Insert Your Title Here∗

Insert Subtitle Here

FirstName Surname†  
 Department Name  
 Institution/University Name  
 City State Country  
 email@email.com

FirstName Surname  
 Department Name  
 Institution/University Name  
 City State Country  
 [email@email.com](mailto:email@email.com)

FirstName Surname  
 Department Name  
 Institution/University Name  
 City State Country  
 email@email.com

3 Development of a Simple Search Engine

In this section, we use Lucene to develop a search engine with single keyword query and phrase query capabilities. In general, the program has two steps, which are creating index and searching.

3.1 Creating Index

Firstly, we need to read each line from the JSON file and parse it with the third-party library fastjson.

Secondly, we need to build the index of review dataset and configure which filed shall be indexed and searchable. After consideration, we set all the field values to be storable and indexable. More specifically, for the field of review\_id, user\_id, business\_id and date, they are retrieved as a whole and therefore do not need to be tokenized. For the field of text, the tokenized process is required, thus using the TextField to initialize it. For the type of each field in the document is configured as shown below.

Text

Description automatically generated

Figure 1: Configuration of Each Field

In the linguistic process of creating index, we need to perform stemming and case folding by using our own StemAnalyzer. The class of StemAnalyzer can also take an analyzer (implemented by Lucene) as a base analyzer and add a StemFilter to it to implement stemming functionality. The Key code for StemAnalyzer is as follows:

Graphical user interface, text, application, email

Description automatically generated

Figure 2: Definition of StemAnalyzer

After configuring the linguistic process, we need to use an index writer with the config of the StemAnalyzer which is mentioned early. At last, we call the relevant method of the index writer to add the document line by line.

Base on the amount of data in the review dataset, a total of 8635403 documents need to be created and the time taken to create each 10% of the number of indexes is shown below.

Table

Description automatically generated with medium confidence

Figure 3: Time of Indexing Every 10% of Documents

As we can see above, the average time of indexing every 10% of documents is about 28 seconds.

3.2 Development of Search Engine

In the process of searching, we need to perform stemming, case folding and stop word removing. Therefore, we have to combine the StemAnalyzer with StandardAnalyzer which is provided by Lucene. In the class of StandardAnalyzer, it performs the lower case folding and stop word removal by using LowerCaseFilter and StopFilter. Specifically, the source of the list of stop words is ENGLISH\_STOP\_WORDS\_SET which contains about 40 common English stop words. What’s more, we also need to perform stemming for the input keyword in order to keep it recognizable in the index.

In order to support single keyword query and phrase query, we need to construct these two types of queries, and the key code is as follows:

Graphical user interface, text, application

Description automatically generated

Figure 4: Construction of Different Queries

As we can see above, the phrase query supports the input of phrases with a word spacing of 0. If the phrase slop is set larger, theoretically more phrases whose words are not directly adjacent to each other can be matched, but the search time will increase accordingly.

3.3 Searching Results

In this part, we simply try some single keyword queries and phrase queries, and we also need to record the time taken to process these queries.

*3.3.1 Single Keyword Query.* After selecting the single keyword searching, we type the word “accessories” via the console and wait for the results. The query time is about 0.2 sec, and the number of matched documents is 16257. Since we set the top N parameter to 10, the query returns only the top 10 documents in terms of score. Interestingly, if we search the word “accessories” again, the query time will be reduced to only 0.047 sec.

*3.3.1 Phrase Query.* After selecting the Phrase searching, we type the phrase “take care of” via the console and wait for the results. The query time is about 0.22 sec, and the number of matched documents is 47772. Among the top 10 results, the phrase appears 2 times in the best matched result, while the phrase in the fourth matched result is “taking care of”, and the scores of these results are quite close to each other’s. Same as before, we try to search this phrase again and the retrieval time is reduced to 0.076 sec, which is way faster than the first time.

Overall, the results returned by this simple search engine are as expected. In addition, to see the specific results, please refer to the readme file.

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00