TFM
RAG

nombres

4. EVALUACIÓN:

==========
[Aquí comentar los objetivos de la evaluación, por qué es necesario, o lo que sea.]

metodología
==========

a. Tipos de evaluación:

LlamaIndex tiene varios módulos destinados a evaluar diversos aspectos del RAG:

- Faithfulness: evalúa si una respuesta concuerda con la información de los nodos en los que se ha basado (i.e. alucinaciones).

- Relevancy: evalúa si la respuesta + nodos fuente concuerdan con la query. Útil para medir si la query ha sido respondida con la respuesta.

- Correctness: evalúa la relevancia y exactitud de una respuesta generada comparándola con una respuesta de referencia.

- Semantic: calcula la similaridad entre los embeddings de la respuesta generada y de una respuesta de referencia.

- Guidelines: evalúa si en la respuesta generada se están cumpliendo las guidelines especificadas.

- Retriever: evalúa un retriever, comparando los documentos que obtiene para cada query dada con los correspondientes documentos de referencia de cada query.

==========
A parte de los módulos de LlamaIndex, existe una gran variedad de librerías de terceros para este efecto, que ofrecen posibilidades como… para este trabajo nos ceñiremos a los módulos que vienen con LlamaIndex.

Métricas.

datasets: datasets de llamahub… comentar por qué es bueno usar varios datasets… la ventaja de tener estos datasets… que se puden generar datasets, ventajas…
==========

c. Descripción del código:

Para la evaluación hemos creado tres archivos de código:

- dataset_downloader.py: descarga datasets de llamahub destinados a la tarea de evaluación.

- eval_batch_multiple_evaluations: ejecuta múltiples evaluadores simultáneamente sobre un modelo.

- definimos la key de OpenAI como una variable de entorno. Esto nos permitirá hacer llamadas a modelos de OpenAI.

- definimos el LLM “gold” que usaremos para evaluar nuestro modelo. En este caso GPT-4 (Tener en cuenta que las llamadas a este modelo son caras con respecto a otros, así que para pruebas podemos usar otros más baratos).

- definimos los evaluadores. Los evaluadores usados en conjunto en este archivo pueden evaluar faithfulness, relevancy, correctness, semantic, y guidelines. Para las guidelines definimos primero las guidelines en un diccionario y a continuación creamos un evaluador por guideline. A todos les pasamos por parámetro el LLM “gold” del paso previo, menos al de semantic, que evalúa similitud entre embeddings y puede pasársele un modelo de embeddings o que coja el que haya en Settings.

- cargamos de un archivo el dataset para evaluación que vayamos a usar, y sus correspondientes documentos. Alternativamente podemos generar un dataset a raíz de unos documentos dados, usando DatasetGenerator.

- del dataset sacamos las queries y las respuestas de refencia. Opcionalmente podemos reducir su número para ahorrar dinero/tiempo en caso de que el dataset sea más grande de lo que queremos.

- definimos el modelo a evaluar.

- inicializamos un BatchEvalRunner con todos los evaluadores, y llamamos a su método aevaluate_queries pasándole nuestro modelo, las queries y las respuestas de referencia, para obtener los resultados de la evaluación.

- analizamos los resultados. Para ello definimos la función get_eval_results, que nos calcula para un evaluador dado, una score como la media de las scores de todas las queries. Podemos también ver resultados de queries concretas para ver otros atributos a parte del score, como por ejemplo el feedback.

- eval_retriever: evalúa un retriever.

- definimos la key de OpenAI como una variable de entorno. Esto nos permitirá hacer llamadas a modelos de OpenAI.

- cargamos de un archivo los documentos que vamos a usar.

- creamos los nodos a partir de los documentos, eligiendo el chunk_size que queramos.

- a partir de los nodos, creamos el vector store index y el retriever.

- si queremos, podemos mostrar los resultados de usar el retriever con una query dada.

- definimos el LLM que usaremos para generar un dataset de queries y contextos a partir de los nodos.

- aquí, o bien podemos generar un dataset de parejas de queries y contextos usando la función generate_question_context_pairs, y luego guardarlo en un archivo, o bien podemos cargar dicho dataset de un archivo de uno que hayamos generado y guardado anteriormente. Es muy recomendable generarlos solo una vez y guardarlos, ya que cuesta tiempo y, dependiendo del modelo usado para generarlos, dinero, el generarlos.

- inicializamos el evaluador, pasándole las métricas que queremos y el retriever.

- si queremos podemos obtener los resultados para una query concreta.

- obtenemos los resultados para todo el dataset.

- definimos una función que toma los resultados y nos devuelve un dataframe con la media de los resultados de cada métrica, y la usamos para mostrar los resultados medios de cada métrica.