COMP-8130 PROJECT REPORT

Retrieval Engine for The University of Memphis

Bhavika Khare[1] and Vasile Rus[2]

[1]Department of Computer Science, University of Memphis
[2]Department of Computer Science, University of Memphis

December 9, 2021

Abstract

This project crawls 1000 pages of the University of Memphis website [2] and for a user-given query, retrieves all documents that have any similarity with the query, ranked by the cosine-similarity of their tf-idf weight vectors. The importance of such an initiative is that it makes the workings of a large-scale retrieval-engine like google crystal-clear to even a layperson.

By the end, this retrieval engine was able to perform exactly what it was intended to, but it suffers from the inherent flaws of the cosine-similarity model and the porter-stemming algorithm. It does not understand word context nor does it differentiate between two documents that contain query words in the same proportion but different numbers. It elucidates the concepts of web search and information retrieval, and thus achieves its purpose.

Keywords
Information Retrieval, Web Search, Google, Search Engine, TF-IDF, Bag Of Words

1 Introduction

This project limits its scope to the first 10000 documents it can crawl from the University of Memphis website but is easily scalable to greater scopes, crawling more pages or more websites. It also only crawls HTML, PHP, text or PDF page but can be expanded to crawl other document types.

It uses the bag-of-words model, and tf-idf weights are applied while calculating the cosine-similarity, which is then used for document ranking. This choice of model and similarity-measure is not without its drawbacks but was made for the sake of simplicity.

2 Approach

The first step in the implementation was to write code to crawl a given page and all pages linked from it, and to calculate the Term Frequency (tf) Document Frequency (df) for every word thus encountered.

The second step was to write code to preprocess text files in a given folder. This required the removal of all URLs and HTML code, all characters except alphabets, changing all text to lowercase, removal of all stopwords, and finally stemming of the retrieved text.

The third step was to retrieve 10000 pages crawled from the University of Memphis website, and to preprocess those retrieved pages and save this output into a folder, along with a mapping from the file-name where a preprocessed text was saved to the URL where the text originated from.

The fourth step was to make a reverse-index for this collection of retrieved preprocessed text files, mapping each word to its term frequency in each document and also finding each word's document frequency.

Finally, the fifth and most challenging piece of work in this project was to write code that takes in a query from the user, preprocesses it, and for each document that shares even one word of the query, the cosine similarity score is calculated.

This was done by reading the reverse-index file and the file-to-URL mapping previously created and using them to calculate the tf-idf weights of each word in the documents and the query.

Finally, code was added that saves the results of the search into a text file titled as the query. This wass saved into a separate folder for query results. This is to enable a feature to be added to the project in future so that repeated queries can be served in less time.

Also, a web interface for this was designed in a user-friendly way that took a query from the user and returned a ranked-by-score list of

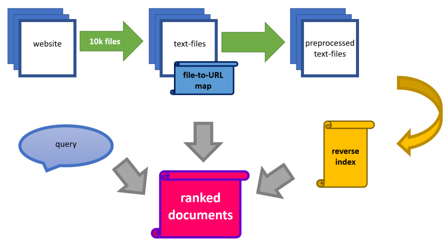

Figure 1: Project Architecture

documents, displaying only their rank, URL, document-number, score and document-length with respect to that query. In case of no relevant documents, such as the case of the query "timothee", a sad emoji is displayed to indicate a failed search where no documents are found.

3 Design

This project limits its scope to the first 10000 documents it can crawl from the University of Memphis website but is easily scalable to greater scopes, crawling more pages or more websites. It also only crawls HTML, PHP, text or PDF page but can be expanded to crawl other document types.

It uses the bag-of-words model, and tf-idf weights are applied while calculating the cosine-similarity, which is then used for document ranking. This choice of model and similarity-measure is not without its drawbacks but was made for the sake of simplicity.

Figure 1 explains the architecture of the project to the layman reader, and describes the flow of information from the website to the user.

4 Implementation

The implementation of this project followed the standard suggested tools on [1].

Strawberry Perl 5.0 was used for programming. The porter-stemmer was used for stemming, and pdftotext software was used for crawling PDFs.

The webpage uses Apache and Perl CGI for the backend.

For backend logic, the bag-of-words model was used and the tf-idf weights applied while calculating the cosine-similarity which was used for document ranking.

5 Results

For the query "Computer Science" this retrieval engine has 0.8 precision and 0.9 recall, which means an F-measure of 0.847.

For queries like "to be or not to be", full of stopwords, the results ought to be null, yet this engine yields plenty of them, since the porter-stemmer reintroduces stopwords into the preprocessed files. Another round of stop-word removal is one way to eliminate this issue.

In conclusion, the retrieval engine does a decent job but has some flaws by design.

6 Future works

The search results would improve in quality if for similar-scoring documents, we rank shorter documents better/higher. Another obvious improvement that would greatly improve the user-experience is the User Interface. A better background image and centered output would be the top priority.

If possible, page preview in the search results would help the user decide which result might be useful.

Also, if previously searched results, which are already stored as text files, could be retrieved, we could cut down on the time taken for retrieval.

Also, using lemmatizers would give better results than the porter-stemmer used here. This stemmer also has the disadvantage of reintroducing undesirable stopwords after stopwords have been removed.

Lastly, this project being available on my personal website would make it accessible to more people, as a project demonstration if not an actual utility.

Acknowledgements

This project has been exciting since its inception and I could not have done it without the easy-to-follow instructions from my professor, Dr. Vasile Rus. He helpfully nudged me in the right direction when I found myself struggling with certain parts of it, such as crawling PDFs or making a web interface.

I am proud of how far I've come in my understanding of web search this semester. I also had a lot of help from my classmates, especially Michael Bowman, Bidhya Shreshtha, Lakshman Sakhamuri, and Sheldon Ebron. The TA Lokesh Das was a reliable source of information regarding the presentation of my work.

References

[1] Vasile Rus, Information Retrieval and Web Search, University of Memphis,

https://www.cs.memphis.edu/ vrus/teaching/ir-websearch/.

[2] The University of Memphis, https://memphis.edu/.