Introduction to Lookit

Liu Jiesong

July 22, 2020

Abstract

Lookit is a search engine, the implementation of which contains web crawling, parsing the page, creating inverted indices and conducting tf-idf weighting. In this document, we provide the specific usage of our application and some brief introduction on how we accomplish all the tasks.

1 Functions of Lookit

As is indicated by the name of our project, Lookit provides functions using information retrieval technologies. To be more specific, users can enter keywords or documents as queries, aiming to find the most 10 relevant documents in a certain website—in our case, it's `http://info.ruc.edu.cn` — and gaining knowledge retrieved and displayed by our search engine. While keywords and documents are both retrievable, we now do not support queries on pictures (i.e. `.jpg` files), pdf files or so(readers of this Introduction can make some adaptations or refinement on our project to support more functions). What's more, we provide the user interface for better searching experience, aiming at making our project more user-friendly and user-feasible. An example of our interface and one certain query experience will be shown in later section.

2 Design of our search engine

The whole design of this project can be divided mainly into 4 parts—web crawling, parsing pages, creating inverted indices and conducting tf-idf weighting, more details of which are as follows. To accomplish this goal, we use a web crawler to get all the web pages.

2.1 Web crawling

To support the final search, raw materials and corpus are needed. To be more concrete, the contents of each page in the targeted website are required to be stored in our database, with which further operation can be conducted.

2.2 Parsing pages

After storing the website pages as files in html format, we need to parse the pages to focus on titles and main contents, leaving out the tags and other information. One document for each page stored in the database.

2.3 Making inverted indices and calculating the tf-idf weighting

Database fixed, we start to deal with the queries. Initially, we read the documents and cut sentences into words so that a sentence can be converted into a sequence of words before we create inverted indices for each word in each file stored in our database to facilitate the following searching. Next, we calculate term frequency($tf_{t,d}$) of each word in one document—the total appearance of the word t in the document d—and document frequency(df_t) of each word (we call it a term) — the total number of documents containing the term t—to indicate the significance of one certain term. Consequently, each document can be converted into a vector, of which the comprised elements are as follows:

$$w_{t,d} = (1 + log_{10} tf_{t,q}) * log_{10}(N/df_t) \text{ for each selected t}$$

with N representing the total amount of documents. Similarly, each query — whether it contains certain keywords or a document—can be represented by a vector using the according elements. We then use Vector Space Model(VSM) to evaluate the relevance of the query and certain documents. Finally, we retrieve the top 10 documents for each query.

3 Implementation Features of our search engine

Store files to save loading time Loading documents into programs and cut sentences into word sequences can really take A LOT OF time. However, for the most of the time, we do not need to refresh our database since the raw materials and documents are fixed—unless we change the starting website. Consequently, we store the documents that are read from local files and cut into words sequences in a compressed file so that each time when initializing the project we can start creating inverted indices directly with a loading time of less than 5 seconds.

Pay more attention to the titles When we write articles, we tend to figure out the most important information and this is how an article title is set. Based on this feature, we assume that if a query word is appeared among the title of one certain page, this page is more likely to be the targeted page users are searching for. We take this assumption into consideration and our search engine brings better outcomes and performance.

4 Conclusion

Lookit is expected to facilitate users' searching experience with its robust capability and fast speed. Though still in its initial stage, it comes with satisfactory searching accuracy and efficiency. Of course, there are some fields in want of improvement such as fuzzy search. However, we are still confident about its bright future.

A Appendix

A.1 Usage of Lookit

A.1.1 Web crawling

In this part, readers can crawl all the pages in `http://info.ruc.edu.cn` by simply using the following command:

```
make
```

Then the crawling process will start. It will take about 20 minutes to finish crawling all the pages in `http://info.ruc.edu.cn` and the contents of these pages will be stored in the corresponding resulting files located in the directory named `web`. In particular, this is a C++ project.

A.1.2 Parsing pages

In this step, we extract the title and main contents of one specific page by executing the program `parser.py` .

Just use the following command:

```
python3 parser.py
```

and the main contents of each page are stored in the directory named `info`.

A.1.3 Creating inverted Indices and retrieve information for each query

After parsing all the pages, we can finally do the indexing job. We make inverted indices and deal with each query. Additionally, we use flask to run our search engine, so readers can move to `127.0.0.0:5000` to see our UI.

Just use the following command:

```
python3 search.py
```

and Lookit — our search engine — get started. In particular, we will ask whether to refresh the database. For the first time, please choose 'yes'. In later tests, if nothing changed with the pages stored in `info`, please select 'no'. Move to `127.0.0.0:5000` then the user interface will be displayed.

A.1.4 Enter queries and gain knowledge

Seeing the user interface, users can enter the keywords or documents for search. Lookit is expected to provide 10 resulting pages that are relevant to the keywords or documents.

B More query examples

There are a few more example queries.

Figure 1

Figure 2


我院师生论文被CCF-A类会议ICML录用
Lookit
我院师生论文被CCF-A类会议ICML录用 - 学院新闻 - 中国人民大学信息学院
http://info.ruc.edu.cn/news_convert_detail.php?id=1780
院新闻近日，信息学院师生论文被CCF-A类会议ICML（2020）录用。ICML（国际机器学习大会，
新闻公告 - 中国人民大学信息学院
http://info.ruc.edu.cn/news_list.php?page=1
我院程絮森教授在信息系统领域顶级期刊JMIS发表论文2020-06-19我院师生论文被CCF-A类会
新闻公告 - 中国人民大学信息学院
http://info.ruc.edu.cn/news_list.php
我院程絮森教授在信息系统领域顶级期刊JMIS发表论文2020-06-19我院师生论文被CCF-A类会
中国人民大学信息学院
http://info.ruc.edu.cn/index.php
我院杜小勇、王珊教授牵头申报的成果“数据库管理系统核心技术的创新与金...
卢志武 - 师资科研 - 中国人民大学信息学院
http://info.ruc.edu.cn/academic_professor.php?teacher_id=30
办事指南 交流简报 交流心得 院友新闻 院友

Figure 3