
INFORMATION RETRIEVAL COURSE PROJECT

KOK-KIP SEARCH ENGINE

Liang Yucheng
16340133

School of Data and Computer Science
Sun Yat-sen University
Guangzhou, Guangdong, China 510006
jacky14.liang@gmail.com

Liang Junhua
16340129

School of Data and Computer Science
Sun Yat-sen University
Guangzhou, Guangdong, China 510006
alva112358@gmail.com

Zeng Guangtao
16340010

School of Data and Computer Science
Sun Yat-sen University
Guangzhou, Guangdong, China 510006
kids937@foxmail.com

July, 2019

ABSTRACT

Search Engine is an indispensable part of people's daily life and document searching is an important branch of the field of information retrieval. What we have achieved in this project is a document-oriented search engine, which is supposed to deliver great search results to users. The most creative part in our project is our score system, which is facilitated by the ensemble idea that bagging various scores to rank documents reasonably based on their semantic similarities toward query words. Even if this is a small course project, under align development, we still have a frontend and a backend separately, providing premium searching experience for users.

Keywords Information Retrieval · Search Engine · Document Retrieval · Word Embedding

1 Introduction

In the last ten years, search engine has become a useful tool for people. People search nearly everything with an search engine, including papers, news, documents, government websites and so on. Now we have a variety of advanced search engines, such as Bing, Google, Search Encrypt, Yandex, Swisscows. Considering that Information Retrieval Course mainly focuses on document retrieval, we are looking forward to implementing a creative document-oriented search engine by applying some of the methods covered in class. We have traditional methods like tf-idf and bm25, mainly considering the document relevance regarding query words. Also we make full use of the latest technology, word embedding. With word embedding, we are able to discover the semantic similarity between words and documents, which is difficult to be captured by traditional methods.

This paper will be illustrated in the following ways: In Section 2, some related works will be discussed, some of which have inspired this project more or less. And I will uncover the details behind this engine in Section 3, both principles and applications. In Section 4, we will present the result of our search engine. And in the last section, we may summarize our project and describe some future works.

2 Related Work

Nowadays, there are a great number of search engines that feed different kinds of users. Google is a popular search engine that performs well in nearly all kinds of searching. And Google Scholar is a document search system that contains millions of papers. PageRank facilitate Google in ranking valuable and popular pages by considering their linking relationships in network. Bing performs better than other rivals in image searching across all browsers, while Search Encrypt focuses on information privacy by encrypting queries. And Yandex is the most popular search engine in Russia, offering a suite of pretty cool tools. So there are various kinds of search engine and our project only concentrate on document searching.

In terms of document retrieval, in SimSeerX, Kyle Williams presents a content-based document retrieval engine to allow a document-to-document query. There are three major algorithm: Key Phrase Similarity, Shingle Similarity and Simhash Similarity. Key Phrase Similarity is mainly based on BM25, extracting key words from a formatted document. Shingle Similarity is inspired by k-gram algorithm. It first generates a series of shingles and then hashes words to numbers in order to calculate similarity. Simhash Similarity maps a document into a 64-bits vector and then use the hamming distance to determine the similarity between two documents.

Most of the methods mentioned above prefer to compress a document into a high dimension vector, which may cause a considerable information loss and thus lead to a huge amount computation. Therefore, we put our effort to use new advanced methods to optimize query result, like word embedding. Our ultimate aim is to deliver good quality result in a short time.

3 Design and Algorithm

As our search engine mainly focuses on document retrieval, we concentrate on developing a score system to deliver good result to users. Our score system combines three algorithms: tf-idf, bm25 and word embedding. The simple reason why we choose these methods is that they complement each other. While tf-idf and bm25 prefer exact matching and valuable words, word embedding favors related words based on semantic similarity. Inspired by the idea of model bagging, we assign different weight to these algorithms and add them linearly. By doing so, we can both capture those key words and return the most relevant document. Also we have other tricks to improve the quality and efficiency of searching. In the following passage we are going to elaborate the details of implementation.

3.1 Document Dataset

All documents are downloaded from a news website. The size of our dataset is about 100 and we may enlarge it in the future if we have enough computational resource.

3.2 Database Design

Database storage is of utmost significance for a search engine. A well-designed storage structure may benefit search efficiency. We prefer construct our database offline in order to speed up query time. We store words, documents and their relationship separately in three tables. Also, out of consideration for computation, we store word embedding for each word in database as bytes, we can significantly reduce calculating time when a query is requested.

3.3 Algorithm

Currently we have ensembled three different algorithms to calculate score.

3.3.1 TF-IDF

3.3.2 BM25

3.3.3 Word Embedding

In this project, we use the most popular word embedding model – word2Vec model. It can be obtained using two methods: Skip Gram and Common Bag Of Words(CBOW). The input is the one-hot encoding of a word and the output is the probability of the occurrence of other words. The higher probability between two words, the closer they stay in the semantic space.

Gensim can compute the similarity score between two embedding vectors easily, but it is mostly based on Euclidean Distance, which may unreliable when two vectors have different in length. Instead, we use cosine similarity and it is proven reliable even two vectors have different length.

Undoubtedly it is time-consuming and extremely difficult to train a good model by us own in such short time. Luckily, with the help of Tencent AI Lab Embedding Corpus for Chinese Words and Phrases, we are able to construct a word2Vec model easily in Python. However for a real time search engine, it still takes long time to load from the dataset, so we decide just use a portion of it. By doing experienments repeatedly, we make a trade-off between accuracy and efficency. And finally we use 700,000 records out of 8, 000, 000 in the dataset, which still deliver good search results.

4 Result

We evaluate our product from vairous angles. Firstly, we care about the used time when a query is requested. Here we use end-to-end time to evaluate the speed of a query. Second, we expect a good result but it is difficult to quantify. Questionnaire is a good alternative but we don't have many users. So instead we merely evaluate the result based on our feeling.

4.1 MainPage

We have design a search engine logo and frontend to present our result, the initial image of the search engine is as **Figure 1**

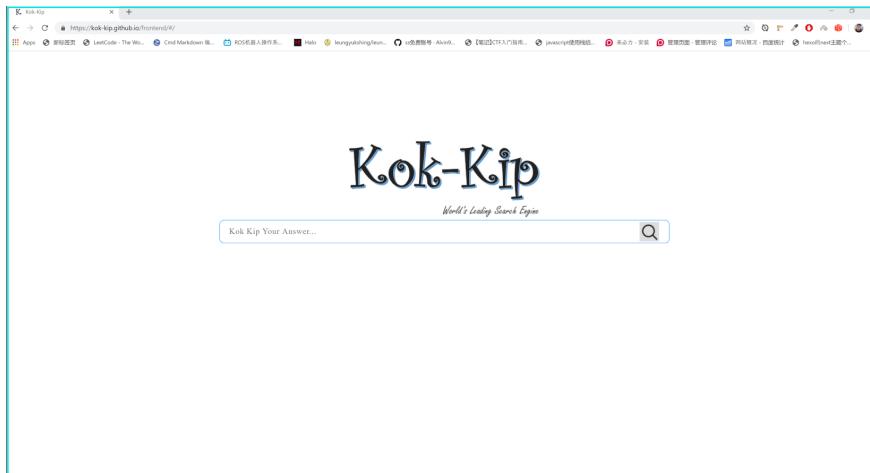


Figure 1

4.2 Search Result

4.2.1 Search for information about Cell

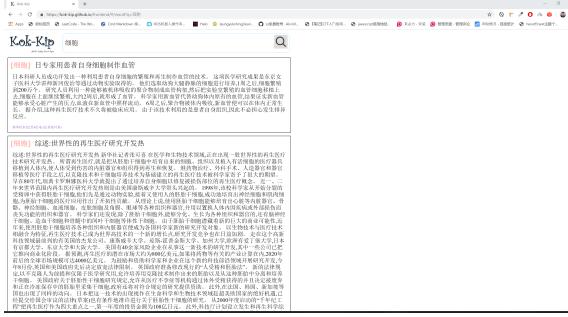


Figure 3: Search for Cell 1



Figure 4: Search for Cell 2

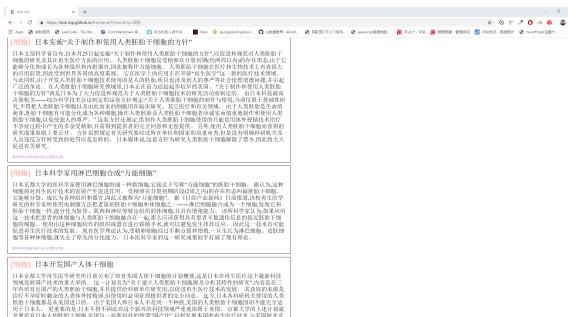


Figure 5: Search for Cell 3

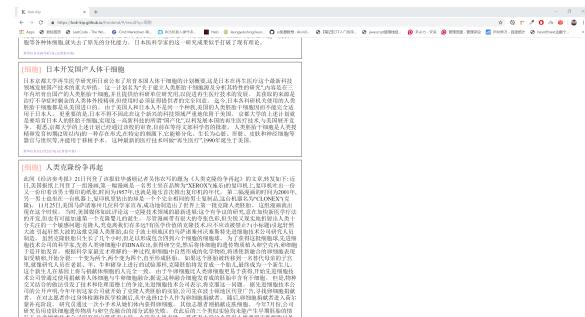


Figure 6: Search for Cell 4

4.2.2 Search for information about China

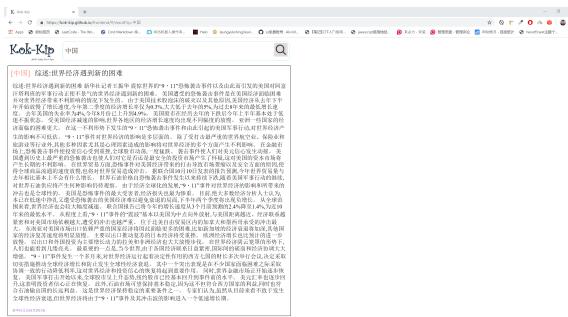


Figure 7: Search for China 1

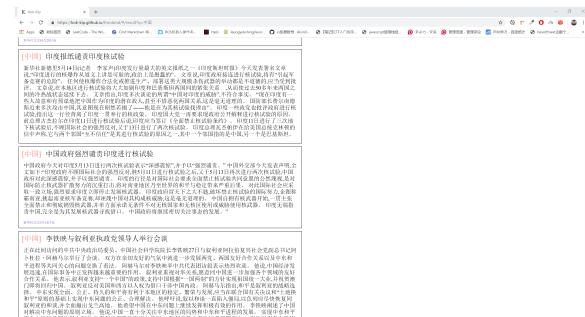


Figure 8: Search for China 2

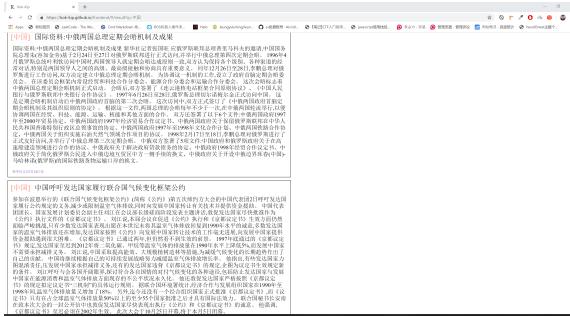


Figure 9: Search for China 3

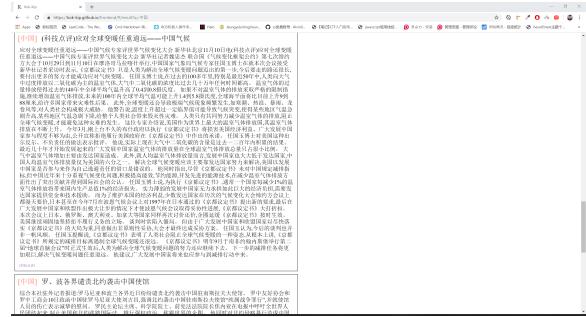


Figure 10: Search for China 4

4.2.3 Search for information about Atmosphere

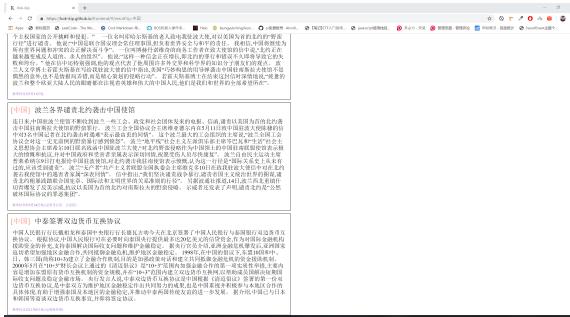


Figure 11: Search for Atmosphere 1

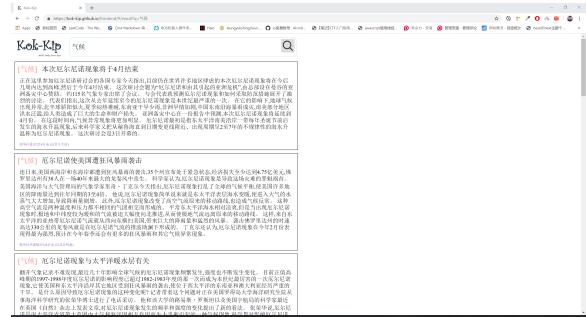


Figure 12: Search for Atmosphere 2

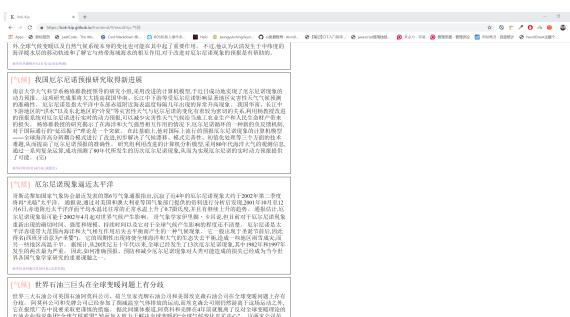


Figure 13: Search for Atmosphere 3



Figure 14: Search for Atmosphere 4

5 Future Work

Our engine is largely limited by the embedding model. We can access the Tencent Corpus Dataset, but it is unlikely to load it into a word2Vec model due to its large size. Thus what we neet to do is that we should filter those high frequency words from the dataset and load them into a model, which could improve search accuracy greatly. Also, our dataset size is too small to support a popular document searching. We may support incremental documents and thus enlarge our document pool to return more relevant results. Moreover, we may try to extract information from documents if they share a similar content structure. This may provide more significant information regarding themselves. Last, we are going to deploy this service in a remote server so we may have more computational resource. But it will take some time to find a suitable machine.

6 Summary

Given limited time and resources, it's quite amazing for our team to finish this project. From design to implementation, we all do it by our own, without any reference. The most thing we want to present in this project is that we learn from Information Retrieval Course this term. Benefiting from Scrum, we assure our product's quality and have a clear division and cooperation. As depicted above, our search engine achieves good performance in this dataset and we believe it could perform well in other dataset. In a word, thanks for contributions from each member and we will perfect this project in the future.

References

- [1] The anatomy of a large-scale hypertextual Web search engine
- [2] The Google PageRank Algorithm
- [3] SimSeerX: A Similar Document Search Engine
- [4] A Full-Text-Based Search Engine for Finding Highly Matched Documents Across Multiple Categories
- [5] A Semantic Search Engine for Indexing and Retrieval of Relavent Text Documents
- [6] Distributed Representations of Words and Phrases and their Compositionality