Distributed BM25

Leung Shing Yuet

The Chinese University of Hong Kong

1. INTRODUCTION

In 21st Century, big data processing and analyzing is a hot topic and having an efficient and expressive data computing system is very important. Hadoop and Spark are some widely used systems for massive data processing and they are easy to use. However, the over-simplified API may not be efficient enough so the users may need to use other domain-specific systems which raises the development cost. This motivates Prof James Cheng and his team to develop the Husky[2], an open-source system, which aims to build a more expressive and most importantly, more efficient system for distributed data analytics.

There are three main features in Husky, fast, general and easy. Husky computing model allows more efficient algorithms to be programmed and has a highly optimized backend. Husky also supports a variety of applications including text mining, graph analytics and machine learning. Moreover, Husky has an easy-to-use interface and even non-technical people can get started with Husky quickly with the Python/Scala frontends. Besides, there are three main components in Husky, master, coordinator and worker. Master represents an application, coordinator manages the workers and worker represent how CPU cores work with their associated data.

To be specific, in this paper, the BM25 ranking function on Husky will be described, in the way of implement, the discussion and then followed by a conclusion. BM25, also known as Okapi BM25, is a ranking function used by search engines to rank matching documents according to their relevance to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others.

2. IMPLEMENTATION

Given a query Q, containing keywords $q_1, \ldots, q_n$, the BM25 score of a document D is :

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i,D) \cdot (k1+1)}{f(q_i,D) + k1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

Where $f(q_i, D)$ is q_i's term frequency in the document D, $|D|$ is the length of the document D in words, and avgdl is the average document length in the text collection from which documents are drawn. $k1$ is the term frequency saturation and b is the field-length normalization. They are usually chosen as $k1 \in [1.2, 2.0]$ and $b = 0.75$. $\text{IDF}(q_i)$ is the IDF (inverse document frequency) weight of the query term q_i. It is usually computed as

$$\text{IDF}(q_i) = \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}$$

where N is the total number of documents in the collection, and n(q_i) is the number of documents containing q_i.[1]

Steps:

Suppose we have a text collection with N documents, input a query Q with i terms

1. read the documents and store the information of each document, including document id and content.
2. read the query Q and split Q into and store in a vector
3. for each document D, split the content into words and compare with q_i, count the no. of q_i appear in D to get $f(q_i, D)$, count the no. of words in D to calculate $|D|$, count the no. of docunments that q_i appear to get n(q_i).
4. calculate the IDF score of each q_i.
5. calculate the $\text{score}(D, Q)$ by summing up $\text{score}(D, q_i)$

Users can choose to output the top k documents with their document id and scores or output the scores of some documents by inputting the document id.

3. DISCUSSION

After finishing the implementation, the BM25 can run on the C++ Husky. After that, the BM25 implementation is wrapped to the PyHusky to be more user-friendly and more API can be used.

4. CONCLUSION

We can see the BM25 can be implemented on Husky easily and after binding to PyHusky, it can work with other API as well so that it can be more user-friendly.

Acknowledgments

I would like to express the deepest appreciation to my supervising Prof. Cheng and his Ph.D. students for their supervision and constant support in the research period.

5. REFERENCE

[1] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[2] F. Yang, J. Li, and J. Cheng. Husky: Towards a more efficient and expressive distributed computing framework. *PVLDB*, 9(5):420–431, 2016.