Azerbaijan Journal of High Performance Computing, Vol 4, Issue 2, 2021, pp. 263-266
https://doi.org/10.32010/26166127.2021.4.2.263.266

A Survey of Retrieval Algorithms and Their Parallelization in Large-Scale Systems

Suleyman Suleymanzade

Institute of Information Technology, Azerbaijan National Academy of Sciences, Baku, Azerbaijan, suleyman.suleymanzade.nicat@gmail.az

*Correspondence: Suleyman Suleymanzade, Institute of Information Technology, Azerbaijan National Academy of Sciences, Baku, Azerbaijan, suleyman.suleymanzade.nicat@gmail.az

Abstract

This article presented a survey of two well-known algorithms, TF-IDF and BM-25 methods, for document ranking on a single CPU and parallel processes via HPC. An amazon review dataset with more than two million reviews was measured to measure the rank parameters. We set up the number of workers for the parallel processing during the experiment, which we selected as one and three. Four benchmarks evaluated the preprocess and reading time, vectorization time, TF-IDF transformation time, and overall time. Results metrics have shown a significant difference in speed.

Keywords: TF-IDF, BM-25, Apache spark, Information retrieval, HPC

1. Introduction

One of the main challenges for searching (Kumar, R., & Sharma, S. C., 2018; Ramli, F., Noah, S. A., & Kurniawan, T. B., 2016, August; Dietz, L., Xiong, C., Dalton, J., & Meij, E., 2019). information on the Internet is the large amount of available data, from which users must extract desired content within multiple links. This led to fundamentally new approaches and strategies for search engines. To create a successful search system, several problems that arise at different levels (Zheng, P., Wu, Z., Sun, J., et al., 2021) and stages of system creation must be solved. Moreover, there must be research for optimizing these methods to use them in HPC (Lawson, M., Gropp, W., & Lofstead, J., 2021). The information retrieval system uses well-ordered queries from a structured database (Järvelin, K., 2007), which must meet the needs of users' information resources. For ordering such data, the search engine also includes document ranking methods. The methods will introduce two well-known methods, such as TF-IDF and Okapi algorithms. Then, there will be experiments for page ranking in parallel by using the Apache Spark framework and comparing results with single cluster approaches.

2. Methods

TF-IDF

TF-IDF (TF - term frequency, IDF (Robertson, S., 2004; Metzler, D., 2008, October; Schütze, H., Manning, C. D., & Raghavan, P., 2008) - inverse document frequency) is a ranking function used to evaluate the priority of a word in a document. Documentation in the original is available in collections of documents (corpus), where a formula defines TF.

$$tf(t,d) = \frac{n_t}{\sum_k n_k}$$

Where n_t is the number of occurrences of the term (word) t in document d, $\sum_k n_k$ - is the total number of words in D set of documents.

$$idf(t,D) = \log\frac{|D|}{|\{d_i \epsilon\ D\ |t\ \epsilon d_i\}|}$$

$|D|$ - documents number,

$|\{d_i \in D \mid t \in d_i\}|$ number of documents from document set D where t term is presented.

The weight of a word is proportional to the frequency of occurrence of this word in the document and inversely proportional to the frequency of occurrence of the word in all documents in the collection. Usually, the base of the logarithm is chosen equal to ten, but this does not play a difference because the ratio of all words remains the same

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D)$$

This method is a classic approach for ranking documents; however, to detect rare terms that are synonyms of more common words, the TF-IDF method may not be productive; for this, extended versions of the TF-İDF algorithm are usually used, where a dictionary of synonyms is selected as a preprocessing for calculating weights " Synonyms Based Term Weighting Scheme: An Extension to TF.IDF" (Kumari, M., Jain, A., & Bhatia, A., 2016), in this work, it was proposed to create a cluster domain with a set of synonyms for words called "Synonyms-Based Term Weighting Scheme" (SBT) estimated by the formula.

$$SB - TFIDF = TF * (SB - IDF)$$

Where SB is the set of the synonyms.

Experiment 1

The experiment Amazon Customer (Mudambi, S. M., & Schuff, D., 2010) dataset was taken with more than 10 million reviews. The data preprocessing included stemming and lemmatization. Word embedding was produced by hashing vectorizer (Tito Svenstrup, D., Hansen, J., & Winther, O., 2017; Argerich, L., Zaffaroni, J. T., & Cano, M. J., 2016), then TF-IDF was calculated corresponding to each review. Data calculation was produced on Nvidia GTX 1660Ti (Krishnan, A. G., & Goswami, D., 2021, December) as seen in the result table below the parallel approach (Mezzoudj, S., Behloul, A., Seghir, R., & Saadna, Y., 2021) with Apache Spark, which gave 3x times more speed performance concerning the single CPU approach for reading, the word embedding by hashing gave 3.18 speed up. In row processing 2.5 times and computation of TFIDF 2.29 times with three parallel processes.

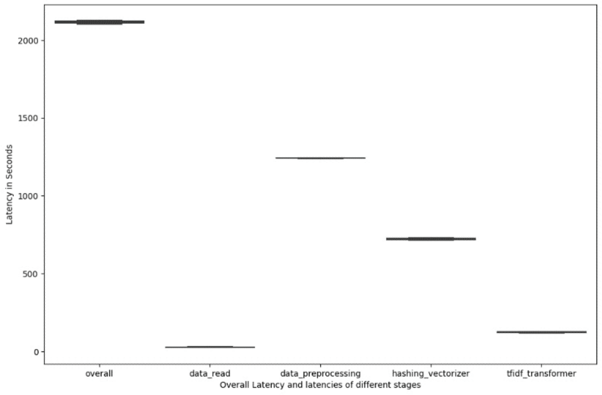
overall	Reading	Hashing	TfIdf trans-former	preprocess	rows	work-ers	run
2123.623687	26.286626	731.965263	1123.351882	1241.842862	2050377	1	1
Parallel with Apache Spark							
818.24355	8.542342	230.478812	489.765313	489.765313	2050377	3	1

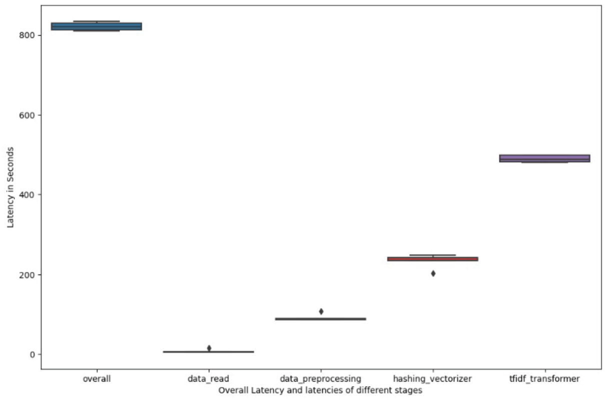
BM-25

BM-25 (Lv, Y., & Zhai, C., 2011, October) is another ranking function based on $B11$ and $B15$ (Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M., 1995). In the Q query, the $BM-25$ function is assembled from the words $q_1, q_2 \dots q_n$ to evaluate the relevance of document D to the Q query:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i)\,\frac{f(q_i,D)(k_1+1)}{f(q_i,D) + k_1\,(1 - b + b\,\frac{|D|}{avgdl})}$$

Where $f(q_i, D)$ is the frequency of word q_i in document D, $|D|$ - length of the document, $avgdl$ - the average length of the document in the collection k_1 and b - free





coefficients, in practice usually $k_1 = 2.0$ and $b = 0.75$.

Experiment 2

For the second experiment, the same amazon review [11] dataset was selected, and the preprocess and hashing were done in the same way as in the first experiment, but the preprocessing took a long time because of the more complicated model. The results show that BM-25 computed 1.36 times faster than on a single CPU with three parallel processes.

Overall	Reading	Hashing	BM25 trans-former	Preprocess	Rows	Work-ers	Run
2123.623687	27.156626	723.343562	315.255326	1934.984372	2050377	1	1
Parallel with Apache Spark							
1211.53112	8.224242	251.254812	230.330112	599.356313	2050377	3	1

Conclusion

For both experiments, the results of the parallel approach show almost n-th times increasing the speed of computation depending on the preprocessing stages. The less scalable part is the vectorization part in the algorithms and transformation phase in the BM-25 example.

References

Argerich, L., Zaffaroni, J. T., & Cano, M. J. (2016). Hash2vec, feature hashing for word embeddings. *arXiv preprint arXiv:1608.08940.*

Dietz, L., Xiong, C., Dalton, J., & Meij, E. (2019). Special issue on knowledge graphs and semantics in text analysis and retrieval. *Information Retrieval Journal, 22*(3), 229-231.

Järvelin, K. (2007). An analysis of two approaches in information retrieval: From frameworks to study designs. *Journal of the American Society for Information Science and Technology, 58*(7), 971-986.

Krishnan, A. G., & Goswami, D. (2021, December). Multi-Stage Memory Efficient Strassen's Matrix Multiplication on GPU. In *2021 IEEE 28th International Conference on High Performance Computing, Data, and Analytics (HiPC)* (pp. 212-221). IEEE.

Kumar, R., & Sharma, S. C. (2018). Information retrieval system: An overview, issues, and challenges. *International Journal of Technology Diffusion (IJTD), 9*(1), 1-10.

Kumari, M., Jain, A., & Bhatia, A. (2016). Synonyms based term weighting scheme: An extension to TF. IDF. *Procedia Computer Science, 89,* 555-561.

Lawson, M., Gropp, W., & Lofstead, J. (2021). Exploring Spatial Indexing for Accelerated Feature Retrieval in HPC. *arXiv preprint arXiv:2106.13972.*

Lv, Y., & Zhai, C. (2011, October). Adaptive term frequency normalization for BM25. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 1985-1988).

Metzler, D. (2008, October). Generalized inverse document frequency. In *Proceedings of the 17th ACM conference on Information and knowledge management* (pp. 399-408).

Mezzoudj, S., Behloul, A., Seghir, R., & Saadna, Y. (2021). A parallel content-based image retrieval system using spark and tachyon frameworks. *Journal of King Saud University-Computer and Information Sciences, 33*(2), 141-149.

Mudambi, S. M., & Schuff, D. (2010). Research note: What makes a helpful online review? A study of customer reviews on Amazon. com. *MIS quarterly,* 185-200.

Ramli, F., Noah, S. A., & Kurniawan, T. B. (2016, August). Ontology-based information retrieval for historical documents. In *2016 Third International Conference on Information Retrieval and Knowledge Management (CAMP)* (pp. 55-59). IEEE.

Robertson, S. (2004). Understanding inverse document frequency: on theoretical arguments for IDF. *Journal of documentation.*

Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. *Nist Special Publication Sp, 109,* 109.

Schütze, H., Manning, C. D., & Raghavan, P. (2008). *Introduction to information retrieval* (Vol. 39, pp. 234-265). Cambridge: Cambridge University Press.

Tito Svenstrup, D., Hansen, J., & Winther, O. (2017). Hash embeddings for efficient word representations. *Advances in neural information processing systems, 30.*

Zheng, P., Wu, Z., Sun, J., et al. (2021). A parallel unmixing-based content retrieval system for distributed hyperspectral imagery repository on cloud computing platforms. *Remote Sensing, 13(*2), 176.

Submitted: 10.09.2021
Accepted: 29.11.2021