24 Jun 2024

FEDERATED LEARNING-ENHANCED RETRIEVAL AUGMENTED GENERATION (RAG)

Eugenia Kim

Amanda L. Holst Ph.D.

Myungjin Lee

FEDERATED LEARNING-ENHANCED RETRIEVAL AUGMENTED GENERATION (RAG)

AUTHORS:

Eugenia Kim
Amanda L. Holst, Ph.D.
Myungjin Lee

ABSTRACT

Techniques are provided to enhance the functions of a retrieval augmented generation (RAG) mechanism for a large language model (LLM). A Federated Learning (FL)-enhanced RAG (FLERAG) mechanism is provided that can account for relevant context-enhancing data from the retrieval process, as well as most recent data from the FL on which a large language model (LLM) may not have been trained. Using FLERAG, the output generation is determined through a scoring or ranking method that indicates whether the response from the LLM or the FL model is most accurate and relevant. This generated response is then provided back to a user.

DETAILED DESCRIPTION

Large language models (LLMs) are being widely adopted across many sectors. However, these pre-trained LLMs may not be the most up-to-date, as there is typically a cut-off date in the data used to train the model. Moreover, training these LLMs is often costly both financially and in compute resources. While Retrieval Augmented Generation (RAG) has advantages in providing contextually relevant responses (thus, improving the performance of LLMs), it is not without its limitations, one of which is its reliance on the knowledge base. This knowledge base requires continuous updates, which can be crucial to sustaining LLM performance.

Federated learning can be used to enhance the RAG output. The benefit of a federated learning (FL)-enhanced RAG (FLERAG) is that it can account for not just the relevant context-enhancing data from the retrieval process, but also the latest, most recent data from the FL on which the LLM may not have been trained. Using FLERAG, the output generation would be determined through a scoring or ranking method that would indicate

whether the response from the LLM or the FL model is most accurate and relevant. This generated response would then be shared back to the user.

Examples of specific procedures for this process are depicted in Figs. 1 and 2 below.

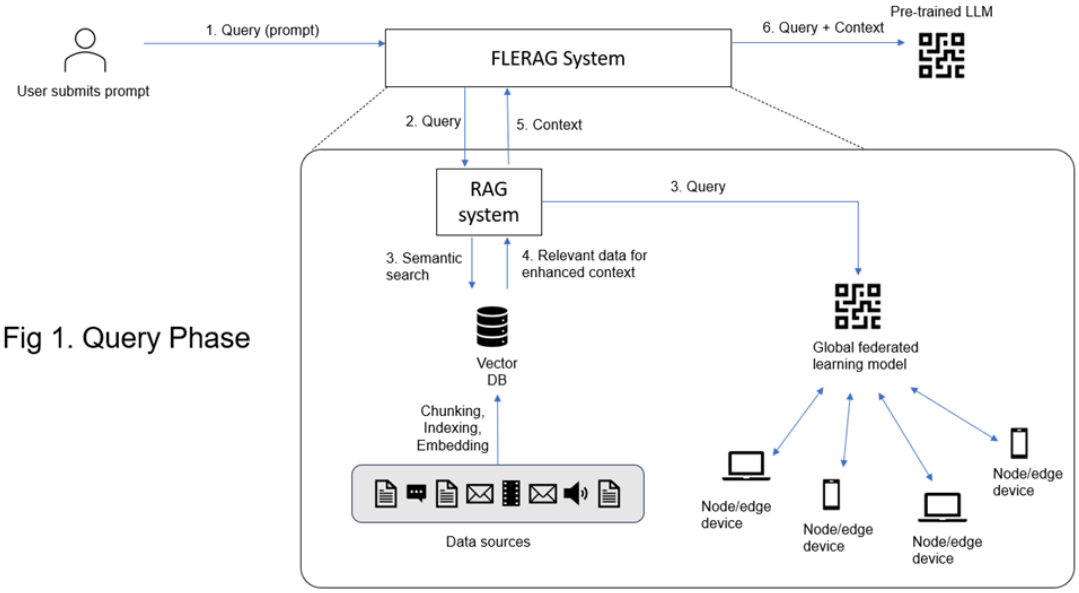

Fig 1. Query Phase

The following describes the workflow of the system.

In the query phase shown in Fig. 1, the system mostly works as a conventional RAG system. One additional operation is to send the query to the FL model (step 3 in the Fig. 1). A FL model can be trained with more up-to-date datasets with the help of clients. By forwarding the query to the FL model, the system can obtain two different query responses (one from the LLM and the other from the FL model), which can be used to enhance the knowledge base in the RAG system (see Fig. 2 below).

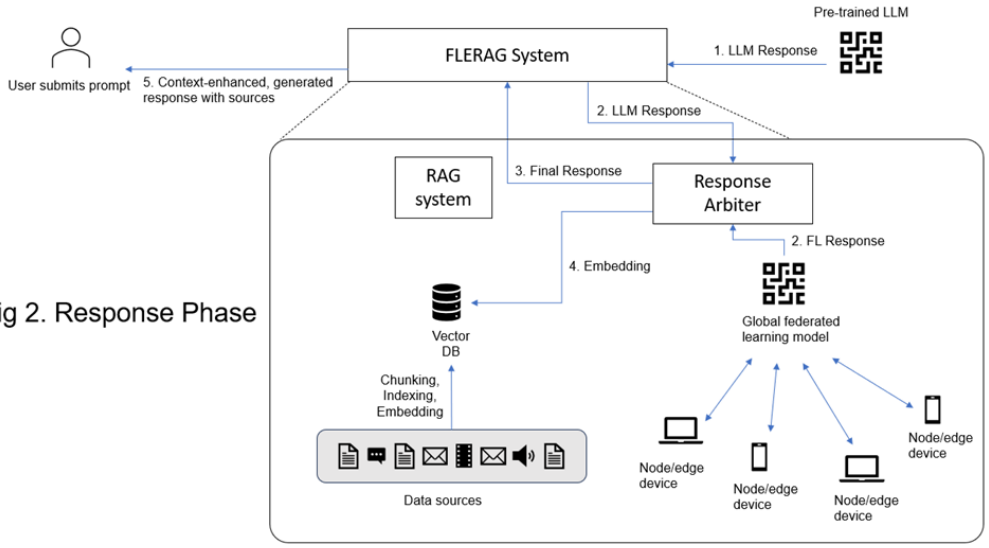

Fig 2. Response Phase

In the response phase shown in Fig. 2, the responses from the LLM and FL model are fed into a component called the response arbiter. The response arbiter is responsible for three tasks: (1) response selection, (2) embedding creation and update into the vector DB and (3) the selected response delivery. First, the response arbiter selects a response with higher confidence score. To compute the score, the response arbiter can rely on a third-party solution such as a public search engine. By comparing the search results with the responses, the response arbiter can estimate the relevance of the responses. Based on the relevance, the response arbiter can compute the confidence scores.

One implementation to build the confidence score is to use the order of responses in the search results and compare each response with responses from the pretrained LLM and the FL-based model. For example, assume that a search engine returned results [q1, ..., qn]. A comparison is made between rp (the response from the pretrained LLM) with the returned results [q1, ..., qn]. A comparison is made for rf (the response from the FL model). Formally, there is function f(ri, [q1, ..., qn]) where i = p (the pretrained LLM) or f (the FL model). The function outputs a score set Si = [si1, ..., sin] where i = p (the pretrained LLM) or f (the FL model). A union is taken of the two sets (Sp, and Sf), a sort is made of the set based on the score (highest to lowest), and a check is made on their rankings. Based on the rankings of the scores, a determination is made as to which model's response should be used. Another method is to use logprobs that reveals the confidence probability of each token. The term "logprobs" refers to a parameter of an LLM (e.g., of the OpenAI platform)

that provides the log probabilities of each output token and a limited number of the most likely tokens at each token position alongside their log probabilities. This can be useful in some cases to assess the confidence of the model in its output or to examine alternative responses the model might have given. By normalizing the confidence of all tokens generated by the pre-trained model and the FL model, a determination can be made as to which response is more reliable.

A combination of both techniques can be used as well. Selecting the FL response implies that the LLM response is generated based on outdated data. Therefore, the response arbiter creates embedding(s) from the FL response and inserts new knowledge (context) into the vector database. At the same time, the selected response will be returned to the user. Using the response arbiter continuously (for every single query) may aggravate the response time. Therefore, the system can maintain the history database that correlates the use of context from RAG and the rate of the LLM response being selected. The system can skip the execution of the response arbiter if there is a high correlation between the two. This can lead to a problem of exploration and exploitation, but those issues are outside the scope of this document.