**Simple Search Engine in Spark**



**Made by:**

**Sobirdzhon Bobiev**

**Beni Zakaria Ar Ridlo**

**Utih Amartiwi**

**Big Data Technology and Analysis Class 2019**

**CHAPTER I**

**INTRODUCTION**

1. **Background**

Since it was built in 2009, Apache Spark has emerged as the next generation big data processing engine [1].It is faster, highly accessible, and can integrate with other Big Data tools. In this project, we want to build a simple search engine using Spark. A fundamental goal of search engine is to identify some relevant documents. As number of files grows significantly, potentially relevant document identified by search engine also increase. Search engine can store some documents even it only contains 1-3 words that similar with user’s keywords. Thus, the users have to check again whether the information is really they need or not. It is difficult and takes much time to check it one by one.

In order to improve the relevancy of search result, we need to make an index of the similarity of the documents and rank them. The most common index used is TF/IDF weight that using frequency of term in indexing. The indexing result will be used to calculate the rank of similarity. There are some of rankers, but 2 most popular rankers are Inner Product and Okapi BM25. Inner product ranker comes from vector space theory that identifies the distance between vectors as the similarity of files. On the other hand, Okapi BM25 computes the similarity using weight between document and query based on term probability [2]. In some cases, inner product shows better result than BM25 [3] but in other cases are vice versa [2]. That motivates us to learn more the difference result of both ranker. Therefore, we will implement both rankers and find the best ranker for our search engine.

1. **Objective**

The aim of this project is to learn how to build a good search engine. Here we will implement big data theory from lecture and the programming skill using Spark. Furthermore, to provide the user the right information they need, we have to use the best ranker. Therefore, we will compare the result of inner product and BM25 using Mean Average Precision (MAP) as the metrics. We will analyze and determine the best one of them in our case. All in all, we hope we can get new insight from problem and result during the process and share it to the reader.

**CHAPTER II**

**PROCESS**

1. **Methods and Steps**
2. Indexing Engine

First, all the words from text corpus will be enumerated in Vocabulary. For each term, we give a unique ID that represents it. In each document *d*, we calculate the frequency of occurrence of term *t* in document *d* that called **Term-Frequency (TF)**. We also calculate the number of documentsthat contain term *t*. It is called **Inverse Document Frequency (IDF).** IDF(t) also can be calculated:

Where N is number of documents and n(t) the number of documents containing the term *t*. We calculate document weight (**Wd**) by:

Then we transform all terms in each document as format: (,).

1. Ranking Engine

After we do indexing the documents, we process the query. Here we calculate the weight of query (**Wq**) using formula above. Then we create a vectorized representation for the query with format: (,). Suppose is a term in query and is a term in document *d,* if we calculate the relevance analysis:

1. Inner Product

Suppose the document and query to be vectors **d** and **q** respectively. The inner product computes the distance between vectors:

is the relevance of document **d** with respect to query **q**.

1. Okapi BM25

Okapi BM25 computes a weight between document and query based on term probability.

and are assumed as same as TF and IDF in indexing engine. **Avgdl** is the average document length in the corpus, **|d|** is the size of the current document, **b** and **k1** are free parameters that will be set to 0.75 and 2.0 respectively.

1. Query Response

If , it means the document doesn’t contains the same term with query so it will not appear in Query Response. If it means the document has the relevant term with query. Query response will list some relevant documents and appear in order based on their rank.

1. Testing and Evaluating

Precision is the proportion of search result is relevant to the user. Mean Average Precision (MAP) for a set of queries is the mean of the average precision scores for each query.

Where **Nrel** is the number of relevant document returned by the ranker, **P(k)** is the precision calculated for the positions from 1 to k, **rel(k)** is an indicator function that is equal 1 when document at position k is a relevant document, and 0 otherwise. Here we calculate MAP of the result from both inner product ranker and BM25 ranker. We compare it and analyze whether their difference is significant or not and which one better than other using *t-test*.

1. **Chart of Steps**
2. **Tasks Distribution**

**CHAPTER III**

**DISCUSSION OF RESULT**

**CHAPTER IV**

**CONCLUSION**

**APPENDIX**

**REFERENCES**

|  |  |
| --- | --- |
| [1] | M. Zahara, H. Karau, A. Konwinski and P. Wendell, Learning Spark, Sebastopol: O’Reilly Media, Inc., 2015. |
| [2] | A. Trotman, "Learning to Rank," *Information Retrieval,* pp. 359-381, 2005. |
| [3] | A. Trotman, "Choosing Document Structure Weight," *Information processing & management,* pp. 243-264, 2005. |