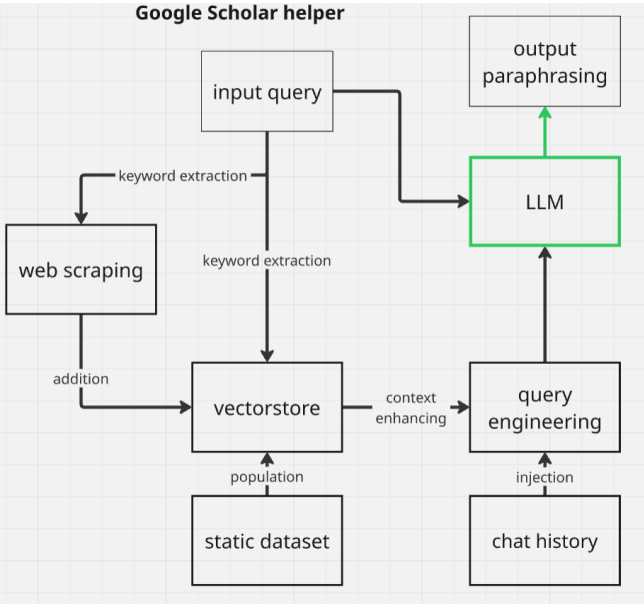

Figure 1. System scheme High-level overview of chat-bot system proposed in this paper.

library.

However, many of the documents findable in Google Scholar are not directly downloadable. To solve this issue, we implemented a secondary functionality for downloading the found papers from SciHub [7], when they are not directly available from Google Scholar.

Document retrieval

After online paper scraping, all documents are split into chunks and kept in a FAISS *vectorstore*. Based on the similarity indicator within the configured threshold, the most relevant documents are provided for further use. These are then filtered out using a re-ranking algorithm for double-check safety and context sanity.

Static dataset

In order to avoid starting from a completely empty FAISS vectorstore, a dataset of science papers [8] was processed and added into the vectorstore, in the same way as described in the previous paragraph.

The dataset consists of zip folders with two sub-folders, one containing whole papers in TXT format, and one with manually gathered keywords and key-phrases. These files are matched based on a common key - included in the filename.

Data combining and paraphrasing

All the relevant documents (found in the static dataset and downloaded from the web) are combined with user input and passed into a pre-trained LLM model. Since the focus of this paper is on the RAG system itself, the choice of any specific LLM is not central - The "Mistral-7B-Instruct-v0.2" model was selected because of its popularity. Additionally, prompt engineering was used to provide the LLM specific instructions on how to handle the specific input combined from the user's question and the documents.

Presentation

For presentation purposes, a simple web application that allows users to use the chatbot was developed. It is a Python application which primarily uses the FastAPI library, and also has a front-end user interface. The user interface can be seen on Figure 2: it has functionality for switching between different chat windows, and in each chat, the user can enter their question which is passed to the chatbot. In addition to the chatbot's textual answer, the user interface also displays the titles of the source documents that the chatbot used for generating the answer (as part of the RAG process described in the previous section). The user can easily download the source documents by clicking on the title links. This functionality enhances the idea of an academic research helper by giving users the chance to delve deeper into the topic themselves and form their own opinion on whether to agree with the chatbot's answer or not.

Evaluation

In addition to manually evaluating the quality of the chatbot's answers, an evaluation dataset of science-related questions [9] was used (https://huggingface.co/datasets/allenai/sciq). A snapshot of it can be seen on figure 3: for each question, it includes the correct answer, supporting reasoning for the correct answer and three distractors.

A subset of 20 questions was selected for evaluating the RAG system (limited by GPU computing time). The chatbot was asked each question in both online and offline mode (with vs without having access to the RAG system). To compare the chatbot's answer's similarity to the ground truth correct answer, as well as to the supporting reasoning text and the distractors, different metrics were calculated:

- **Exact match**: simply checking if the chatbot's answer text contains the ground truth correct answer
- **Fuzzy score**: Fuzzy string matching, uses Levenshtein distance. Provided matching score in the range from 0 to 100.
- **Embedding similarity**: The chatbot's answer, the correct answer and the distractors are converted to text embeddings, using sentence transformers (specifically, the "all-MiniLM-L6-v2" model was used), and cosine similarity between the chatbot's answer's embedding and the embeddings of ground truth and supporting text are calculated. Additionally, it was checked whether the chatbot's answer's embedding is more similar to any of the distractors' embeddings than to the correct answer.

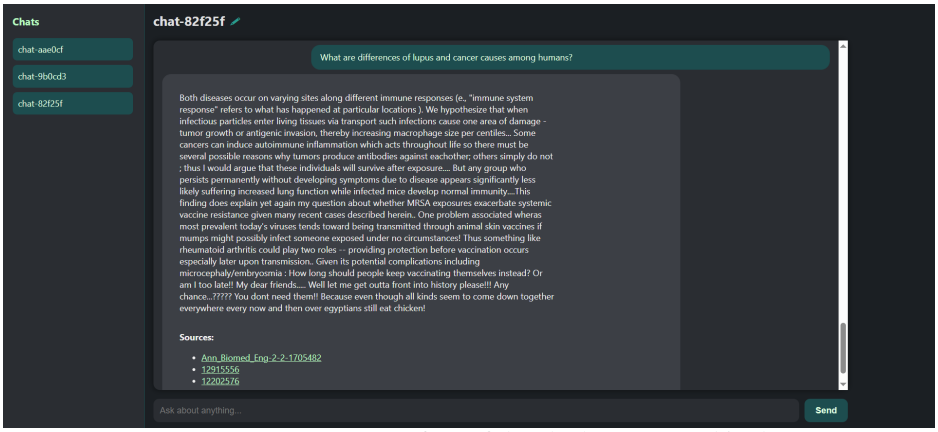

Figure 2. Front-end User Interface of the chatbot proposed in the paper.

question string · lengths	distractor3 string · lengths	distractor1 string · lengths	distractor2 string · lengths	correct_answer string · lengths	support string · lengths
53⇔92 52.7%	1⇔8 33.3%	1⇔7 23.4%	1⇔8 33.2%	8⇔15 47.7%	0⇔356 58.3%
Changes from a less-ordered stat…	endothermic	unbalanced	reactive	exothermic	Summary Changes of state are exampl changes, or phase transitions. All
What is the least dangerous…	zeta decay	beta decay	gamma decay	alpha decay	All radioactive decay is dangerous things, but alpha decay is the leas
Kilauea in hawaii is the world's…	magma	greenhouse gases	carbon and smog	smoke and ash	Example 3.5 Calculating Projectile Projectile Kilauea in Hawaii is the
When a meteoroid reaches earth, what is the remaining object called?	orbit	comet	meteor	meteorite	Meteoroids are smaller than asteroi the size of boulders to the size of When meteoroids enter Earth's atmos vaporize, creating a trail of glowi meteor. If any of the meteoroid rea remaining object is called a meteor

Figure 3. Science questions dataset used for evaluation.

This same evaluation process was done for three different configurations of reranking top k and minimal score:

- top k = 2, min score = 0.3
- top k = 3, min score = 0.5
- top k = 5, min score = 0.6

These values are similar to those used in FAISS *vector-store* - where a maximum of top *k* documents is returned from a similarity lookup. Each returned document has to be similar enough in order to be considered relevant - it is the similarity threshold in FAISS implementation, and the min score in the reranking algorithm. The pre-trained reranking model was used in our research without further fine-tuning.

The results can be seen on figures 4 and 5. The difference between online mode and offline mode (with vs without RAG system) is relatively minor. However, a clear trend is noticeable across all metrics: the first configuration (top k = 2, min score = 0.3) underperforms the base offline mode, the second configuration (top k = 3, min score = 0.5) is roughly equal, and the third configuration (top k = 5, min score = 0.6) has higher scores than the offline mode.

Summary

We have successfully implemented a RAG-based chatbot specialized in research papers thanks to updating the LLM context with relevant documents. Each output contains relevant sources used for message generation, allowing LLM reasoning transparency, and a possibility to further delve into the research and readings of relevant documents.

The web application was developed to enhance the user experience of chat history and user-friendly output. Chat history is passed to the LLM to provide relevant context, too. This enhances the quality of LLM answers.

Proper query-engineering techniques were implemented as well, mitigating the risk of model hallucinations, stressing its clarity and honesty when answering questions, which is crucial for research-intended querying, especially in STEM areas.

Conclusions

After analyzing the output of evaluation, the online version seems to perform better, thanks to downloading the relevant documents before generating an answer. Interestingly, its similarity to correct answers worsens, tending to lose attention towards common distractors. We believe that it is due to the

Configuration	Online?	Exact Match	Fuzzy Score	Similarity to Correct Answer	Similarity to Support Text	Better Than Distractors
top_k_2-min_score_0.3	False	0.45	70.85	0.41705	0.66835	0.70
top_k_2-min_score_0.3	True	0.20	63.35	0.38405	0.63035	0.60
top_k_3-min_score_0.5	False	0.35	71.45	0.39145	0.64125	0.70
top_k_3-min_score_0.5	True	0.25	69.30	0.38395	0.65220	0.65
top_k_5-min_score_0.6	False	0.40	63.70	0.36245	0.61200	0.65
top_k_5-min_score_0.6	True	0.35	73.80	0.41520	0.64740	0.70

Figure 4. Evaluation metrics averaged across all the question. "Better Than Distractors" column shows the average of whether the chatbot's answer is most similar to the correct answer (value 1) or not (value 0).

fact that when downloading a small amount of documents (small values of k) a single piece can easily overwhelm the context, resulting in that behavior. This result may be correlated with the allowance of low-relevant documents (small values of min score) and further testing would be necessary to testify it.

References

[1] Peter D Turney. Learning algorithms for keyphrase extraction. *Information retrieval*, 2:303–336, 2000.

[2] Sifatullah Siddiqi and Aditi Sharan. Keyword and keyphrase extraction techniques: a literature review. *International Journal of Computer Applications*, 109(2), 2015.

[3] Suhan Pan, Zhiqiang Li, and Juan Dai. An improved textrank keywords extraction algorithm. In *Proceedings of the ACM Turing Celebration Conference - China*, ACM TURC '19, New York, NY, USA, 2019. Association for Computing Machinery.

[4] Adrien Bougouin, Florian Boudin, and Béatrice Daille. Topicrank: Graph-based topic ranking for keyphrase extraction. In *International joint conference on natural language processing (IJCNLP)*, pages 543–551, 2013.

[5] Maarten Grootendorst. Keybert: Minimal keyword extraction with bert., 2020.

[6] Steven A. Cholewiak, Panos Ipeirotis, Victor Silva, and Arun Kannawadi. SCHOLARLY: Simple access to Google Scholar authors and citation using Python, 2021.

[7] Sci-Hub. Sci-Hub: Free and unrestricted access to all scientific knowledge. https://sci-hub.se, 2025. Accessed: 2025-05-28.

[8] rncampos and Vítor Mangaravite. Datasets of automatic keyphrase extraction. https://github.com/INESCTEC/KeywordExtractor-Datasets, 2025. Accessed: 2025-05-28.

[9] Matt Gardner Johannes Welbl, Nelson F. Liu. Crowdsourcing multiple choice science questions. In *SciQ*, 2017.

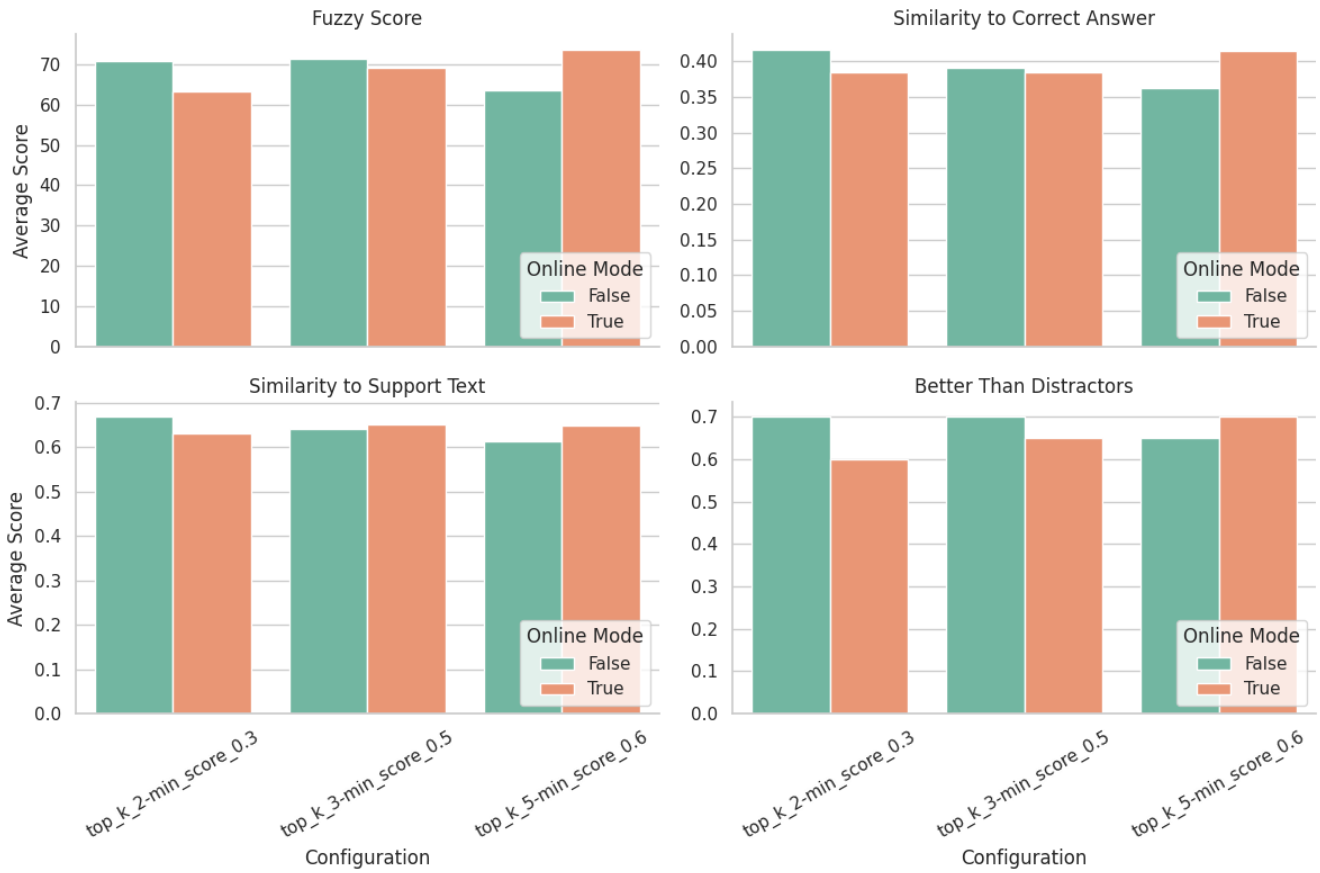

Figure 5. Evaluation metrics averaged across all the questions.