RAG

MCDA5511 Assignment #2

Due: Mar 7 before midnight

Set Up

Retrieval Augmented Generation (RAG) is very popular in industry because it can make business processes more efficient by enabling "chat with docs" functionality. For example, call center staff can be equipped with a RAG system to provide them with instructions on how to action complicated client requests, reducing training costs and fulfillment time. For roles that have a research component, RAG can be used to draft reports for analysts.

In this assignment you will develop your model criticism skills by assessing a RAG system. In practice, these systems typically require considerable customization to work well. However, the purpose of the assignment is NOT to build the perfect RAG, but rather to assess model performance. If your RAG performs poorly, you will still receive full marks if you properly identify and describe its failure modes.

Select a document repository for your project. Choose something that is a manageable size, covers diverse topics that you can ask questions about, and is interesting to you. IMPORTANT: You can simplify the assignment by choosing a dataset that contains relatively succinct documents (so that chunk size is not an issue) and has topic labels (so that you don't have to add a topic modeling step). For example, arXiv Paper Abstracts is a small collection of paper titles and abstracts with a category label, allowing you to ask questions like "What are some recent developments in quantum computing?"

You will submit your answers in a Jupyter notebook (*.ipynb* file). Use markdown cells to create section titles, python cells for code, and markdown cells to answer written questions. Make sure that the layout of your code and written answers is easy to follow.

Homework

(1) As always, begin by exploring your data. Note that at this step you can curate your dataset by sampling from it. Write up a summary of what you discover, making sure to include the following:

- What sampling or processing did you do (if any)?
- Basic summary statistics such as the distribution of document length and vocabulary.
- What topics are covered in this dataset?
- What are the relevant frequencies of the topics? (This will be useful for subsequent questions.)

(2) Construct your RAG model. Recommended components for the simplest possible implementation are provided below, but you are free to choose whatever components you wish. If you choose your own, write a few sentences explaining your rationale.

- Embeddings: `BAAI/bge-small-en`
- Embedding generation: `sentence-transformers`
- Retrieval: Retrieve top `k` most similar documents to the query using cosine similarity. Start with a small value like `k=3` and amend later if necessary. If you chose longer documents for your dataset, you may also need to experiment with chunk size.
- Generation: Generate an answer based on a prompt, the user query, and the retrieved documents. Informally experiment with a few different prompts to see what works best. A small generative model that you can use is `FLAN-T5`, but for this component in particular, it would be

a good idea to experiment with larger models to see if they give you better results. But again, remember that the focus of this assignment is not to build the best performing RAG.

As an alternative to the above implementation, you may use an AutoRAG tool if you prefer.

(3) Construct a dataset of Q-A pairs by creating a set of questions covering a couple of different topics, and for each question store the retrieved documents, their topic labels, and the generated response. The more Q-A pairs you generate the better – 15 at a minimum. Try to find questions that demonstrate the model working both well and poorly. One thing you might try to produce poor performance is to ask questions about topics that are not present in the data.

(4) Manually review the documents that were retrieved for each question. Label each retrieved document as correct or incorrect. On a best-efforts basis, identify whether any documents should have been retrieved that were not and label those as well. Measure precision, recall, F1-score, and accuracy and comment on the results. Do you detect any differences in performance between topics that are well-represented in the data compared with those that are not? (This last question may not be feasible if the number of Q-A pairs is small, in which case you can just explain that.)

In your own words, explain what cosine similarity is, how it is used for retrieval, and what its limitations are. Can you find an instance where retrieval failed due to one or more of these limitations?

(5) Now manually review the generated responses by comparing them against the retrieved documents. Can you identify any instances where the model hallucinated (i.e. produced information that was not in the retrieved documents)? Comment on the overall quality of the generated responses. (If you wish you can write a ground truth response for each question and use cosine similarity or another metric to quantitatively assess the generated responses. If you choose to do this, comment on the limitations of the approach, similar as you did in question 4.)

(6) It is common practice to use an "LLM as a judge" to automate evaluation of RAG systems. Use an LLM of your choice to automate your model testing pipeline (i.e use it to redo questions 3 – 5). This will allow you to generate many more test cases to measure performance metrics. Inspect the results. Can you find any instances where the LLM has assessed RAG performance incorrectly? Overall, how useful was the LLM in assessing RAG performance?

Submit both your Jupyter notebook and your data from questions 3 – 6 for marking.