Intelligent chunking methods for code documentation RAG

Internship Task

Michal Svec

Introduction

This is a report for an internship task. It describes the implementation of the retrieval evaluation pipeline for a `FixedTokenChunker` algorithm. At the end, we analyze the retrieval results obtained using various parameters.

Implementation

Most of the logic is contained in the file `src/retrieval_evaluation_pipeline.py`. In this file, we implemented the class `RetrievalEvaluationPipeline`. This class takes an embedding function, a corpus, and question labels as input. The pipeline can be run using the `.run()` method with specified parameters such as chunk size, overlap, and the number of retrieved chunks. This method outputs the mean and standard deviation of recall and precision metrics. The paper that was referred in the task uses the Chroma database, but our script does not use it. For similarity search, we use standard cosine similarity.

The embedding function for the pipeline can be chosen arbitrarily. We used `all-MiniLM-L6-v2` model because it is more lightweight to run and appears to be more widely used compared to `multi-qa-mpnet-base-dot-v1`. However, they claim that `multi-qa-mpnet-base-dot-v1` was specifically trained for semantic search, so this model could potentially yield better performance in benchmarks.

When it comes to `FixedTokenChunker`, we are using mostly default settings. We looked at the available encodings but decided to stay with the default `cl100k_base` encoding. This encoding is best for OpenAI models like GPT-4 and is optimized for efficiency and lower token count. The only modifications we made were to the chunk size and overlap.

Besides the `src/retrieval_evaluation_pipeline.py` file, we also added some other files, but most of them are taken from the Chunking Evaluation repository. These include implementations for finding the start and end indices of excerpts in a corpus and various operations with ranges. It would be unnecessary to implement them from scratch.

Metrics

The referred paper in the task uses multiple metrics such as precision, recall, precision$_\Omega$, F1-score, and IoU. As specified in the task, we implement only recall and precision, which are defined as follows:

$$\text{Recall}_q(C) = \frac{|t_e \cap t_r|}{|t_r|},$$
$$\text{Precision}_q(C) = \frac{|t_e \cap t_r|}{|t_e|},$$

where the query is denoted by q, C represents the chunked corpus, t_e is the set of tokens among all relevant excerpts, and t_r is the set of all retrieved tokens from the chunks. The final score is the mean of the scores across all queries.

Results

chunk size	overlap	num chunks	precision	recall
200	0	2	9.0 ± 7.1	84.2 ± 32.7
200	0	4	4.7 ± 3.5	88.8 ± 27.7
200	0	6	3.3 ± 2.2	94.7 ± 18.7
200	0	10	2.0 ± 1.3	98.0 ± 12.0
200	50	2	9.1 ± 7.2	82.6 ± 31.9
200	50	4	5.4 ± 4.0	93.4 ± 20.9
200	50	6	3.7 ± 2.5	96.7 ± 13.2
200	50	10	2.3 ± 1.5	97.6 ± 11.7
200	100	2	10.9 ± 8.6	85.5 ± 32.1
200	100	4	6.4 ± 5.0	94.7 ± 19.4
200	100	6	4.6 ± 3.4	97.6 ± 14.5
200	100	10	2.8 ± 1.9	**100.0 ± 0.0**
200	150	2	**12.4 ± 10.4**	84.8 ± 33.9
200	150	4	8.9 ± 7.0	92.6 ± 22.4
200	150	6	6.6 ± 5.0	98.6 ± 8.2
200	150	10	4.3 ± 3.2	**100.0 ± 0.0**
400	0	2	3.9 ± 3.9	70.2 ± 44.8
400	0	4	2.2 ± 1.9	79.6 ± 39.8
400	0	6	1.5 ± 1.2	83.6 ± 36.6
400	0	10	1.0 ± 0.7	93.4 ± 23.4
400	50	2	3.8 ± 3.7	71.4 ± 42.6
400	50	4	2.1 ± 1.8	81.5 ± 36.7
400	50	6	1.5 ± 1.1	88.1 ± 29.8
400	50	10	1.0 ± 0.7	93.2 ± 21.9
400	100	2	4.5 ± 3.9	81.5 ± 35.6
400	100	4	2.5 ± 1.8	88.7 ± 28.4
400	100	6	1.8 ± 1.3	93.4 ± 21.5
400	100	10	1.1 ± 0.8	96.1 ± 19.3
400	150	2	5.2 ± 4.1	89.4 ± 29.1
400	150	4	2.8 ± 1.9	96.1 ± 16.8
400	150	6	2.0 ± 1.2	99.4 ± 3.4
400	150	10	1.2 ± 0.8	99.4 ± 3.4
600	0	2	2.1 ± 2.2	65.9 ± 45.3
600	0	4	1.3 ± 1.2	76.2 ± 40.9
600	0	6	0.9 ± 0.8	82.7 ± 36.9
600	0	10	0.6 ± 0.5	89.3 ± 29.7
600	50	2	2.2 ± 2.0	63.8 ± 46.6
600	50	4	1.2 ± 1.1	72.5 ± 43.5
600	50	6	0.9 ± 0.7	79.0 ± 39.5
600	50	10	0.6 ± 0.5	89.2 ± 30.7
600	100	2	2.6 ± 2.6	70.4 ± 44.4
600	100	4	1.5 ± 1.3	79.6 ± 38.8
600	100	6	1.1 ± 0.8	90.5 ± 28.9
600	100	10	0.7 ± 0.5	97.1 ± 16.2
600	150	2	2.7 ± 2.4	72.9 ± 42.5
600	150	4	1.5 ± 1.2	80.6 ± 38.2
600	150	6	1.1 ± 0.8	85.2 ± 34.4
600	150	10	0.7 ± 0.5	87.8 ± 31.6

Table 1: Results of the evaluation of the `state_of_the_union` corpus

For the evaluation, we tried multiple hyperparameters using a simple grid search. The parameters can take the following values:

$$\begin{aligned} \textit{chunk\_size} &\in \{200, 400, 600\}, \\ \textit{overlap} &\in \{0, 50, 100, 150\}, \\ \textit{num\_chunks} &\in \{2, 4, 6, 10\}. \end{aligned}$$

From the results, we can see that configurations with `chunk_size = 200` achieve the best precision, then it significantly drops. The maximum precision reached is 12.4% with `chunk_size=200, overlap=150, num_chunks=2`. Higher overlaps tend to contribute to better precision. This occurs because the answers are relatively short. Precision values are generally quite low. This indicates that while relevant information is often retrieved, it is mixed with a large amount of irrelevant information within the retrieved chunks.

There also appears to be a correlation between overlap and recall. The higher the overlap, the higher the recall. In addition, recall increases as the number of retrieved chunks grows. However, larger chunks tend to negatively impact recall.

The presented results come from the `state_of_the_union` corpus. The results may change based on the nature of the used data. For instance, the best precision for `wikitexts` dataset is around 10.2% and recall scores are also lower. In our code, we have also included results in a CSV file for the `wikitexts` and `chatlogs` corpora.

Conclusion

From the results, it seems that smaller chunks, a certain overlap, and a larger number of chunks can provide a balance between precision and recall. If precision is critical, fewer chunks with overlap may be sufficient. If recall is important, more chunks with overlap are ideal. However, this conclusion does not necessarily apply to every scenario, as it depends on the nature of the input corpus.

We are aware that evaluation and analysis could be done more thoroughly, such as graph plotting, using a more advanced approach to hyperparameter tuning, or utilizing experiment tracking software. However, due to time constraints (sometimes it is difficult to juggle both university and other activities), we decided to stick to the task description.