Task 2 Report

Carl Zhang, Siyi Hu

23 Mar 2025

Overview

In this report, we describe our implementation of a dense retrieval system for document search. The overall goal is to embed both corpus documents and user queries into a high-dimensional space, then retrieve the documents most similar to the query embeddings. We use a CSV/JSONL corpus converted into a Hugging Face `Dataset`, embed the corpus documents once, and then perform retrieval by computing cosine or dot-product similarities.

Data Preprocessing and Loading

Before loading the corpus for our retrieval system, we performed a series of data preprocessing steps on our raw dataset, which originally came in a mix of PDF and JSON formats. Initially, we attempted to store the dataset in CSV format as instructed, but we found that CSV files loaded slowly and were not well-suited for storing large documents. Therefore, after applying text cleaning procedures to remove newline characters, extra spaces, and extraneous metadata such as DOI links, copyright statements, funding details, correspondence information, and reference markers, we converted the CSV files into a single JSONL file. Finally, the cleaned dataset was deduplicated based on the title and text fields and converted into a Python dictionary where each record contains an ID, title, and text.

```
{
"doc_id_1": {"title": "...", "text": "..."},
"doc_id_2": {"title": "...", "text": "..."},
...
}
```

Model Selection and Retrieval Workflow

We began our study by analyzing the MTEB leaderboard[1] to identify suitable text embedding models that support both Chinese and English. Since we have limited GPU resources, we focused on lightweight models that run efficiently on CPU. Based on these criteria, we ultimately selected `sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2` for our project's multilingual performance and efficiency.

[1] https://huggingface.co/spaces/mteb/leaderboard

Our dense retrieval system works by loading the `SentenceTransformer` model, indexing documents by encoding their title and text, computing query embeddings, and then retrieving the top-k documents based on cosine or dot-product similarity.

Model Architecture

Our chosen model leverages a MiniLM-v2 architecture, which is a distilled version of BERT using knowledge distillation to reduce computational costs. The model combines a 12-layer Transformer encoder with a mean-pooling layer:

```
SentenceTransformer(
  (0): Transformer({'max_seq_length': 128, ...})
  (1): Pooling({'word_embedding_dimension': 384, 'pooling_mode_mean_tokens': True})
)
```

During our experiments, we evaluated several Hugging Face checkpoints:

- **BAAI/Aquila-135M:** Provided high-quality embeddings but was too slow for practical use.
- **lajavaness/bilingual-embedding-small:** Targeted for English and French, but did not offer significant improvements over common multilingual models.
- **sentence-transformers/distiluse-base-multilingual-cased-v2:** Did not outperform our final choice in terms of retrieval quality.

After thorough testing, we decided to use `sentence-transformers/paraphrase-multilingual-Mir` which offers an excellent weight between embedding quality and computational efficiency, making it ideal for local or Colab-based usage.

Results and Discussion

Our dense retriever successfully ranks documents for a wide range of queries. However, we note some limitations. For example, in some demo queries (e.g., mentioning both "flying fish" and "strange building"), the retriever heavily focused on one phrase (flying fish) and missed results relevant to the other (strange building). This suggests the need for better handling of multitopic queries.

Therefore, in the future, we plan to break long or complex queries into subqueries, retrieve relevant documents for each subquery, and then merge results to capture multiple aspects of user intent.

Conclusion

We have built a dense retrieval system that takes advantage of a SentenceTransformer-based encoder to achieve a robust document ranking. Although we explored several models, we found a mid-sized BERT variant with mean-pooling to be most balanced in terms of speed and accuracy. Future improvements will focus on subqueries for complex queries and better multifactor retrieval.