Information Retrieval Project Report

Liangqun Lu

**Abstract**

In the age of information overload, information retrieval provides approaches to design and implement solutions for handling large collections of information, and obtain relevant documents. In this report, we presented our work in information retrieval project where we implemented a local information search engine inside university of Memphis site. Starting with the main website of university of Memphis, we crawled web pages within university of Memphis. After we applied text operation on the web texts, including tokenization, removal of stop words and non-alphabetic words and morphological stemming, we saved 10, 000 webpages with at least 50 words as text database. In the retrieval model, we calculated tf-idf weights from vector space model and used cosine similarity between queries and documents to extract and rank relevant documents. Through 10 example queries, we evaluated our information retrieval model using precision and recall partly based on google search reference. Moreover, we used python Django module to build an interface to implement our search engine.

**Keyword** information retrieval, tf-idf, cosine similarity, model evaluation, interface

**1.Introduction**

Information retrieval (IR) is a subject of searching for information in a document, and it can be searching for documents, as well as searching for metadata, including images and sounds. Information retrieval can avoid overwhelmed of information overload and obtain relevant information efficiently. Nowadays, it has been widely used on search engines. One of the most successful examples is google, which uses PageRank algorithm to rank relevant websites and supports daily billions of searches in the worldwide.

Various IR models have been developed to provide efficient search and they vary on document and query representations and retrieval functions. With different mathematical basis, three general models set-theoretic, algebraic and probabilistic models are used to represent words or phrases. They are corresponding to these three classific IR models, boolean models, vector space models and probabilistic models. In boolean models, all documents are represented as a set of keywords while queries are boolean expression of keywords, connected by AND, OR and NOT. The output document relevancy is binary, meaning relevant or non-relevant. In vector space models, each document has its own distinct index terms after text preprocessing and operation. Term Frequency (TF) and IDF (Inverse Document Frequency) are calculated as term weights, which makes documents vectorization and comparable. For the query, TF-IDF is also calculated. With document vectors and query vector, similarity measurements including Euclidean distance, Manhattan distance, inner product and cosine similarity are used to indicate relevancy and output relevant documents. In probabilistic models, the retrieval of documents is considered as a probabilistic inference. Then similarities are computed as probabilities of document being relevant to the query. Besides the text representation, retrieval models can also consider term dependence. Some advanced IR models Fuzzy retrieval, Latent semantic indexing assumes term dependence while Boolean and vector space models are opposite. In the evaluation of IR models, precision and recall are generally used to measure efficiency regarding the relevancy of retrieved terms.

In this project, we crawled 10000 web pages from site memphis.edu and processed the texts as a text database. Then we used vector space model and cosine similarity between the query and documents to implement our search engine. We also evaluated the efficiency of our model using precision and recall.

**2.Approach**

In the implementation, we made use of web crawling to generate text database, calculated term TF-IDF at each documents, and used cosine similarity to retrieve and rank the relevant documents.

2.1 Text database

We utilized python modules urllib and beautifulsoup to crawl memphis.edu site and saved the texts for the following information retrieval. Starting with the main website [www.memphis.edu](http://www.memphis.edu), we stored the linked web pages within memphis.edu site. To make it efficient, we used breadth-first search and queue to save the links and go through the links. For each accessed web page, we operated the text using tokenization, stop word removal, non alphabetical removal, and word stemming. When the word number was at least 50, we saved the urls of the webpages and the processed texts. In this way, we collected 10000 web pages within memphis.edu site.

2.2 Vector space model and TF-IDF weight

Vector space model is a weighted model which represented documents, queries as vectors of index terms. It has been widely used in information filtering, information retrieval and relevance ranking. In one classic vector space model, the term frequency (TF) and inverse document frequency (IDF) are used to weight the vectors. The formulas can be seen below.

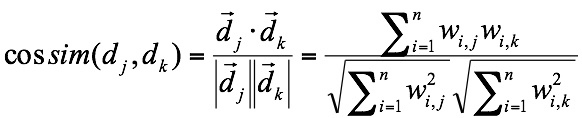
Tft,d = term t frequency in document d

dft = number of documents containing term t

Idft = log10 |N|/ dft

2.3 Cosine similarity

Following TF-IDF formulas, we calculated TF-IDF weight value for each index term inside each document. In the same way, we obtained weights for queries. Then we implemented cosine similarity to reflect the relevance between queries and documents. Cosine similarity is defined as below. Dj and dk represent two vectors from a query and a document. The similarity is the angle between them, calculated from dot product normalized in length.



In the 10 example queries, we processed the query words in the same way as we processed web page texts, such as stop word removal, non alphabetical removal and word stemming. Then we calculated the weight for queries through tf-idf where the term frequency is the frequency among the search words. After that, we multiplied the query tf-idf and document tf-idf which had weights for all words in the document and normalized by their distance, which is literally cosine similarity formula. In the same way, we have cosine similarity values for each potential relevant documents. In the output, we ranked the retrieval documents using cosine similarity descendingly and added the original url and title for the final retrieval results.

2.5 Evaluation

After we searched the relevant documents using cosine similarity, we evaluated the results by making use of google search. At first, we searched the 10 example queries and limit site to memphis.edu at google search, then we used a script to download 100 retrieval links with the rank for each query. Considering that 10000 web pages are not the all memphis webpages google search against, we removed the linked urls outside our document pool in google search results. After we prepared our google search, we evaluated the precision and recall from our cosine similarity using google results. In google, we add site: memphis.edu to each query, in the hope to extract results only in memphis.edu site.

**3. Design**



Figure 1: architecture of search engine

The figure 1 shows the architecture of our search engine, as well as google search and web crawler. Inside the search engine, we included the operation on user queries and text database. Then we used cosine similarity to extract relevant ranked search results. In order to prepare the text database, we used a web crawler to grab 10, 000 web pages inside memphis.edu site. In the evaluation, we used google search results on the same query inside memphis.edu site. We evaluated the efficiency of our search engine using precision and evaluation, which will be described in the result section.

**4. Implementation**

We mainly used python language to implement the web page crawl, text operation, tf-idf calculation, cosine similarity, rank results and results comparison between our search engine and google search. The main modules include urllib, BeautifulSoup, stemming, pandas and numpy. In the interface, we used python web frame module Django to implement the use query and result retrieval.

**5. Result**

For the 10 example queries, we used our search engine and obtained the relevant documents for each of them. The top 100 retrieved results can be seen in supplementary table 100\_doc\_10\_query table. Surprisingly, the number of relevant documents from 8 queries is over 9000 among 10000 text pool while ‘Cookie’ has 17 matches and ‘semantic similarity’ has 245. Since it is not practical to identify all relevant documents, we assume that those queries with almost total documents have retrieval lots of relevant documents and have high recall scores.

In the calculation of precision, we planned to use google search results as reference. However, it is not realistic for reasons like google engine does not allow scripts to extract results constantly, as well lots of results are from web pages outside memphis.edu even the site is added in the search. Therefore, we used top relevant results from google search and manually checked top retrieved results. In total, we counted the relevant documents in top 15 results and calculated the precision as the total precision. The evaluation results are seen below. The precision was not high as expected, but it is encouraging to improve model and get better results.

Table 1 IR model performance evaluation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **User query** | **Precision** | **Recall** | **Numbers**  **(precision -- recall)** |
| Q1 | international office | 0.13 | 0.98 | 15 -- 9832 |
| Q2 | software engineering research | 0.33 | 0.98 | 15 -- 9846 |
| Q3 | Cookie | 0.13 | 0.017 | 15 -- 17 |
| Q4 | president of the university | 0.5 | 0.99 | 15 -- 9941 |
| Q5 | computer science research awards | 0.53 | 0.99 | 15 -- 9890 |
| Q6 | semantic similarity | 0.267 | 0.025 | 15 -- 245 |
| Q7 | tiger bike's current offer | 0.2 | 0.99 | 15 -- 9999 |
| Q8 | where is the library located? | 0.13 | 0.98 | 15 -- 9844 |
| Q9 | How to graduate with honors? | 0.2 | 0.99 | 15 -- 9871 |
| Q10 | scholarships in computer science | 0.6 | 0.98 | 15 -- 9847 |

6. Future work

This search engine is the implementation from information retrieval, based on the understanding of the course. It consists of text database preparation, query operation, cosine similarity calculation and documents rank. We can retrieve relevant documents for the 10 example queries, but it is not simple to evaluate the performance. We used google search results as reference, but either our results did not cover some obvious results as we did not crawl those in our text database, or google provided results outside memphis.edu site. Besides, the search engine is not appropriate to answer questions directly, because the question mask is removed when the query goes through text operation. Cosine similarity calculates the angle between texts, while in our results, the scores are quite low, which might indicate low relevance. What is more, in our engine, we assumed the term independence while it is not true. In the example queries, it is obvious that these terms are semantic relevant.

In the next step, improvement can be done in text database preparation, query operation, cosine similarity and term relevance processing.

**References**

1.http://www.cs.memphis.edu/~vrus/teaching/ir-websearch/lectures/IR.intro.session-2.ppt

2.http://www.cs.memphis.edu/~vrus/teaching/ir-websearch/lectures/IR.models.session-5&6.ppt

3.http://www.cs.memphis.edu/~vrus/teaching/ir-websearch/lectures/IR.eval.session-7&8.ppt

4.<https://en.wikipedia.org/wiki/Information_retrieval>

5.<https://en.wikipedia.org/wiki/PageRank>