COMP38210 Lab 1 Report
Documents and Data on the Web

Shurong Ge

24/11/2020

1. Introduction

This program develops inverted index function using Hadoop, the input is six Wikipedia articles, and the output is the word mapping its location in the document. The detailed functionalities that the program implements are described below.

2. Functionality

Basically, this algorithm enables users to search particular words, it will return the name of the documents that the word occur and its position in the document.

2.1 Punctuation Removing / Non-alphabetic Characters Removing

Using replaceAll() method to remove all the punctuation, for example hanging" will become hanging after this method, which helps the index to be more clear.
However, it will cause problem when the token is a decimal fraction, for instance, 10.3 will become 103, which changes its original meaning.

2.2 Case Folding

As we know, sentences start with capital letter in English. In other words, a word with the first letter in upper case is probably the first word of a sentence so we can turn all terms to lower case; and if the capital letter is in the middle, it may because of mistyping, we can still turn it to lower case. While a word with all terms in upper case should remain the same, for example BBC.
The limitation is that my caseFolding() method cannot detect names, for example Bill, it will become bill which may lead to misunderstanding.

2.3 Stopword

The package already got a function called isStopWord(), it will filter out those most common words such as and, about, which saves time when searching.

2.4 Stemming

Stemming will remove the prefixes and suffixes of the word in order to get the root of the word. This algorithm is also provided in the given code.
After stemming, it is not always produced the valid word, taking active as an example, it will become activ which is not a real word.

2.5 In-Mapper Aggregation

It is likely that one word occurs more than one time in a document, therefore we can categorize all identical words into one, so that the jobs for reducer can be reduced and the running speed will increase.

2.6 Positional Indexing

(Attempted. Using VirtualBox to do my coursework, but it will suddenly frozen forever after about 10 minutes and it is very slow when I use it so I do not have enough patient to finish it. The remaining problem is in map function: the context.write() method cannot allow to print my arraylist which contains filename and position.)
Positional indexing reflects the location of the token in the documents, and we can also know the token frequency from it. For example, the input is ‘Apple loves eating apple’, we will get apple (file_name, [0, 3]), it not only tells us where ‘apple’ occurs but also shows that ‘apple’ occurs twice in this file.

2.7 TFIDF

(token i and document j)
TF = number of occurrences of i in j / total number of tokens in j
IDF = log(total number of documents / number of documents containing i + 1)
TF - IDF = TF * IDF
The greater the TF-IDF of a word in an article, the greater the importance of that word in the article in general, so by calculating the TF-IDF of each word in the article, sorted from largest to smallest, the top few words are the keywords of the article.

3. Performance

The run-time complexity of each operation is O(n) in relation of the length of the input.

3.1 Design Patterns

Summarization pattern - The program summarizes the same and repeated words which saves a lot of memory on the disk and increases the running time.

Input and output pattern - Hadoop allows us to change how the data is loaded on disk. In my program, a combiner (semi-reducer) is added between mapper and reducer (input-mapper-combiner-reducer-output), which means the workload of reducer will be reduced, and the time required to run is also reduced.

3.2 Limitation Factors

Language - My program is more concerned with searching documents in English, e.g., it only takes into account that English is written from left to right.

Small files - Since Hadoop prefers a few large files, it will cause trouble if there are many small files, it will take more time to retrieve them.

4. Conclusion

In conclusion, my MapReduce program is able to realise basic invert indexing function, and clease the input data by stemming, case folding, and removing stopword as well as punctuation.

I learn a lot from this lab. It not only shows how indexing program runs using Hadoop, but also helps us to get deeper understanding of how search engines deal with the data.

5. References

Donald Miner and Adam Sbook, *MapReduce Desgin Patterns*, O'Reilly, 2012.