Information Retrieval project

Dataset

Firstly, we installed the essential libraries: 'datasets' that facilitates easy access and management of various datasets and 'python-terrier', which is a tool used for efficient retrieval and analysis of textual data, commonly employed in natural language processing tasks

Then we entered the code that provides a convenient way to navigate and operate within the directory structure of Google Drive. We used 'cd_drive_directory(dir='')' and 'mount_google_drive()' functions to ensure that Google Drive is properly mounted.

After that, we imported necessary libraries and using them we simplified the process of downloading and accessing specific splits of datasets from the Hugging Face model hub.

BeIR dataset preparer

This module prepares the BeIR dataset for use in information retrieval tasks. It includes functions to download and convert the Quora corpus, queries and qrels to the formats expected by PyTerrier, a library for building and evaluating information retrieval systems. The code is structured to download the datasets, convert them to the desired formats and return the converted datasets for further use.

Text preprocessor

Next module is a text preprocessor that provides functions to preprocess textual data. The code includes functions to check for and use a cached preprocessed dataset, preprocess the dataset if not cached and perform various text preprocessing steps such as converting to lowercase, removing non-words, removing stop words and stemming. It employs the NLTK library for natural language processing tasks and incorporates regular expressions for text manipulation.

Index creator

Then we created an index. The 'create_index' function checks if an index has been built earlier by looking for a cache. If a cached index is found, it returns the index based on the cached data. The function '_create_index' creates an instance of DFIndexer and indexes the dataset with two columns: text that contains the text of the documents and docno which contains the id of the documents. The '_return_index_from_cache' function facilitates the retrieval of a cached index with a warning to ensure users are mindful of the cache's potential limitations. We also used '_initialize_pyterrier' that ensures if PyTerrier is initialized. This function involves starting the Java Virtual Machine, downloading Terrier files and configuring them to work with Python.

Search performer

This module performs search and facilitates interactive searches using two different weighting models: BM25 and TD-IDF. They give a user a prompt to input queries. The '_search' function first preprocesses the query and through the 'get_retriever' it retrieves an instance of a retriever using the specified weighting model. The results are then merged with the original dataset, based on the docno column from the search results and the _id column from the dataset. In summary, the module serves an interactive search interface, allowing users to input queries and receive readable search results based on different weighting models.

Interactive part

Firstly, we changed the current working directory to the specified path in Google Drive. Functions, such as 'quora_corpus', 'quora_queries', 'quora_qrels = get_beir_quora_dataset()', retrieve the BeIR Quora dataset which consists of the corpus, queries and relevance judgments. The index for the corpus is created. It prepares the dataset for subsequent information retrieval tasks, enabling efficient search and evaluation of retrieval models on the Quora data.

This code, including providing an interactive search experience, allows the users to input queries and receive human-readable search results from the Quora corpus. The two different weighting models (BM25 and TF-IDF) offer users flexibility in exploring search results with different relevance ranking strategies.

Part with long calculations

In next lines of code, the functions 'evaluate_search_engine' assess the performance of the search engine using the BM25 and TF-IDF weighting models on the Quora dataset. The results are stored in the variables 'bm25_evaluation_results' and 'tfidf_evaluation_results'. The evaluation results are displayed in a tabular format using the 'display' function which provide valuable insights for understanding and comparing the performance of the search engine with different weighting models.

The displayed metrics for BM25 include:

- Precision at *k* (P@ *k*) that measures the fraction of relevant documents among the top *k* retrieved documents

- Recall at *k* (R@ *k*) that measures the proportion of relevant documents retrieved among all relevant documents in the dataset

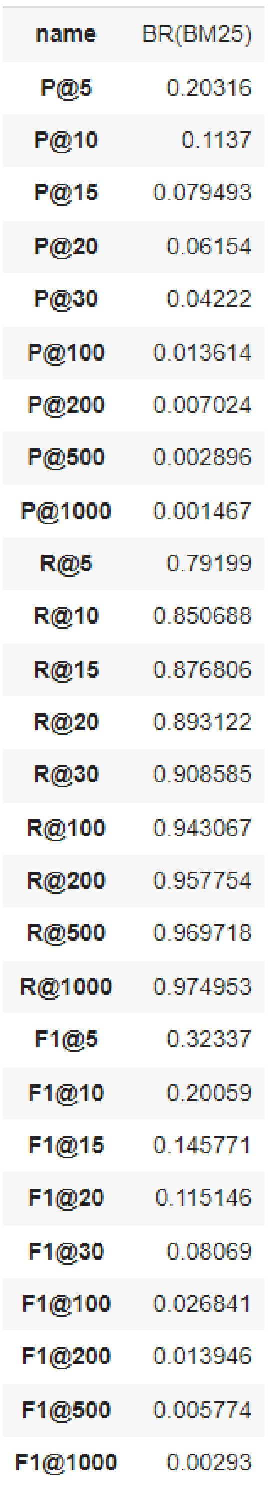
- F1 score at *k* (F1@ *k*) that computes the F1 score based on precision and recall at the top *k* retrieved documents

Description for some examples:

- P@5 means that precision at the top 5 retrieved documents is 0.20316, indicating that 20.32% of the top 5 documents are relevant

- R@1000 means that recall at the top 1000 retrieved documents is 0.974953, indicating that 97.50% of all relevant documents in the dataset are captured within the top 1000

- F1@200: at the top 200 retrieved documents, the F1 score is 0.013946, so the search engine is achieving a low balance between precision and recall



name	BR(BM25)
P@5	0.20316
P@10	0.1137
P@15	0.079493
P@20	0.06154
P@30	0.04222
P@100	0.013614
P@200	0.007024
P@500	0.002896
P@1000	0.001467
R@5	0.79199
R@10	0.850688
R@15	0.876806
R@20	0.893122
R@30	0.908585
R@100	0.943067
R@200	0.957754
R@500	0.969718
R@1000	0.974953
F1@5	0.32337
F1@10	0.20059
F1@15	0.145771
F1@20	0.115146
F1@30	0.08069
F1@100	0.026841
F1@200	0.013946
F1@500	0.005774
F1@1000	0.00293

The displayed metrics for TD-IDF:

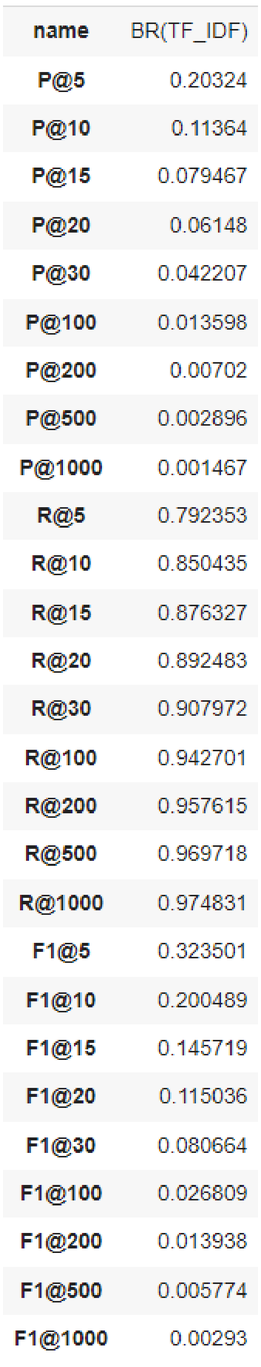
P@ *k*, R@ *k* and F1@ *k*: the metrics are calculated at various values of *k*, representing the top *k* retrieved documents.

Description for some examples:

- P@5 means that among the top 5 retrieved documents, approximately 20.32% are relevant

- R@10 means that out of all relevant documents, around 85.04% are included among the top 10 retrieved documents

- F1@500 means that the F1 score at the top 500 retrieved documents is 0.005774, indicating a trade-off between precision and recall



name	BR(TF_IDF)
P@5	0.20324
P@10	0.11364
P@15	0.079467
P@20	0.06148
P@30	0.042207
P@100	0.013598
P@200	0.00702
P@500	0.002896
P@1000	0.001467
R@5	0.792353
R@10	0.850435
R@15	0.876327
R@20	0.892483
R@30	0.907972
R@100	0.942701
R@200	0.957615
R@500	0.969718
R@1000	0.974831
F1@5	0.323501
F1@10	0.200489
F1@15	0.145719
F1@20	0.115036
F1@30	0.080664
F1@100	0.026809
F1@200	0.013938
F1@500	0.005774
F1@1000	0.00293

Word2Vec

We installed the Gensim library which is a natural language processing library in Python, often used for topic modeling and word embedding tasks. The loaded Word2vec model, referred as ‘model’, is ready to be utilized for a range of natural language processing tasks.

Using the Word2Vec model, we calculated the similarity between the query tokens and the document tokens. The function identifies documents with a similarity score above a specified threshold, normalizes the similarity scores and returns a list of document IDs that have the similarity criteria. The output provides a list of document IDs having the similarity criteria for further analysis or retrieval tasks.

Using the ‘find_similar_documents’ function, a comprehensive search is performed for each unique query ID in the Quora queries dataset. The results are stored in a dictionary named ‘search_results’, where each query ID maps to a list of retrieved document IDs with similarity scores above 0.5.

With ‘pickle’ library, the code prints the retrieved document IDs for each query and the results are serialized and stored for future use. The evaluations are calculated based on the counts of true positives, false positives and false negatives. The counts are utilized to calculate precision, recall and F1 score, taking into account the trade-off between precision and recall.

Results

Precision: 0.0002

Recall: 0.8857

F1 Score: 0.0004

Precision measures the accuracy of the retrieved results. In this context, it indicates the proportion of the retrieved documents that are truly relevant to the queries. Our value of 0.0002 suggests a very low precision, which means that a large number of the retrieved documents are false positives.

Recall measures the completeness of the retrieval, representing the proportion of relevant documents that were successfully retrieved. Our recall of 0.8857 implies that a high percentage of the relevant documents were identified, however, the low precision suggests that many irrelevant documents were also included in the results.

The F1 score represents a balanced measure between precision and recall. The F1 score of 0.0004 reflects a satisfactory balance.