The Okapi BM25

Joshua Wu

The Okapi BM25 is one of the most popular ranking functions used for information retrieval today. Originally developed in the 1970's by Robertson et al., the BM25 algorithm calculates the relevance of documents to a search query to an impressively accurate degree. The BM25 functions by combining three vital parts of a document: term frequency, document frequency, and document length. It utilizes these in a clever manner to rank documents and provide the most relevant information to the user. However, when people speak of using BM25, it isn't always clear which algorithm they're referring to. While the original BM25 by Robertson is the base, there are multiple derivatives of BM25, each with their own unique adjustment to the scoring function.

We begin with the original BM25 algorithm. As mentioned previously, the core of every BM25 algorithm is a document's term frequency, document frequency, and document length. In the original algorithm, BM25 accounts for term frequency and document frequency using TF-IDF weighting. In general, words that appear more often in a document will have more weight, but common words that are regularly used in the English language have less weight. Additionally, documents that have a longer total length relative to the average document length are penalized, whereas shorter documents are rewarded. The BM25 combines these three components together and gives a final weight to a document relating to the search query - the higher the score, the better. The algorithm then can rank the documents in order to provide the user the best possible results for their query. When users use the BM25 algorithm, such as the implementation in metapy, we typically pass in up to three parameters. k1, which is used for term frequency weight, b, which is used for document length normalization, and k3, which is used as a user-specified weight. Users are free to use any values they so choose, however typically k1 is chosen to be 1.2 and b is set to 0.75 for the best results. Ultimately, the creation of the BM25 algorithm was groundbreaking, and had immense effects on the field of text retrieval. As proof, it's core concepts are still being used today, decades later. Now, we will take a look at some of the many BM25 variants that are used today.

The BM25F was developed to improve results with structured documents. A structured document is when a document's individual parts are marked with an identifier. For example, a section can be labeled as the introduction, or a section can be labeled as the body, and so on. The original BM25 algorithm doesn't take into consideration the structure of a document, only the words that appear in it. Created by Robertson and Zaragoza, BM25F aims to fix that. It does so by weighing term frequencies depending on the importance of the field that the term is in. It then combines these term frequencies and uses those in the traditional BM25 algorithm. For example, it could weigh terms shown in the title section with a much higher weight. By applying different weights for the different fields and using the new pseudo-frequencies of terms, BM25F is able to provide better search results for documents with structures.

Tackling a different issue, BM25L is an elegant solution to a problem that affected BM25: long documents are unfairly punished, by far too much. Developed by Lv and Zhai, this simple extension of BM25 adjusts the term frequency normalization formula in order to score long documents higher, with no adjustment to the cost of computation. It primarily accomplishes this by adjusting the IDF component and adding a constant, delta, to the c'(q,D) component, which contains the document length normalization. The authors of BM25L suggest setting the constant to 0.5 for the best results. This constant improves the score of longer documents in a straightforward manner, and as such, no longer penalizes very long documents. While the change is minor, the impact is noticeable, and gives longer documents a much fairer chance when getting ranked.

BM25+ builds upon BM25L, solving a very similar problem on the opposite side of the spectrum. Whereas BM25L focused on reducing punishment for long documents, BM25+ attempts to reduce the reward that shorter documents receive. It accomplishes this goal by providing a lower-bound to TF weighting. The original BM25 algorithm does not properly lower-bound the term frequency component combined with document length normalization. As such, long documents which match a query term can end up having a similar score to a shorter document that doesn't match the query term. BM25+ changes the algorithm such that when a term occurs at least once in a document, it is given a lower bound to the bonus that it receives. The algorithm accomplishes this by adding another constant, delta. This constant is added to the entire term frequency and document length normalization component. This also comes with a slight adjustment to the IDF component, so that the final weight cannot be negative. By adding this lower bound, BM25+ offers another solution for the imbalance between long and short documents.

Finally, we discuss Lucene, which focuses on a completely different problem than any other BM25 variant here. Lucene has multiple variants, but we will only discuss the default one. What Lucene hopes to do is reduce the cost of calculating the score. In other words, reduce the number of computations the ranking function must do to rank the documents. In order to do so, Lucene compresses the document length into a one byte value. As such, there are only 256 distinct document lengths, and the algorithm can pre-compute the document length normalization portion of BM25, which is (k1*(1-b+b*dl/davg)), for each of the 256 lengths. Since each document falls into one of these 256 pre-computed lengths, there are fewer computations to do during the time of the query, since the algorithm can quickly look up this pre-computed value. However, in exchange for faster runtime, this may result in a slight loss of accuracy, since the document length is compressed into a single byte.

This paper has only touched on a fraction of the BM25 variants. There exist basic but powerful adjustments like BM11 and BM15, which set the value of b to 1 and 0 respectively. Others like ATIRE, BM25-adpt, and more are still used today, each with their own adjustments and uses. Different situations require different ranking algorithms for the best results, however, one thing is abundantly clear. The BM25 algorithm changed the landscape of text retrieval immensely, and it's variants will continue to remain some of the most powerful ranking functions for the foreseeable future.

References

Garcia. (2016, October). *A Tutorial on the BM25F Model*.

https://www.researchgate.net/publication/308991534_A_Tutorial_on_the_BM25F_Model

Kamphuis, Vries, Boytsov, & Lin. (2020, April 8). *Which BM25 Do You Mean? A Large-Scale Reproducibility Study of Scoring Variants*.

https://link.springer.com/chapter/10.1007/978-3-030-45442-5_4

Lv, & Zhai. (2011). *When Documents Are Very Long, BM25 Fails!*

http://sifaka.cs.uiuc.edu/~ylv2/pub/sigir11-bm25l.pdf

Connelly. (2018, April 19). *Practical BM25 - Part 2: The BM25 Algorithm and its Variables*.

https://www.elastic.co/blog/practical-bm25-part-2-the-bm25-algorithm-and-its-variables

Lv, & Zhai. (2011). *Lower-Bounding Term Frequency Normalization*.

https://d1wqtxts1xzle7.cloudfront.net/30739191/cikm11-lowerbound-with-cover-page-v2.pdf?Expires=1635362842&Signature=QckVkWebEFiCR1ev3V1O7L4T0GdA9SZq32WpUNtnbhOJpYKkO-VJW-gR2eKfxYKSCG-xlXiv1~lNFMgy67GTnPqJboVyzzCDdpJUvO~6HUadtlXTewHfvgBoKkczsbrCP3PFERYv9V0Kb93zmW0yhzUC1jhkoAs2GABIp4-cX2nVGwu1Cmg8MOWFxiYDd53OtdWg3c3fRJTwrdvKIDPlJdwpgHXZeMPn20ghXW0jimCS37dnFO6haicW3wQ3QDRNZmx8KSuY7MLJ3ev3hWyeCoUUh3~n~1xOBNcq4n-vsYoWMP2PMztH0b00t4fz8vKrVW4oCyBezfAXHful9HYfTg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA