Retrieval augmented generation to improve text summarization of biomedical research articles

Andrew Ton, Yuxuan Cheng
Department of Physics
Yale University
New Haven, CT, 06511

Code is available on Github [1].

1 Introduction

Publications in biomedical research journals convey important information and research findings regarding public and personal health. The general public is typically interested in learning about these results, but publications in such journals can use highly technical language that is hard for a non-technical audience to understand. We aim to assess the state of the art on producing meaningful lay summaries of technical papers. Our approach will be to use retrieval to augment pre-trained large language models (LLMs) to output easily understood lay summaries of biomedical research papers.

This is a difficult problem because the expected output text should keep all relevant information, omit all non-relevant information, remove repeated or redundant information, extract the main idea of the input text, and write in simpler language than the input [7]. Previous approaches take advantage of massive amounts of available online information to train sophisticated statistical models, including extractive methods and abstractive methods. Before LLMs, much of previous text summarization progress used extractive methods, though there was also significant progress in abstractive methods e.g. sequence to sequence (seq2seq) models [1]. The current state of the art in summarization is abstractive, and uses LLMs to achieve better-than-human performance in terms of fluency, and match or exceed the factual consistency of human-generated summaries in general tasks [6]. However, there is still room for improvement when applying LLMs to downstream tasks such as in summarizing research papers.

Some of the problems of applying LLMs to research summarization include difficulties processing the long length of research papers, the lack of domain-specific understanding, and challenges in selection and augmentation of training data [2]. To solve these problems, we will integrate Retrieval-Augmented Generation (RAG) with LLMs to enable query on local database. This approach takes advantage of the high fluency and generalizability of pre-trained language models, while emphasizing the contribution of in-domain knowledge by retrieving relevant information from private data source.

2 Large Language Models

2.1 Auto-regressive Language Models and Transformers

Auto-regressive language models are a type of language model that predicts the next word in a sequence based on the previous words. Examples of such models are RNNs, LSTMs and Transformer decoders. Transformer decoders, which are heavily used in LLMs, are inherently autoregressive during inference time, but differ from RNNs by adding attention mechanisms, which allow each word in a sequence to dynamically adjust its representation by considering the relevance of all other words in the sequence. This means that LLMs can understand the context of each word more accurately,

[1] Code is available at https://github.com/YUXUANCHENG/477RAG

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

which is crucial for understanding the meaning of sentences where the significance of words can change based on their context. Transformers also have improved long-term memory since attention mechanisms directly compute relationships between all pairs of inputs, no matter how far apart they are in the sequence. This enables the model to capture long-range dependencies more effectively.

2.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) combines the powers of pre-trained language models with information retrieval to enhance the generation of text. For every input query, relevant information is retrieved from a local data source. This information is then provided to LLMs as additional context for generating more accurate, informative, and contextually relevant responses. This approach allows models to incorporate vast amounts of information beyond what they were trained on, enabling them to provide responses based on a broader range of knowledge and context.

3 Methodology

To enable our LLMs to take advantage of RAG, we took advantage of an existing RAG pipeline that we adapted for our purposes. For more information on this pipeline, see the public repo https://github.com/PromtEngineer/localGPT. This pipeline functions to preprocess the biomedical research articles and store them in a vector database. The documents are loaded and split recursively by character into chunks using Langchain, then loaded into a vector store (Chroma DB) using an instruction fine-tuned embedding model that can be found on HuggingFace: https://huggingface.co/hkunlp/instructor-large. We set this vector store to persist on physical memory, such that we call the *ingest.py* function to preprocess, chunk, embed, and store our input documents in memory. Vector stores can easily be converted into a document retrieval system within Langchain, which enables us to use the Chroma DB from *ingest.py* as a retriever in the future. Once the vector store has been saved, it is loaded as a retriever whenever accessing the query pipeline. The retriever has been set up to use the embedding to compare and return the most relevant documents to a given input string query.

Our base model is a Llama-2 model that can be found at https://huggingface.co/TheBloke/Llama-2-7B-Chat-GGUF. This is a Llama-2 model with 7B parameters with 4 bit quantization according to the Q4_K_M method. We ran inference on a NVIDIA RTX 3070.

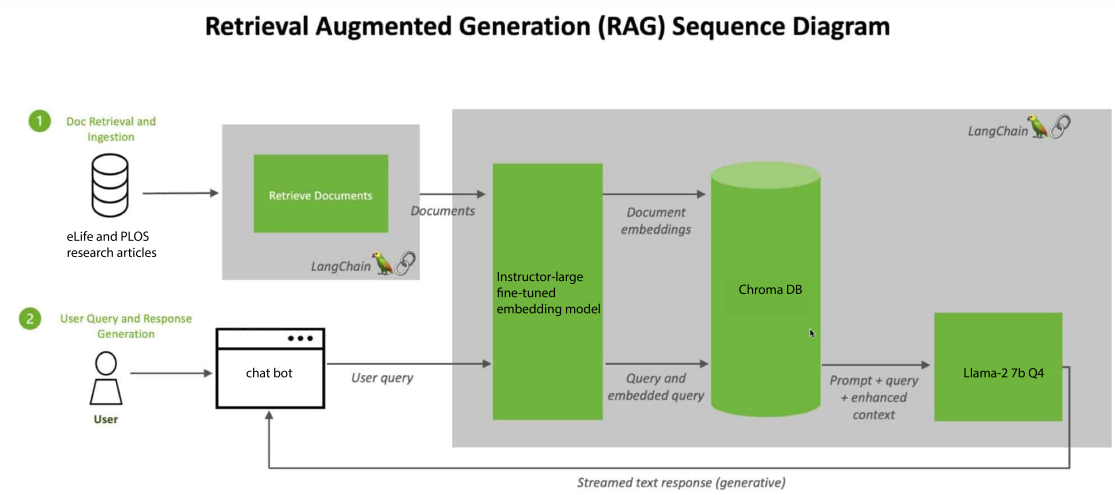

Figure 1: Schematic overview of our RAG pipeline. Documents are split, chunked, and digested in stage 1, then stored in a vector database called Chroma DB using the Instructor-large embedding model. In stage 2, the user queries a chat bot, which matches the query to the most similar documents (according to the embedding model) stored in the vector database. The embedded user prompt, the query, and the retrieval augmented query are passed to Llama-2 which generates a response to the user. Figure adapted with minor changes from NVIDIA blog [5].

4 Experiments

4.1 Dataset

[2] The dataset used for lay summarization has been published in [3], and consists of data corresponding to two major biomedical publications: Public Library of Science (PLoS) and eLife. The dataset is comprised of research articles, technical abstracts for each article, and an expert-generated lay summary for each article. Specifically, the dataset has 24,773/1,376 entries for training/validation from PLoS and 4,346/241 entries from eLife. The test data has 142 entries and does not have corresponding expert lay summaries.

There is a difference in the style of article and lay summary between PLoS and eLife. PLoS article summaries are contributed by the article's authors, and articles are pulled from 5 different journals covering subfields of biology. Summaries from eLife are contributed by the journal editors, and articles cover all of life science and medicine.

4.2 Evaluation metrics

The performance of the models will be evaluated for relevance and readability.

- Relevance: ROUGE (1, 2, and L), and BERTScore
- Readability: Flesch-Kincaid Grade Level (FKGL), Dale-Chall Readability Score (DCRS)

A quick summary of the metrics follows. ROUGE-N counts the number of overlapping n-grams between the summary and the text, and ROUGE-L counts the length of the longest common subsequence between the summary and the text[4]. BERTscore uses BERT to generate a similarity score between embeddings of tokens in the summary and the text [8]. FKGL uses the number of sentences, syllables, and words in the text. DCRS uses the average sentence length and the usage frequency of the 3000 most common English words.

5 Results

We experimented with augmenting Llama-2 with RAG to compare against GPT-3.5 on the PLoS and eLife articles dataset. For compute reasons, we are selecting 10 random articles (5 PloS and 5 eLife), and the results are summarized in Table 1. All of the listed results are computed with zero-shot learning within the RAG pipeline.

5.1 RAG improves relevance and readability

We first collect responses from both Llama-RAG and GPT 3.5 using zero shot prompts such as "Provide lay summary for the following paper ...(paper title)". We then manually analyse the responses and check for signs of hallucination. In Fig.2 (a) we show the first three sentences of an example response from Llama-RAG. We find the response provides general-reader friendly lay summary that avoids jargon and complex words. More importantly, the response provides factually correct information such as "activity in different parts of the frontal cortex while they performed a task that required them to make decisions based on conflicting sensory inputs", which is the main experiment carried out in the paper. However, as shown in Fig.2 (b), GPT 3.5 does not mention this key information. Instead, the response from GPT 3.5 is too generic, missing important details (the first sentence), and provides wrong information that is inconsistent with the paper (the second and third sentences). This behavior of GPT 3.5 is known as hallucination: an LLM generates a response that is factually incorrect, nonsensical, or disconnected from the input prompt. In this case, GPT 3.5 training data set does not include the specific paper we ask it to summarize. Therefore GPT naturally won't be able to generate accurate summary due to lack of information, and hallucinates to output grammatically crrect but irrelevant responses. In contrast, we find Llama-RAG to be less prone to hallucinations. This is because RAG retrieves relevant paragraphs/text chunks from a large external knowledge source, in this case scientific papers. By directly incorporating knowledge from

[2]This year's competition website can be found here: https://www.codabench.org/competitions/1920/

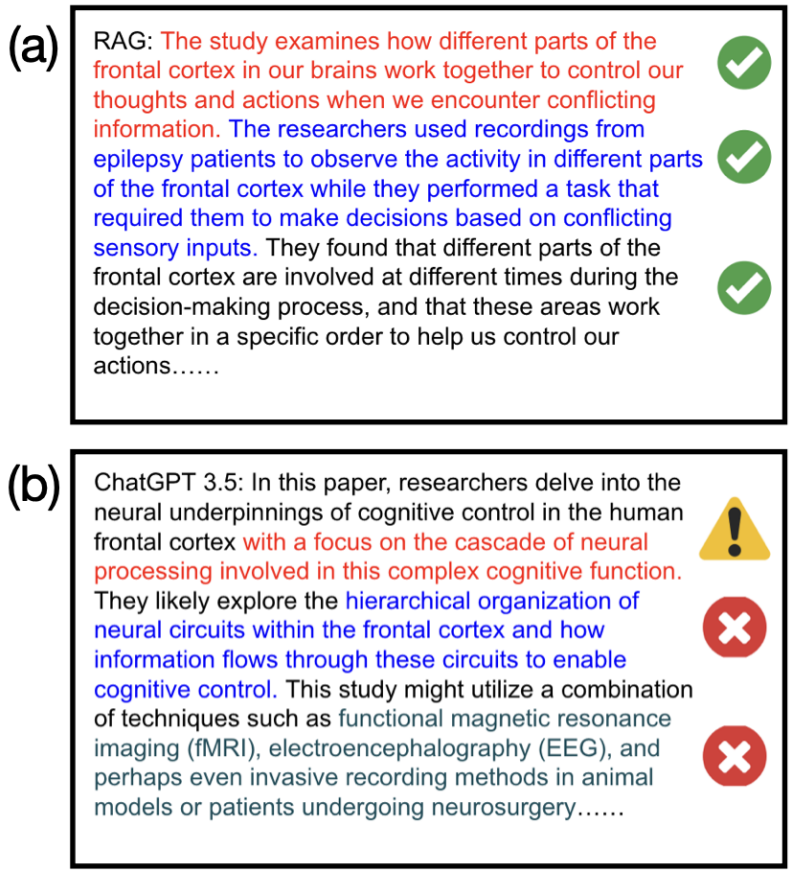

Figure 2: The first three sentences of responses from (a) Llamma-RAG and (b) GPT 3.5 using prompt "Provide lay summary for paper 'Cascade of neural processing orchestrates cognitive control in human frontal cortex'". All three sentences colored in (a) are concise summaries of the paper and are factually correct. In contrast, colored information in (b) is inconsistent with the actual paper, therefore show signs of hallucination.

the paper, Llama-RAG gains a better understanding of the context and can generate more accurate summarizations.

We also quantitatively compare performance of Llama-RAG and GPT3.5 via metrics mentioned in the previous section. In Table 1 we show metric scores for both models and find Llama-RAG outperforms GPT3.5 models. We find Llama-RAG model has higher ROUGE-1, ROUGE-2, ROUGE-L, and BERTScore scores, indicating more relevant summaries. This indicates that using RAG allows models to achieve better performance without any additional training.

We note that GPT3.5 has higher KFGL and DCRS scores, indicating better readability comparing to Llama based model. This is not surprising since GPT3.5 has more than 100 billion parameters, whereas the Llama model we used is a highly quantized 4 bit 7 billion parameter model. Larger models often have more parameters and greater complexity, allowing them to capture finer nuances of language. This increased capacity can enhance readability by enabling the model to produce text that is grammatically correct and stylistically coherent.

6 Conclusion

For this project, we used retrieval augmented generation in order to improve the ability of Llama-2 to access domain knowledge for better lay summarization of biomedical research papers. We evaluated

Model	ROUGE-1	ROUGE-2	ROUGE-L	BERTScore	FKGL	DCRS
Llama-2 + RAG	0.50	0.32	0.48	0.88	12.2	9.53
GPT3.5	0.34	0.09	0.33	0.84	13.9	10.23

Table 1: Score metrics for LLama2 with RAG and GPT3.5. Scores consist of ROUGE-1, ROUGE-2, ROUGE-L, BERTScore, FKGL, DCRS which are defined in the Evaluation metrics section.

Llama-2 with RAG and note that it outperformed GPT3.5 in terms of relevance. This is a significant outcome, given that Llama-RAG has 7 billion parameters whereas GPT3.5 has 100 billion parameters. This means that without additional compute for training, a smaller model was able to outperform a larger model on a downstream task. Retrieval augmented generation methods have a lot of promise for improving language models' performances without drastically increasing compute requirements, as we have demonstrated on this biomedical research dataset.

For future directions, we are interested in evaluating metrics that capture the factuality of the summaries and in using greater compute to generate summaries for the entire biomedical research article dataset. Due to time and compute limits, we decided to stick to around 10 articles. A further improvement to our study would be to incorporate few-shot learning and additional prompt engineering. More benchmarking would also improve our study, such as using other models with a variety of parameter sizes and using RAG to augment each of them.

In terms of broader impacts, we are improving the accessibility of important, interesting research by being able to better summarize biomedical research papers. This will make it easier for the public to understand and engage with impactful science. The general population will find it easier to understand the current landscape of medical science, which can improve public interest in funding scientific work, accelerate the adoption of good health practices, and reduce medical misinformation due to faulty interpretation of scientific articles.

Contribution statement: Both Ton and Cheng worked on the implementation and write-up of this project synchronously and asynchronously, and consider the amount of work done by each to be roughly equal. Ton contributed Figure 1, Cheng contributed Figure 2, and both contributed to Table 1.

7 Reproducibility checklist

Model Description, algorithm, Mathematical Setting:

✓Include a thorough explanation of the model/approach or the mathematical framework

Source Code Accessibility:

✓Provide a link to the source code on github.

✓Ensure the code is well-documented

✓Ensure that the github repo has instructions for setting up the experimental environment.

✓Clearly list all dependencies and external libraries used, along with their versions.

Computing Infrastructure:

✓Detail the computing environment, including hardware (GPUs, CPUs) and software (operating system, machine learning frameworks) specifications used for your results.

✓Mention any specific configurations or optimizations used.

Dataset Description:

✓Clearly describe the datasets used, including sources, preprocessing steps, and any modifications.

✓If possible, provide links to the datasets or instructions on how to obtain them.

Hyperparameters and Tuning Process:

✓Detail the hyperparameters used and the process for selecting them. (Example: The model was fine-tuned using a batch size of 16, learning rate of 1e-5, and trained on 1000 steps with 100 steps of learning rate linear warmup with linear decay)

Evaluation Metrics and Statistical Methods:

✓Clearly define the evaluation metrics and statistical methods used in assessing the model.

Experimental Results:

✓Present a comprehensive set of results, including performance on test sets and/or any relevant validation sets.

✓Include comparisons with baseline models and state-of-the-art, where applicable.

Limitations and future work:

✓Include a discussion of the limitations of your approach and potential areas for future work.

References

[1] L. Abualigah, M. Q. Bashabsheh, H. Alabool, and M. Shehab. Text summarization: a brief review. *Recent Advances in NLP: the case of Arabic language*, pages 1–15, 2020.

[2] T. Goldsack, Z. Luo, Q. Xie, C. Scarton, M. Shardlow, S. Ananiadou, and C. Lin. Overview of the biolaysumm 2023 shared task on lay summarization of biomedical research articles. *arXiv preprint arXiv:2309.17332*, 2023.

[3] T. Goldsack, Z. Zhang, C. Lin, and C. Scarton. Making science simple: corpora for the lay summarisation of scientific literature. *arXiv preprint arXiv:2210.09932*, 2022.

[4] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[5] R. Merrit. What is retrieval-augmented generation aka rag? *NVIDIA Blog*, 2023.

[6] X. Pu, M. Gao, and X. Wan. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*, 2023.

[7] C. Wang, X. Liu, Y. Yue, X. Tang, T. Zhang, C. Jiayang, Y. Yao, W. Gao, X. Hu, Z. Qi, et al. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*, 2023.

[8] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.