Thomas Cheal

11/7/2021

CS410

Literature Review:

An Examination of BM25 Variants

Intro

` Ever since BM25 was introduced to the world by Robertson et al.[4], people have been clamoring to create even stronger algorithms that improve upon the original design in several different areas. Many people consider BM25 to have a fault in one particular area, and that's penalizing long documents far too much, so many algorithms such as BM25+ or BM25L attempt to fix this[1]. Other algorithms, such as BM25-adpt try to change the algorithm in a more complex way in an attempt to increase precision and recall of the algorithm. And finally, there are also algorithms such as TF$^{log(op)}$IDF attempt to further improve on the idea that long documents are penalized by changing the way individual terms are counted[2]. This paper will attempt to explore these three areas, specifically talking about BM25+, BM25-adpt, and TF$^{log(op)}$IDF and comparing these to the original BM25 implementation.

Body

BM25+ is an interesting variation on the original algorithm, introducing a lower bound for term frequency (TF) normalization[1], which helps to alleviate the issue of over penalizing long documents. This is one of the most well known and generally considered one of the most effective variation on BM25, as it does indeed fix one of the issues with the original BM25 algorithm. According to the original paper, it performs significantly better than the original algorithm. This is indisputably true, however it does not large impact on the overall performance of the algorithm. That being said, it is also true that implementing the changes does not incur a significant computational cost increase[1], so it is therefore almost always implemented over the original algorithm, as it is simply an improvement.

BM25-adp takes a different approach to solving issues within the BM25 algorithm. It attempts to use term specific k1 values[1]. K1 is one of the parameters that is generally given to the BM25 algorithm, but in this case BM25-adp attempts to calculate an individual k1 value for each term. Even though this seems like it would be a major breakthrough in information retrieval, this calculation of k1 is not actually prevalent within the algorithm because k1 can only be calculated if certain conditions are met. If those conditions are not met, the k1 value is set to a default value. Due to this fact, BM25-adp does not cause a major change in effectiveness for BM25, although it does induce more computational cost[3]. It is still generally considered to be more effective than the original algorithm.

The last algorithm to discuss is TF$^{log(op)}$IDF, which is an expansion upon BM25+. Indeed, it is common for these variations to have even further variations being researched upon,

every avenue is being exhausted in an attempt to create a better algorithm. This algorithm changes the way terms are counted to try to alleviate the problem of long document penalization. It is the only variation that is widely used that has a double logarithmic implementation, creating a non-linear model of term counts. This is an attempt to curve the algorithm against long documents without penalizing them so harshly that they will never be selected[2]. While this approach does show a significant improvement over BM25, it is not considered significantly more effective than BM25+, which was the goal of the algorithm.

Comparing these three algorithms is difficult, as they all explore different ways to achieve slightly different goals. Of course, BM25+ is generally considered the best way of fixing the long document penalization issue, but certainly TF$^{log(p)}$ IDF also works very well. BM25-adp has potential to be a very successful algorithm if it could apply its changes more liberally, but the limitations of the algorithm allow it to only mildly change how the algorithm behaves and can therefore not make a significant improvement upon the original construction. In the modern day, it seems practical to choose any one of these three implementations over the original BM25 algorithm, as all three of these do make solid improvements that are generally not more difficult to implement or significantly more computationally expensive. The Lucene implementation, which this paper did not discuss, is the standard implementation that open source developers use, as it is free and already implemented generally. Most regard the Lucene implementation to be better than the original BM25 but less effective than many other variations.

Conclusion

To sum it up, there have been numerous improvements upon BM25, but none of them offer a major breakthrough in the technology. There are plenty of other variations that were not discussed, such as ATIRE, BM25L, or the Lucene implementations[1]. All of these offer slightly different algorithms that all have their pros and cons. Of course, major companies also likely have their own implementations of BM25 or perhaps other search engine algorithms that one can only assume are different and perhaps even more advanced than the publicly available implementations. It is also difficult to test search engine algorithms in a significantly large algorithm when compared to production environments, because of the massively widespread use of search engines, there are so many users and so many different cases to consider. That being said, it is generally acceptable that even on a small dataset you can compare different algorithms, even if you cannot really get a solid idea of how any algorithm will perform in a production environment.

References

1. Kamphuis, C., de Vries, A. P., Boytsov, L., & Lin, J. (2020). Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants. *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II, 12036*, 28–34. https://doi.org/10.1007/978-3-030-45442-5_4

2. Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and Language Models Examined. In *Proceedings of the 2014 Australasian Document Computing Symposium* (*ADCS '14*). Association for Computing Machinery, New York, NY, USA, 58–65. DOI:https://doi.org/10.1145/2682862.2682863

3. Lv, Y., & Zhai, C. (2011). Lower-bounding term frequency normalization. In *CIKM'11 - Proceedings of the 2011 ACM International Conference on Information and Knowledge Management* (pp. 7-16). (International Conference on Information and Knowledge Management, Proceedings). https://doi.org/10.1145/2063576.2063584

4. Robertson, S.E., Walker, S., Jones, S., Hancock-Beaulieu, M., Gatford, M.: Okapi at TREC-3. In: Proceedings of the 3rd Text Retrieval Conference (TREC-3), pp. 109–126, Gaithersburg (1994)