Best Matching 25 (BM25) is a text-matching algorithm that improves on the Term Frequency-Inverse Document Frequency (TF-IDF) method. TF-IDF is a way to figure out how important a keyword is in a specific document compared to a bunch of other documents.

For instance, if we're looking for the keyword "machine learning" to find the best matching document, the TF-IDF algorithm would score how relevant that keyword is in each document using a specific formula:

$$TF(x, y) = number\ of\ occurrences\ of\ keyword\ x\ within\ document\ y$$

$$IDF(x) = log(\frac{total\ number\ of\ documents\ in\ the\ corpus}{number\ of\ documents\ containing\ keyword\ x})$$

$$TF\text{–}IDF(x, y) = TF(x, y) * IDF(x)$$

The document with the highest relevancy score will be suggested to the user.

BM25 improves on TF-IDF for two main reasons:

1. It takes the length of the document into account.

2. It adds a keyword saturation term.

Now, why are these upgrades important? The standard TF-IDF doesn't factor in document length, which means longer documents tend to score better just because the keyword appears more often. BM25 fixes this by normalizing the length in the Term Frequency (TF) calculation.

The saturation term in BM25 also tackles the issue of keyword spamming. In TF-IDF, each time a keyword shows up, the score just goes up linearly. But it makes more sense that going from 1 to 2 mentions should matter more than going from 100 to 101. The saturation term in BM25 helps with this by reducing the impact of a keyword's frequency as it increases.

With these key improvements in the BM25 equation, the algorithm offers better information retrieval results than the traditional TF-IDF method.

BM25 is a type of sparse embedding, meaning the resulting vector mostly consists of zeros. The size of the BM25 vector depends on the number of unique entities (like subwords, words, or tokens) in the document collection. The non-zero entries in the BM25 vector indicate where the query keyword appears, and their values show how relevant that keyword is.

If we're using a single-word keyword, and each entity represents a word, then only one part of the BM25 vector will have a non-zero value when the exact match for the keyword is found in the document. If there's no exact match, all values will be zero.

This shows a limitation of traditional sparse vectors. When the exact match for the keyword isn't found, the BM25 sparse vector doesn't capture the importance of the keyword, even if the document is related to the topic.

To overcome this issue, newer advancements in sparse embedding have led to the creation of SPLADE.

Comparision between BM25 and SPLADE:

BM25 and SPLADE are both sparse embedding vectors, each with their own pros and cons.

BM25 is a straightforward information retrieval algorithm that provides predictable results that are easy to explain. Its vector has fewer non-zero values compared to SPLADE, which makes the search process more efficient.

One of the strengths of BM25 is that it relies on exact string matching and doesn't require any specialized deep learning models, so no extra fine-tuning is necessary. However, it does need the query vocabulary to match what's in the document collection; otherwise, it might not return any results.

On the other hand, SPLADE produces a vector with more non-zero values, which can slow down retrieval. But this can also lead to better results since SPLADE can include semantically similar terms for each input word. Its BERT foundation means it can perform well even if the exact query terms aren't present in the corpus.

That said, SPLADE doesn't automatically outperform BM25 across different data sets. Like any deep learning model, it needs to be fine-tuned for the specific data to really shine compared to BM25.