WSM PROJECT

June 16, 2023

Abstract

We implement an simple search engine based on elasticsearch. We implement boolean search and ranked search. And for further processing, we include multi-news summarization and question answering.

1 Introduction

Our project is a comprehensive search engine that has successfully fulfilled the specified requirements. It provides a powerful and efficient search system to cater to users' information needs related to news. Let's delve into the details of the project.

To ensure the authenticity and relevance of the indexed documents, we have utilized the real-news-like folder of the C4 dataset, which encompasses approximately 15 gigabytes of data. This dataset can be accessed through the following link: C4 Dataset.

Our search engine offers two forms of search functionality, allowing users to explore news-related information through different approaches.

Boolean Search: Users can input search keys and apply operations such as AND, OR, and NOT to refine their search queries. The search engine retrieves and returns all the original documents that satisfy the specified search conditions. This ensures precise and flexible search capabilities, enabling users to narrow down their results effectively.

Ranked Search: In this search mode, users provide a query, and our search system generates a ranked list of search results (origin documents). When ranking the search results, we consider factors like semantic relevance and freshness. We employ various ranking methods to provide users with the most relevant and up-to-date information based on their queries.

In addition to returning the original documents in the search results, our search engine incorporates advanced functions for enhanced user experience and understanding of the retrieved information.

Multi-news Summarization: As an advanced feature of ranked search, our system groups news articles related to the same event into a single category and generates a summary. This allows users to obtain a comprehensive overview of the different news articles associated with their query. Each generated summary is associated with the original documents that contributed to its creation, enabling users to access the detailed information if desired.

Question-Answering (QA): Our search engine supports question-answering functionality within the scope of news. For instance, when a user queries "How old is Donald Trump?", our system provides the answer "76" as the top result and links it to the relevant document collection. Although our corpus does not contain knowledge data like Wikipedia, our QA feature enables users to ask questions related to news events, such as "How many casualties in the xxx incident?".

By incorporating these advanced features, our search engine ensures an intuitive user interface, accurate search results, and comprehensive information retrieval. We continuously strive to optimize and refine our system to deliver an exceptional search experience and provide users with the most relevant and useful news-related information.

2 Search system

2.1 Index Construction

Index construction plays a crucial role in information retrieval, especially in search engines. It involves the process of organizing and structuring documents or data in a way that facilitates efficient and accurate retrieval of information. The index serves as a lookup structure that enables quick access to relevant documents based on user queries.

We use ElasticSearch for index construction. ElasticSearch is a distributed text search engine based on Lucene. As a powerful and widely used search engine, it provides robust capabilities for index construction and retrieval.

2.2 Boolean Search

We use the indexer to generate the dictionary file and the posting file. In the posting file, we organize a term and its corresponding document IDs as a key-value pair, and store all the pairs in a dictionary order of the key. Moreover, in the dictionary file, we store all the terms and their byte offsets in the posting file as key-value pairs to enable fast lookups.

We implement Boolean Search engine based on the dictionary file and the posting file constructed by the indexer. For a query of a single term, we just lookup the dictionary file to get its byte offset in the posting file, and fetch the corresponding pair in the posting file. The value of this pair is a list consisting of the document IDs of all the documents containing the term. For a query like *a AND b*, where each the operand may be a query or a term, we intersect the results of these two queries. And, we use the union of the two queries when processing *OR* operation. In addition, for query *NOT c*, we calculate the complement set of the document IDs obtained from query c on all document IDs.

In addition to operators *AND*, *OR* and *NOT*, we further implement the operators *(* and *)*, where *(* and *)* have the highest priority, followed by *NOT*, *AND* and *OR*. We use stack to convert the input expression into a suffix expression, and then process the results in the order of the suffix expression. Fig. 1 illustrates this progress with an example input *NOT(NOT(a OR b) AND c)*.

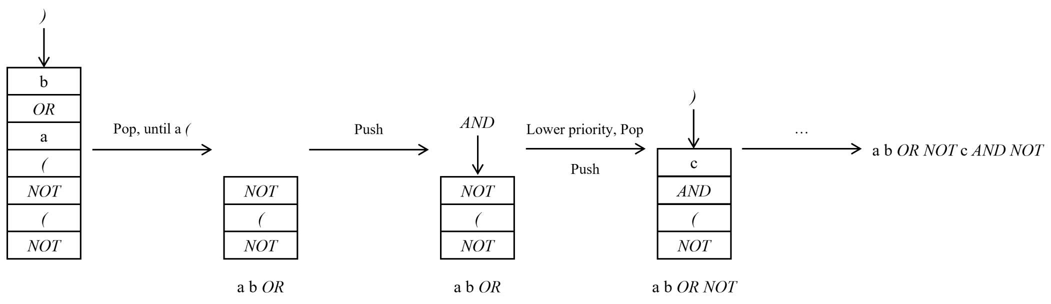

Figure 1: Convert the input into a suffix expression.

2.3 Ranked Search

Thus far we have dealt with indexes that support Boolean queries: a document either matches or does not match a query. In the case of large document collections, the resulting number of matching documents can far exceed the number a human user could possibly sift through. Accordingly, it is essential for a search engine to rank-order the documents matching a query. To do this, the search engine computes, for each matching document, a score with respect to the query at hand.

We assign to each term in a document a weight for that term, that depends on the number of occurrences of the term in the document. We would like to compute a score between a query term t and a document d, based on the weight of t in d. The simplest approach is to assign the weight to be equal to the number of occurrences of term t in document d. This weighting scheme is referred to as term frequency and is denoted $\mathrm{tf}_{t,d}$, with the subscripts denoting the term and the document in order.

Raw term frequency as above suffers from a critical problem: all terms are considered equally important when it comes to assessing relevancy on a query. In fact certain terms have little or no discriminating power in determining relevance.Denoting as usual the total number of documents in a collection by N, we define the inverse document frequency (idf) of a term t as follows:

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t} \tag{1}$$

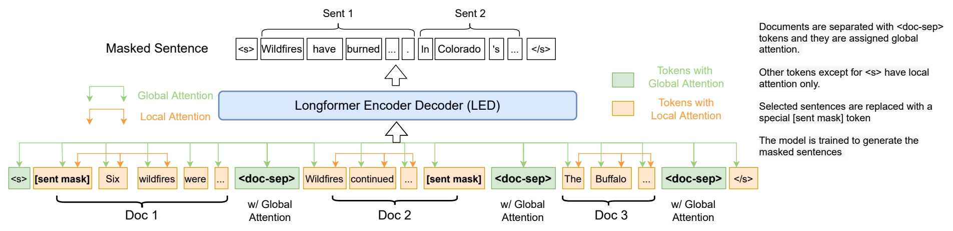
Sent 1
Sent 2
Masked Sentence
<s>
Wildfires
have
burned
...
.
In
Colorado
's
...
</s>
Global Attention
Local Attention
Longformer Encoder Decoder (LED)
🟩 Tokens with Global Attention
🟧 Tokens with Local Attention
Documents are separated with <doc-sep> tokens and they are assigned global attention.
Other tokens except for <s> have local attention only.
Selected sentences are replaced with a special [sent mask] token
The model is trained to generate the masked sentences
<s>
[sent mask]
Six
wildfires
were
...
<doc-sep>
Wildfires
continued
...
[sent mask]
<doc-sep>
The
Buffalo
...
<doc-sep>
</s>
Doc 1
w/ Global Attention
Doc 2
w/ Global Attention
Doc 3
w/ Global Attention

Figure 2: The model architecture of PRIMERA.

The tf-idf weighting scheme assigns to term t a weight in document d given by

$$\mathrm{tf}-\mathrm{idf}_{t,d} = \mathrm{tf}_{t,d} \times \mathrm{idf}_t \tag{2}$$

we may view each document as a vector with one component corresponding to each term in the dictionary, together with a weight for each component.We introduce the overlap score measure: the score of a document d is the sum, over all query terms, of the number of times each of the query terms occurs in d:

$$\mathrm{Score}(q, d) = \sum_{t \in q} \mathrm{tf}-\mathrm{idf}_{t,d} \tag{3}$$

3 Search result processing

3.1 Multi-News Summarization

Multi-News Summarization is the task of generating a summary from a cluster of related news documents. Graph-based networks and pre-trained language models (typically encoder-decoder transformers) have shown great advantages for generating summarizations. In this part, we will present the model we use, named PRIMERA [XBCC22], a pre-trained model for multi-document representation with a focus on summarization that reduces the need for dataset-specific architectures and large amounts of fine-tuning labeled data. PRIMERA achieves superior performance compared with prior state-of-the-art pretrained models as well as dataset-specific models in both few-shot and full finetuning settings.

Unlike prior work, PRIMERA minimizes dataset-specific modeling by simply concatenating a set of documents and processing them with a general efficient encoder-decoder transformer model. The underlying transformer model is pre-trained on an unlabeled multi-document dataset, with a new entity-based sentence masking objective to capture the salient information within a set of related documents. Our own experiments show the great potential of this model and it can performly summarize the key information given several news.

3.2 Question-Answering

RoBERTa [LOG+19] has been shown to outperform BERT and other state-of-the-art models on a variety of natural language processing tasks, including language translation, text classification, and question answering. In their work, they present a replication study of BERT that carefully measures the impact of many key hyperparameters and training data size. We find that BERT was significantly undertrained, and can match or exceed the performance of every model published after it. The modifications of the architecture include: training the model longer, with bigger batches, over more data removing the next sentence prediction objective training on longer sequences dynamically changing the masking pattern applied to the training data.

In this work, we use a roberta-base model, fine-tuned using the SQuAD2.0 dataset. It's been trained on question-answer pairs, including unanswerable questions, for the task of Question Answering.

4 Conclusion

In this project, we sucessfully implement an search engine with boolean search and ranked search. For furthur, processing, we implement multi-news summarization and question answering.

References

[LOG$^+$19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[XBCC22] Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. Primera: Pyramid-based masked sentence pre-training for multi-document summarization, 2022.