Introduction

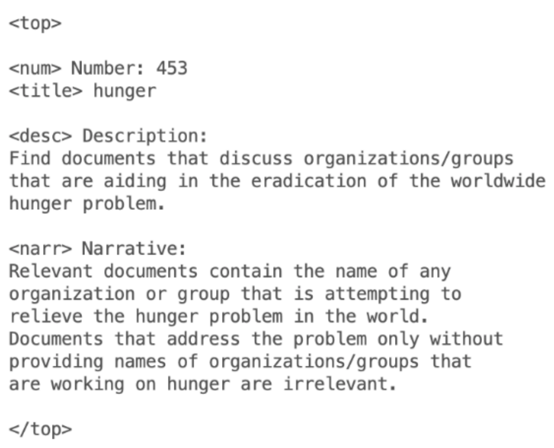
The objective of this project is to design a search engine using a specific retrieval model. In order to implement a model in a search engine design, the fundamental IR tasks are to be carried out, which involve indexing, retrieving, and evaluating. We aim to design a search engine that can return results with consistent performance in the given sets of data. In this project, we focus on the BM25 model and its modified variants. Over the years, there has been improvements on BM25, which attempts to improve the model by either modifying the BM25 formulation or integrating additional information [1]. The modified BM25 model chosen introduces additional parameters, collection term frequencies (CTF), which are used to reveal the informativeness of certain terms that may have similar or equal term frequencies (TF) or inverse document frequencies (IDF). We decided to use the BM25-CTF model proposed by Sergio, et al [1] as the retrieval model for our search engine. The reason for us choosing this model is because we want to investigate the performance of BM25-CTF on different datasets by conducting ad-hoc evaluations and then analyze the results to observe the new model's ability to discriminate between different important terms.

Datasets

The web datasets we chose are from Text Retrieval Conference (TREC) which has been collected since 1999. The datasets chosen are aimed to explore specific aspects of Web retrieval, including named page finding, topic distillation, and traditional ad-hoc retrieval [3]. All datasets originate from free-text data and are stored in JSON format.

We will use 6 datasets to test the performance of our search engine - TREC Web Track 1999 – 2004 dataset. Meanwhile, we are trying to apply for ClueWeb09-T11Crowd (TREC-2011 Crowdsourcing) dataset for free from TREC official website, which may spend two weeks. For the web datasets from 1999 to 2003, each of them has 50 documents and each document, but the 2004 dataset has 225 documents. In total, there are 475 documents for the usage of our search engine.

For each document, it contains four parts: identifier, title, description and narrative, though the 2003 and 2004 datasets lack some parts. The identifier is denoted by a number or a combination of letters as well as numbers, the title is a topic which is commonly comprised by a couple of words, the description is very brief about the topic, and the narrative is a detailed content about the topic. An example in the TREC 2000 and an example in the TREC 2003 Web Track are shown below.



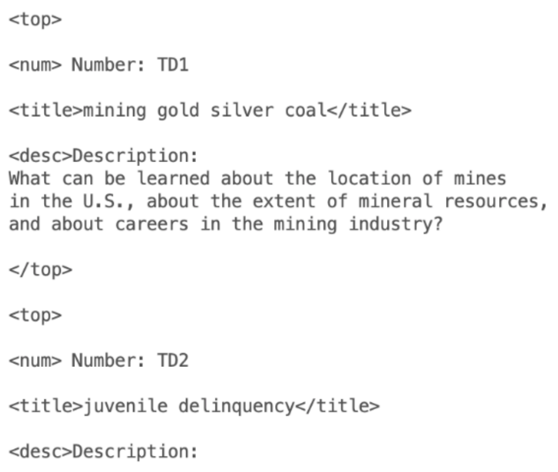
```
<top>

<num> Number: 453
<title> hunger

<desc> Description:
Find documents that discuss organizations/groups
that are aiding in the eradication of the worldwide
hunger problem.

<narr> Narrative:
Relevant documents contain the name of any
organization or group that is attempting to
relieve the hunger problem in the world.
Documents that address the problem only without
providing names of organizations/groups that
are working on hunger are irrelevant.

</top>
```



```
<top>

<num> Number: TD1

<title>mining gold silver coal</title>

<desc>Description:
What can be learned about the location of mines
in the U.S., about the extent of mineral resources,
and about careers in the mining industry?

</top>

<top>

<num> Number: TD2

<title>juvenile delinquency</title>

<desc>Description:
```

Fig 1. The left image shows an example of a document of the TREC 2000 dataset; The right image shows an example of a document of the TREC 2003 dataset.

Tools to Use

Building a search engine requires understanding a few important points regarding the design of a system. The most common approach is to define the architecture, modules, interfaces and data to satisfy specified requirements. The architecture involves the entire search process framework, from which it can be broken into several modules. The modules within the search engine architecture can be broken down into a client and server model, where the client sends a query and the server sends back a response. In information retrieval, the more complicated module is built on the server end where the retrieval model is implemented. For these modules to communicate, interfaces are needed for these programs to talk to each other, and this is where RestAPI comes into place. RestAPI allows data to be communicated in a structured format, which plays an important role in sending data back to the user. Lastly, the data used are free text, which includes query terms as well as indexed terms for a collection of documents.

From a model design point of view, the tasks can be divided into implementation and evaluation. In this project, Elasticsearch and Kibana are the main modules integrated into the client and server model mentioned earlier for implementation as well as testing. The figure below shows a brief architecture between the Elasticsearch, Kibana and the default Apache server. In short, Elasticsearch contains multiple analytic tools that can be used for model implementation (e.g. Lucene/BM25 formula), whereas Kibana is a visualization tool used for illustrating data in a graphical manner. For instance, Elasticsearch provides simple client and server libraries, which can be used for the environment setup. In addition, it also provides evaluation API to enable quality evaluation on query results.

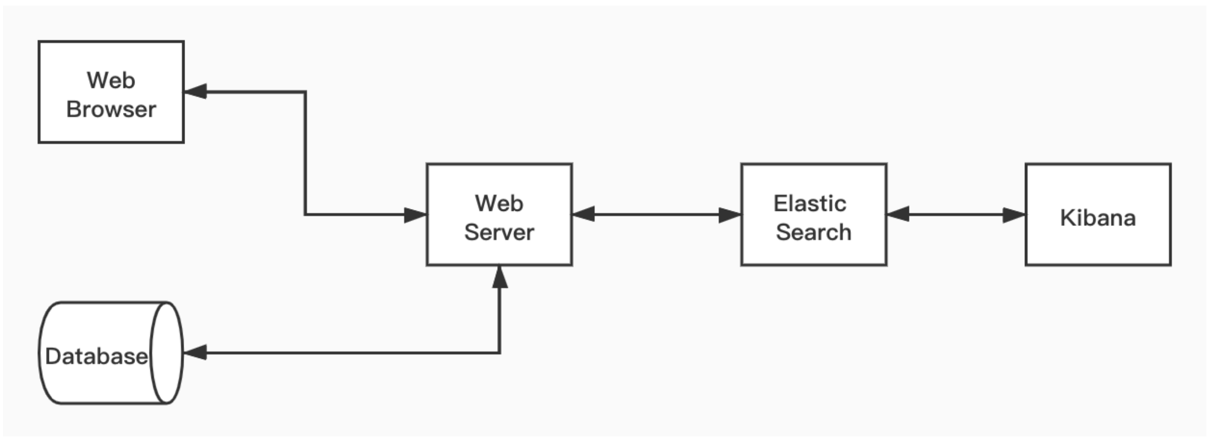


Fig 2. Tools integrated to the server and client Model (Elasticsearch, Kibana)

Search Process Architecture

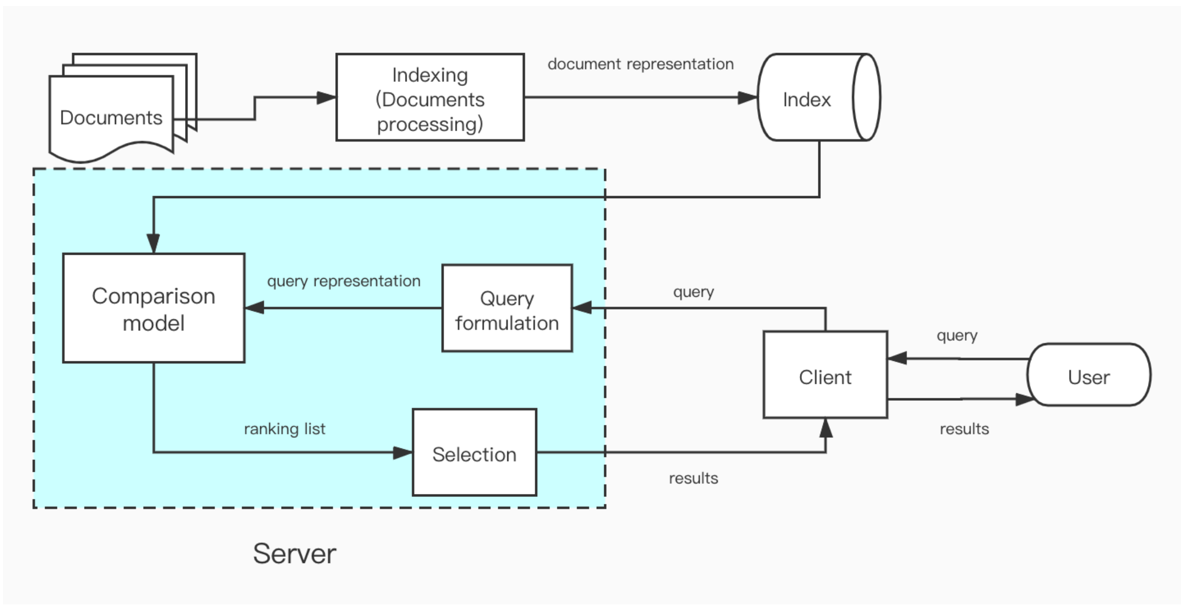


Fig 3. The architecture of the search engine

The architecture can be roughly divided into three stages: document indexing, user interaction and model application. First, there are some basic processes in document indexing stage such as tokenization, stop-word removal, lemmatization, stemming etc. After that, some statistical information (e.g. term frequency, document frequency, collection term frequency) should be counted as well. In the second phase - user interaction, users could send their queries to the server by client. Finally, as shown in the blue area above, the server will process query first, then the comparison models will measure similarity and compute ranking scores, and the next step is that the results will be chosen from the ranking list and then returned to the client. In our case, the server denotes Elasticsearch, and Kibana is the client we will use.

Methodology

The methodology used for the search engine design can be distinguished with respect to each stage of the search process. It can also be understood as directly corresponding to the main methods used for each of the IR tasks (indexing, retrieving and evaluating). The retrieval model (BM25-CTF) can be implemented by modifying the built-in function provided by Elasticsearch. The new statistic introduced, CTF, is to be calculated at the document indexing stage. In order to compute the new idf weights (bidf, as in the paper) and the new tf weights (btf), the methods for BM25 similarity measurement in the apache.lucene library would have to be modified to replace idf with bidf and tf with btf. For instance, the “public final long totalTermFreq()” method can be directly used to compute ctf, and the other mathematical operations can then be carried out. In short, according to paper by Sergio, ctf is introduced as an additional parameter. Since BM25 can be broken down into four parts, namely tf, idf and two saturation functions, tf and idf can be treated separately with ctf. The BM25-CTF formula can then be developed based on these modifications by combining the two modified terms, bidf and bctf. For instance, according to the paper, in order to describe the one of the two extremes constrained on the two ends by Zipf’s law, it is said that bidf (boosted-idf) can be built on the hypothesis that when the idfs for two terms are equal, ctf can be a factor that distinguishes between the two terms. Bidf composes of cidf * idf and pidf (Poisson idf), where pidf is computed based on the divergence from randomness method. As for each of these components, computations can be done by either creating new methods or by utilizing the existing constructors (objects in Apache Lucene libraries). The same technique can then be applied for modification in tf, and then the computed bidf and bctf are combined to form the new BM25-CTF formula [1], as shown below. By following the architecture described above, the tf, idf, and ctf are computed at the indexing stage, and the computation of BM25 is modified in the and thus represents the new comparison model. In addition, for the experimental setup since the experimental model we are going to use is BM25-CTF, some other modified BM25 models as control groups can be used to compare against our experimental model. The models of control group are said to be decided later on in the implementation stage of the project [2].

$$\sum_{w \in q} bidf(w) \times \frac{(k_1 + 1) \times btf(w,d)}{k_1 \times K(d) + btf(w,d)} \times \frac{(k_3 + 1) \times btf(w,q)}{k_3 + btf(w,q)} \tag{1}$$

For evaluation, we focus on the system's effectiveness (system-oriented evaluation). In other words, the goal of an IR system is to retrieve as many relevant documents as possible and as few irrelevant documents as possible. The evaluation metric is measured by the combination of precision and recall. We intend to evaluate the performance of models by single value methods - Precision at document cut-off level (written as P@I) and Mean Average Precision (MAP). These methods will be used since they are very robust to large test collections. The ROC curve will also be used in order to show a graphical representation of precision and recall relationships for different retrieval models.

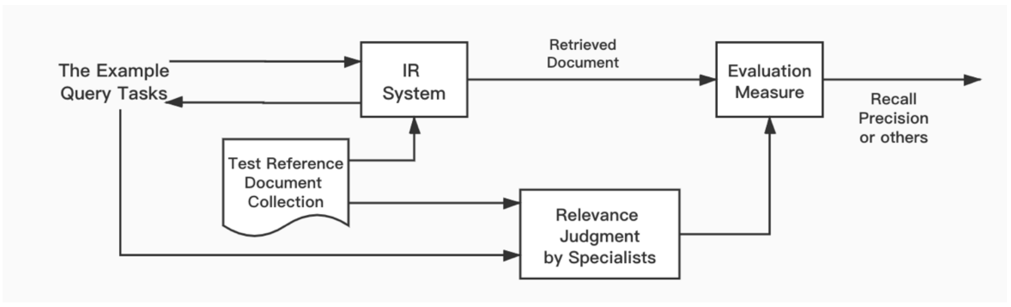
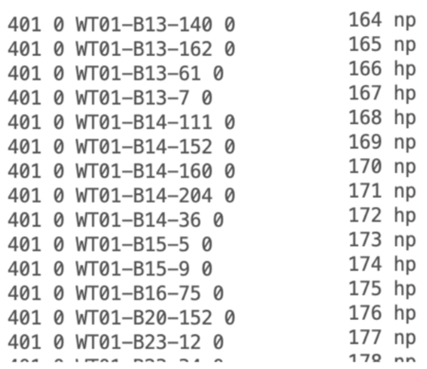

The Example Query Tasks
IR System
Retrieved Document
Evaluation Measure
Recall Precision or others
Test Reference Document Collection
Relevance Judgment by Specialists

Fig 4. Evaluation process

The figure above shows a complete overview of the search system with respect to evaluation measures. There are two parts (two routes) to go through in order for evaluation. By using the same test documents and query tasks, relevance is judged by both the search system and human experts. In our case, the "judgement" datasets are given by the human experts. These datasets are included in each year's track from TREC. After obtaining precision and recall values, single value summaries can then be carried out. Further analysis can be done on Kibana. Figure 6 shows the relevance judgement results for web track obtained from TREC.



```
401 0 WT01-B13-140 0        164 np
401 0 WT01-B13-162 0        165 np
401 0 WT01-B13-61 0         166 hp
401 0 WT01-B13-7 0          167 hp
401 0 WT01-B14-111 0        168 hp
401 0 WT01-B14-152 0        169 np
401 0 WT01-B14-160 0        170 np
401 0 WT01-B14-204 0        171 np
401 0 WT01-B14-36 0         172 hp
401 0 WT01-B15-5 0          173 np
401 0 WT01-B15-9 0          174 hp
401 0 WT01-B16-75 0         175 hp
401 0 WT01-B20-152 0        176 hp
401 0 WT01-B23-12 0         177 np
```

Fig 5. Two examples of relevant judgment of TREC datasets: the left image shows relevance judgment of TREC 1999 dataset by specialists; The right image shows relevance judgment of the TREC 2003 dataset by specialists.

Workplan/Timeline/Individual Responsibility

Tab. 1 The workplan and individual responsibility

Tasks/Deliverables	Start Time	Participants	Deadline	Expected Outcome
Prepare presentation materials	2020.3.2	all	2020.3.3	Slides
Presentation	2020.3.3	all	2020.3.3	
Knowledge and skill gain	2020.3.4	all	2020.3.12	Elasticsearch and Kibana
Model implementation	2020.3.6	Chung-Hao	2020.3.15	Elasticsearch server contains the model
Experiment - 1	2020.3.16	Yinghao	2020.3.20	Evaluation results gain (such as P@5, P@10, P@20, MAP etc.)
Experiment - 2	2020.3.16	Jingye	2020.3.20	Evaluation results gain (such as P@5, P@10, P@20, MAP etc.)
Result Analysis	2020.3.21	Shangyu	2020.3.25	Notes about the result analysis
Report merge and improvement	2020.3.26	all	2020.3.28	Report
Make Slides	2020.3.28	all	2020.3.31	Slides
Final Presentation		all	2020.3.31	

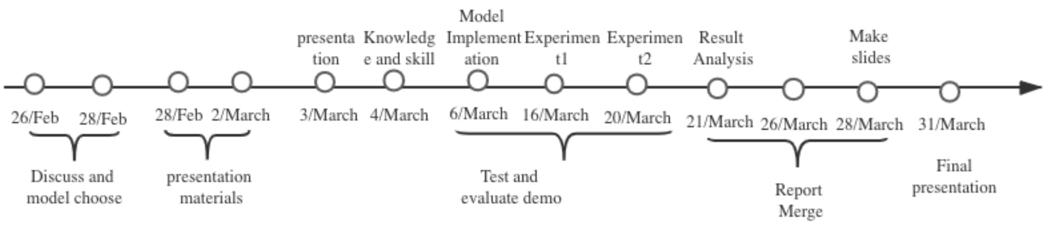


Fig 6. Project timeline

References

[1] Jimenez, Sergio & Cucerzan, Silviu & González, Fabio & Gelbukh, Alexander & Dueñas, George. (2018). BM25-CTF: Improving TF and IDF factors in BM25 by using collection term frequencies. Journal of Intelligent & Fuzzy Systems. 34. 1-13. 10.3233/JIFS-169475.
[2] Connelly, Shane. Practical BM25 – Part 2: The BM25 Algorithm and its Variables. 19/April/2018. https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables.
[3] Web Track Text Retrieval Conference. https://trec.nist.gov/data/webmain.html