Query Translation

Caroline Baker and Jacob Cox

I. What is a Query Translator?

There are two main aspects of query translation. The first is when a user enters a query it is in "natural language" which is how we talk. The information being used to answer the queries is stored in a vector store. A vector store holds information as numbers within vectors, then stores the vectors as data points. This structure allows for relational searches instead of exact matching searches. For example, "bee" would be closer to "wasp" than "see" and the vector store would be able to use that to answer queries with greater accuracy and intelligence. A query translator is used to take the natural language and put it into vector formatting, this vector is then sent to the router to be directed where to go for comparison.

The other main aspect is that when a user enters a query, that query can sometimes be in forms that will not be able to properly get the information desired. This is where multi-query translation comes in. Multi-query translation uses an LLM to write multiple versions of the original query to reduce ambiguity and increase the likelihood of obtaining the desired answer. Retrieval is done with each rewritten question and the total retrieved documentation is combined when provided to the generator.

II. Why is a Query Translator Needed Within the Pipeline?

The query translator is needed in order to ensure the query is in the same form as the stored documents so that data retrieval can occur. Also, multi-query translation is likely needed in order to ensure that the user's query will produce the desired result.

III. What are the Possible Problems with the Query Translation?

One potential problem is that with our current system, the goal is for both technical and non-technical users to be able to query information. With this goal, the query translator needs to be able to accurately assess the type of user and query based off of the anticipated desired results.

Another potential problem is that many of the data stores will likely have a high degree of similarity. If a query is not specific enough or is worded incorrectly, then the router might route to the incorrect area, then the retrieval will be incorrect. Also, if information is needed from both data stores, then there could be issues with that as well such as routing to both data stores and getting the correct information from both data stores.

IV. Our Proposed Solutions

Our current solutions are based on existing routing techniques that use an LLM and information about the available data to make retrieval decisions. This approach will be explored further in future sprints.

Other possible solutions to the multiple user types are specifying the user as either technical or non-technical in order to help the query translator tailor the queries accordingly and adding implementing extra natural language processing techniques.

Possible solutions to the data store similarities issue are setting up the system to pose clarifying questions about queries if they are not clear enough, implementing a metadata catalog to aid in routing, and adding in a unified API to aid in handling multiple data source queries.