Evaluation pipeline for code documentation RAG

Process of building pipeline

As my first step, I decided to use the State of the Union corpora. Since only one corpus was needed, I preprocessed the set of queries and excerpts by removing irrelevant queries and the `corpus_id` column. Next, I used the Fixed Token Chunker from the provided implementation. I cleaned up unnecessary code and placed the final version in a file called `fixed_token_chunker.py` so that it could be later imported and used in the pipeline.

My next step was to implement precision and recall metrics for evaluating the RAG system's performance. I referred to the implementation from the referenced paper, but since the paper included four different evaluation metrics, my version was much simpler and required significantly less code. The `Evaluator` class can be found in the `evaluation.py` file.

For the embedding model, I chose `all-MiniLM-L6-v2`, as it was suggested in the task description and I was already familiar with it from my previous RAG-related projects. I implemented it by installing the `sentence-transformers` module, as recommended in the Hugging Face documentation. I then created a function in the pipeline that takes a batch of texts and returns their corresponding embeddings.

Once all components were ready, I built the evaluation pipeline. The process begins by loading the queries and excerpts, and then chunking the corpus using the previously implemented chunker. For evaluation purposes, I also generated metadata containing the start and end indexes of each chunk.

The next step in the pipeline was to create a ChromaDB collection using the embedding function. I then inserted the chunks, which were automatically converted into embeddings. Once the vector database was prepared, the evaluation phase began: the pipeline created embeddings for each question and queried the ChromaDB collection to retrieve a specified number of relevant chunks.

Finally, the evaluator computed precision and recall by comparing the metadata of the retrieved chunks with the ground truth references. The program returns the average precision and recall scores across all queries.

Findings and conclusions

The results of the evaluation can be found in the table below. I evaluated chunk sizes of 200, 400, and 800, using retrieval settings of 2, 5, and 10 chunks.

The first observation is that precision scores are consistently low across all configurations, while recall scores tend to be relatively high. This outcome is expected because precision is sensitive to the number of false positive tokens—tokens that are retrieved but are not actually relevant. Since both precision and recall are calculated at the token level (not the chunk level), the retrieved chunks often include more tokens than necessary, negatively impacting precision.

On the other hand, the consistently high recall indicates that the RAG system is generally successful in retrieving relevant tokens—it rarely misses them. The worst-performing configuration was a chunk size of 800 with only 2 retrieved chunks. This setup resulted in the lowest recall (53.5) and one of the lowest precision scores (1.5). This makes sense: the large chunks contained many unrelated tokens (lowering precision), and retrieving only two chunks wasn't sufficient to cover all relevant references (lowering recall).

In contrast, the best-performing setup—based on both precision and recall—was a chunk size of 200 with 2 retrieved chunks. The smaller chunks and fewer retrievals reduced the number of irrelevant tokens (leading to the highest precision), while still retrieving many relevant tokens (resulting in a strong recall score). This suggests that the RAG system using the `all-MiniLM-L6-v2` model performs well on the given corpus under these conditions.

As expected, the highest recall scores (close to 100) were achieved with a higher number of retrieved chunks. However, this also led to the lowest precision scores, due to the inclusion of many irrelevant tokens.

It's important to note that results could vary with different corpora or reference structures (e.g., longer references), but based on the current dataset, we can conclude that the evaluation pipeline is functioning correctly, and the outcomes are logically consistent.

	Chunk Size	Number of Retrieved Chunks	Precision Score	Recall Score
1	200	2	8.5	81.4
2	400	2	3.4	63.9
3	800	2	1.5	53.5
4	200	5	3.8	91.9
5	400	5	1.8	81.5
6	800	5	0.8	76.1
7	200	10	2.0	99.5
8	400	10	0.9	90.7
9	800	10	0.5	97.2