17-630 Prompt Engineering

Assignment: Prompt Augmentation

Prompt augmentation includes various techniques in which prompts are augmented with additional context to complete a task. This context may include content retrieved from databases or websites, or content returned from function calls. Because a model's parametric "knowledge" is limited to the model pre-training data, tasks may be outside the model's training distribution and thus be subject to hallucinations. In addition, without proper sourcing, model responses may be impossible to attribute to source texts. Attribution to source texts can be used to improve the credibility of generated responses, because the source text can be assessed for trustworthiness or factuality and because the response can be compared to the text for faithfulness.

In this assignment, students will develop a simple Retrieval Augmented Generation (RAG) system to answer questions about research papers. Because research is quickly advancing, and questions about research can be narrowly focused and deeply technical, users of language models cannot rely on a model's parametric knowledge alone to answer these questions. Moreover, the ability to trace answers to specific sentences within the papers allows the user to dig deeper in their understanding of both the question and answer.

In this assignment, students will explore ways to pre-process and index data for retrieval, to answer a user query by summarizing retrieved content and to evaluate the resulting summarizations. Students will index their data using an open-source vector database.

Recommended Readings

- Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," (NeurIPS 2020)
- Edge et al., "From Local to Global: A Graph RAG Approach to Query-Focused Summarization." (arXiv:2404.16130 2024)
- Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey" (arXiv:2312.10997v 2024)

Learning Objectives

- Pre-process and index data to create a RAG system using a vector database.
- Design and evaluate basic prompt augmentation experiments.

Assignment Deliverables

This is an individual assignment. To complete the assignment, perform the following steps: (1) parse the dataset provided, which consists of an arbitrary number of PDF files containing recent research papers. BibTEX files, which contain the paper's title, authors and publication information have been included but are not necessary to use. (2) Optionally, pre-process the data (see further guidance). (3) Chunk and index the data, wherein the indices should be stored in the vector database for comparison to user queries. For this assignment, students will generate questions from each chunk to be used as indices. (4) Build the vector database using the generated questions while maintaining a mapping between each generated question and the chunk from which it was generated. (5) Evaluate the RAG system using the user queries provided, plus *nine of your own queries* to test the limits of the system. Your queries should include: three queries with verbatim quotable answers, three queries with paraphrased answers and three queries that cannot obviously be answered. When evaluating your solution, separately report the precision, recall and F1 scores using BertScore by evaluating your own queries against the provided queries.

Develop a Jupyter Notebook that includes sections for building the RAG system and a section for conducting the. Students are encouraged to use the notebook provided in the assignment package.

Submit a single ZIP archive with the following files and exact filenames:

- notebook.ipynb – your Jupyter Notebook containing your code used to conduct your experiment. Please document major sections of your notebook.
- test-queries.json – the queries and answers that you created manually. The format of this file should match the dev-queries.json file: `[{"query": …, "answer": …}, {…}]`
- questions.txt – Answers to the questions provided in the package.

Further Guidance

- We recommend using chromadb as your vector database, which is open source and available here: https://docs.trychroma.com/
- We require you to use BertScore to evaluate RAG responses against the provided, ground-truth answers.
- *Pre-processing*. Raw research paper content extracted from PDF files contains information irrelevant to the task, such as the title, authors, table of contents and references. You can choose to clean the data by extracting the entire text from each PDF file and then writing the text to a corresponding text file. Next, you can manually edit each file by removing the unwanted content. For a small number of files, this is generally faster than designing, developing and evaluating an automated cleaning procedure. After reading the text file content, you will chunk the cleaned content later in a separate step, so do not feel the need to clean and chunk at the same time.
- When choosing a chunk size, consider the span of text needed to answer a query. Chunks that are too small will not include a sufficient number of sentences to convey an idea needed to answer a query. Chunks that are too large will require you to limit the number of relevant chunks retrieved that you can include in the input context. Experiment with different chunk sizes by inspecting the chunks and draw your own conclusion, consider whether you want your chunks to overlap, etc.
- The vector database serves as a way to index text that is semantically similar to a query, in this case a question about the research papers posed by a hypothetical user. In this assignment, we recommend prompting an LLM to generate questions that can be answered by each chunk; chunks may yield more than one question depending on their size. Next, create a mapping between the chunks and the generated questions. The generated questions will serve as your “documents” to be

indexed by your vector database. When querying the database, the query will be compared to the generated questions and the most similar generated questions will be returned. Next, you will use the retrieved questions to select the chunks associated with the retrieved questions from the mapping that you separately maintain. These chunks should be used in a subsequent question-answering prompt where the chunks are provided as “augmentation” in your prompt. We recommend using the “ids” field from chromadb as a key to maintain the mapping between generated questions and the chunks from which they were generated.

- When prompting the language model to answer a query from the augmented context, include an instruction to handle cases where the answer is not present in the augmentation. We recommend instructing the model to respond with a special token (e.g., “IDK”), which can easily be checked against test queries without an answer.
- To evaluate the RAG system, develop your own queries and ground-truth answers. Skim a few research papers on your own and identify queries that you can ask about the papers and the content associated with the answer. Hand-write your queries and the ground-truth answers. Include three queries and their answers that are copied verbatim from the papers, plus three queries and answers that must be paraphrased from multiple sentences, and finally three queries that cannot obviously be answered. For the last three queries, write the answer as “IDK”.

Evaluation Criteria

- Correctness of the implementation of the experiment and JSON file.
- Thoughtfulness of the answers to the questions in questions.txt, including examples from your work to illustrate your answers.