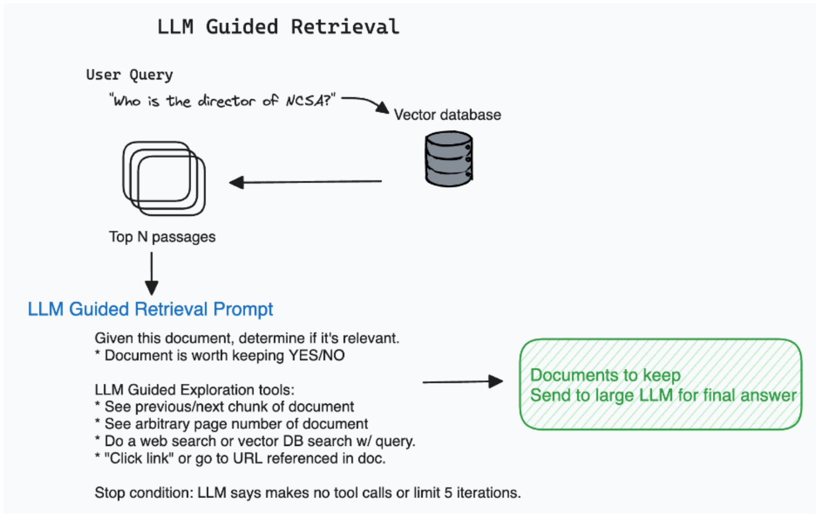
LLM Guided Retrieval

Kastan Day[a], Rohan Marwaha[a], Minu Mathews[a], Volodymyr Kindratenko[a,b]

[a)] Center for AI Innovation, NCSA; [b)] Department of Computer Science

This proposal addresses several critical issues in naïve RAG for factual Q&A. We observe that the responses generated by RAG-based systems depend on the retrieval accuracy; if the relevant document(s) do not make it into the context window, generating an accurate answer is often impossible. A common failure case that we observe on our production RAG application (UIUC.chat) is the “off-by-one" problem: when we use a naïve vector search to identify the most relevant documents, we often match user’s problem *statements*, but we fail to retrieve the problem *solution* that’s a few paragraphs below. Thus, the information truly needed for answering the user query is not included in the context window. This is especially true for large documents containing sparce information. Another frequent case that leads to a poor response is due to a poorly stated user query that does not have the right keywords, thus failing to retrieve relevant documents in the first place. Finally, if the relevant documents are not already in the LLM's dataset, current RAG implementations do not allow for the addition of new content at run-time. E.g., a typical RAG implementation vectorizes documents provided by the chatbot designer before it can be used, thus limiting the responses only to those preloaded ahead of time materials.


LLM Guided Retrieval
User Query
"Who is the director of NCSA?"
Vector database
Top N passages
LLM Guided Retrieval Prompt
Given this document, determine if it's relevant.
* Document is worth keeping YES/NO
LLM Guided Exploration tools:
* See previous/next chunk of document
* See arbitrary page number of document
* Do a web search or vector DB search w/ query.
* "Click link" or go to URL referenced in doc.
Stop condition: LLM says makes no tool calls or limit 5 iterations.
Documents to keep
Send to large LLM for final answer

To address these challenges, we propose to investigate how to inject intelligence into the retrieval process to detect the above cases and intelligently explore the document(s), just like a human would skim and skip around. We propose “LLM Guided Retrieval” where we let a smaller LLM explore the pages of a document given several ‘function calling’ abilities. Given a set of passages (1-3 paragraphs per passage) returned from a naïve vector search, for each document we ask the LLM to rate its relevancy similar to the approach introduced in RElevance-Aware Retrieval augmented framework (REAR)[1], and allow it to use tools to find more relevant passages from the same paper or other papers that were not returned by the naïve vector embedding search. A possible prompt for this technique is given in Appendix A. For every passage from the naïve vector search, we ask the LLM to determine if it's relevant (*Document is worth keeping: YES/NO*). Then, optionally, the LLM may call on the following *tools* to further explore the document:

- see previous/next chunk of document, see arbitrary page number of document,
- do a web search or vector DB search w/ query, and
- "click link" or go to URL referenced in passage. This action can also lead to adding on-the-fly a new document to the RAG vector store and using this newly added information in retrieval.

As we navigate through the document(s) using tools, a knowledge graph can also be built from the “passages that are worth keeping”, which is then stored as part of the document’s context in a manner similar to GraphRAG[2] and is augmented in the content window during the retrieval.

[1] https://arxiv.org/abs/2402.17497

[2] https://www.microsoft.com/en-us/research/blog/graphrag-unlocking-llm-discovery-on-narrative-private-data/

Appendix A: Full prompt for LLM Guided Retrieval using Mistral prompt template

~~~
<|system|>
You are an expert at determining if a passage is relevant and helpful for answering a question, and
further exploring the document to find the most relevant passages.
To be valuable, a passage must have at least some amount of useful and meaningful information with
more than a passing mention of the topic.
As part of your thinking process, you first write a few sentences evaluating the utility of the
passage, given the question we're trying to answer. Limit yourself to writing only a sentence or
two, no more.
Finally, you must submit your final answer by adding two newline characters then "Yes." or "No." or
"I don't know.". Provide a single answer only. Providing multiple final results will disqualify you.
The goal is to find all relevant information to best answer the user's query.

First evaluate if the current passage is worth keeping.
* This passage is relevant and worth keeping: <Enter one of "Yes." or "No." or "I don't know.">
You can take one or more of the following actions:
* [PREV] See previous passage of document
* [NEXT] See next passage of document
* [JUMP TO PAGE: <X>] Jump to see any arbitrary page number of document (such as when the document
says something like "more information available on page x")
* [SEARCH: <QUERY>] Do a web search w/ a new query that you provide
* [GO TO URL: <URL>] "Click link" or go to URL referenced in doc, just tell me the URL you want to
visit.

Here's a template code snippet of how it should work (with placeholder variables):
```
Passage: <The full text of the passage>
Question: <The question we're using to determine relevancy of the passage>
Your evaluation of the utility of the passage: <A few sentences exploring how useful the passage is
for this particular question>


Final answer: <Enter one of "Yes." or "No." or "I don't know.">
* [PREV]
* [SEARCH: <My search query>]

```
Here's a complete example. Follow this formatting exactly.
```
Passage: Figure 4.6: Overview of the CUDA device memory model

In order to fully appreciate the difference between registers, shared memory and global memory, we
need to go into a little more detail of how these different memory types are realized and used in
modern processors. Virtually all modern processors find their root in the model proposed by John von
Neumann in 1945, which is shown in Figure 4.7. The CUDA devices are no exception. The Global Memory
in a CUDA device maps to the Memory box in Figure 4.7. The processor box corresponds to the processor
chip boundary that we typically see today. The Global Memory is off the processor chip and is
implemented with DRAM technology, which implies long access latencies and relatively low bandwidth.

Question: Explain how tiling helps with global memory bandwidth.
Your evaluation of the utility of the passage: The passage briefly mentions the use of shared memory
as a means to reduce global memory bandwidth, but it doesn't provide a detailed explanation or
analysis of how tiling helps with global memory bandwidth. Therefore, the passage is not helpful
when answering the question.

Final answer: No.
* [NEXT]
```</s>
<|user|>
Passage: {context}
Question: {user_query}
Your evaluation of the utility of the passage: </s>
<|assistant|>
~~~