# Understanding the BM25 Ranking Algorithm

Everton Gomede, PhD · Follow

5 min read · May 27, 2023

## Introduction

In the realm of information retrieval, search engines play a crucial role in assisting users in finding relevant information from vast amounts of data. The efficiency and effectiveness of search engines heavily rely on ranking algorithms. One such algorithm that has gained significant attention and success is the BM25 (**Best Match 25**) ranking algorithm. BM25 is a term-based ranking model that aims to provide accurate and relevant search results by scoring documents based on their term frequencies and document lengths. This essay explores the fundamental concepts and working principles of the BM25 ranking algorithm.

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

## Overview of the BM25 Algorithm

The BM25 algorithm was introduced by Robertson and Walker in 1994 as an improvement over the previous Okapi BM11 algorithm. BM25 takes into account both term frequency (TF) and document length normalization to determine the relevance of a document to a given query. It follows the probabilistic retrieval framework, which assumes that relevant and non-relevant documents follow different statistical distributions.

## Key Components of BM25

1. **Term Frequency (TF)**: TF refers to the number of times a particular term appears in a document. However, BM25 uses a modified term frequency that takes into account saturation effects to prevent overemphasizing heavily repeated terms.

2. **Inverse Document Frequency (IDF)**: IDF measures the importance of a term in the entire corpus. It assigns higher weights to terms that are rare in the corpus and lower weights to terms that are common. IDF is calculated using the formula: *IDF = log((N — n + 0.5) / (n + 0.5))*, where N is the total number of documents and n is the number of documents containing the term.

3. **Document Length Normalization**: BM25 incorporates document length normalization to address the impact of document length on relevance scoring. Longer documents tend to have more occurrences of a term,

leading to potential bias. Document length normalization counteracts this bias by dividing the term frequency by the document's length and applying a normalization factor.

4. **Query Term Saturation**: BM25 also includes a term saturation function to mitigate the impact of excessively high term frequencies. This function reduces the effect of extremely high term frequencies on relevance scoring, as very high frequencies often correspond to less informative terms.

Top highlight

### Calculation of BM25 Score

The BM25 score for a document D with respect to a query Q is calculated as the sum of the scores for individual query terms. The formula for calculating the BM25 score is as follows:

$$BM25(D, Q) = \sum(IDF(q) * ((TF(q, D) * (k1 + 1)) / (TF(q, D) + k1 * (1 - b + b * (|D| / avgdl)))))$$

In this formula, IDF(q) represents the inverse document frequency of the query term q, TF(q, D) denotes the modified term frequency of term q in document D, |D| represents the length of document D, and avgdl is the average document length in the corpus. Parameters k1 and b are tunable constants that control the impact of term frequency saturation and document length normalization, respectively.

### Advantages and Limitations of BM25

Advantages

1. BM25 is a widely used ranking algorithm due to its simplicity and effectiveness in producing relevant search results.
2. It takes into account both term frequency and document length normalization, which helps address the issue of document length bias.
3. The algorithm can handle large document collections efficiently, making it scalable for real-world search scenarios.

**Limitations**

1. BM25 does not consider the semantic meaning or context of the query and the documents, which may result in suboptimal ranking for certain types of queries.
2. It assumes statistical independence between query terms, which may not hold true in some cases where term dependencies exist.
3. The algorithm heavily relies on term frequency and document length, potentially overlooking other important factors like document structure and relevance feedback.

Here’s an example implementation of the BM25 ranking algorithm in Python:

```
import math
from collections import Counter

class BM25:
    def __init__(self, documents):
        self.documents = documents
        self.document_count = len(documents)
        self.avg_document_length = sum(len(doc) for doc in documents) / self.doc
        self.term_counts = self.calculate_term_counts()
        self.k1 = 1.2
        self.b = 0.75

    def calculate_term_counts(self):
        term_counts = Counter()
        for document in self.documents:
            term_counts.update(document)
        return term_counts

    def calculate_idf(self, term):
        document_with_term_count = self.term_counts[term]
        return math.log((self.document_count - document_with_term_count + 0.5) /

    def calculate_bm25_score(self, query, document):
        score = 0.0
        document_length = len(document)
        query_terms = Counter(query)

        for term in query_terms:
            if term not in self.documents:
                continue
            idf = self.calculate_idf(term)
            term_frequency = document.count(term)
            numerator = term_frequency * (self.k1 + 1)
            denominator = term_frequency + self.k1 * (1 - self.b + self.b * (doc
            score += idf * (numerator / denominator)

        return score

    def rank_documents(self, query):
        document_scores = []
        for document in self.documents:
            score = self.calculate_bm25_score(query, document)
            document_scores.append((document, score))

        ranked_documents = sorted(document_scores, key=lambda x: x[1], reverse=T
        return ranked_documents


# Example usage:
documents = [
    ['apple', 'banana', 'orange', 'apple'],
    ['banana', 'orange', 'orange'],
    ['apple', 'apple', 'banana', 'banana'],
    ['orange', 'orange', 'banana']
]

bm25 = BM25(documents)
query = ['apple', 'banana']
ranked_documents = bm25.rank_documents(query)

for document, score in ranked_documents:
    print(f"Document: {document}, Score: {score}")
```

In this example, we create a class `BM25` that takes a list of documents as input. The `calculate_term_counts` method calculates the frequency of each term in the corpus, and the `calculate_idf` method computes the inverse document frequency for a given term. The `calculate_bm25_score` method calculates the BM25 score for a query and a document. Finally, the `rank_documents` method ranks the documents based on their scores for a given query.

In the example usage, we create a `BM25` object with a list of example documents. We then specify a query and obtain the ranked documents and their corresponding scores. The documents are printed in descending order of their scores.

Note: This is a simplified implementation for demonstration purposes. In practice, you may need to preprocess the documents (e.g., tokenization, stemming) and handle additional considerations like stop words and query expansion for a more robust implementation.

## Conclusion

The BM25 ranking algorithm has played a significant role in improving the performance of search engines by considering term frequency, inverse document frequency, and document length normalization. While BM25 has become a standard in the field, it is not without its limitations. Future research and advancements may address these limitations by incorporating more sophisticated techniques, such as machine learning and natural language processing, to enhance the overall relevance and accuracy of search results. Nonetheless, BM25 remains a crucial milestone in the evolution of information retrieval and continues to be a foundation for further advancements in search engine technology.

Artificial Intelligence    Machine Learning

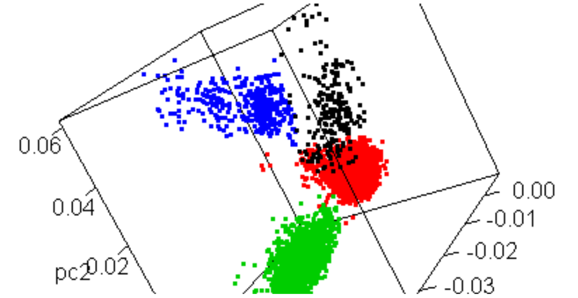
Written by Everton Gomede, PhD

6.6K Followers

Follow

Computer Scientist developing algorithms, solutions, and tools that enable companies and their analysts to extract insights from data to decision-makers.

## More from Everton Gomede, PhD



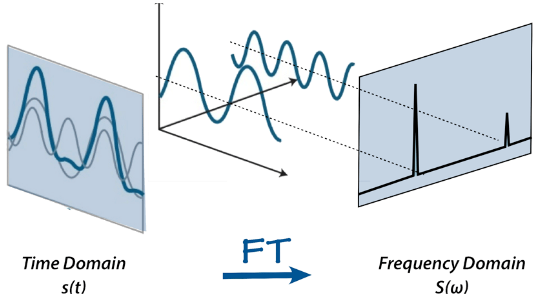
Everton Gomede, PhD

### Step by Step of Principal Component Analysis

Introduction

7 min read · Nov 7, 2023

102



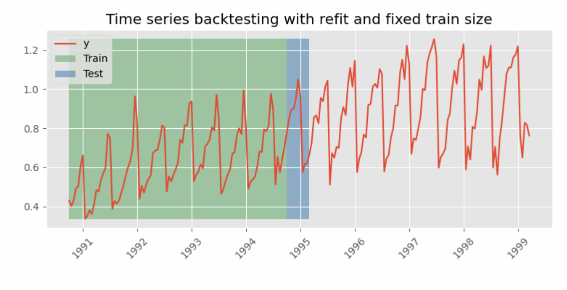
Everton Gomede, PhD in The Modern Scientist

### The Fourier Transform and its Application in Machine Learning

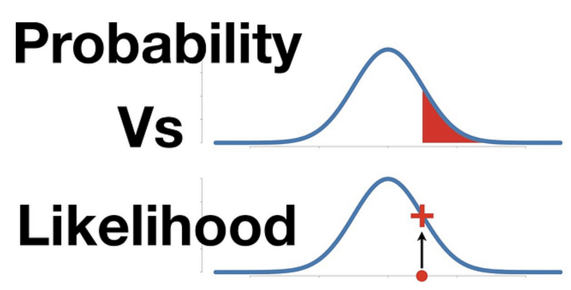
Introduction

5 min read · Nov 3, 2023

1.4K 10



Everton Gomede, PhD



Everton Gomede, PhD in *AI monks.io*

**Forecasting Non-Stationary Time Series**

Introduction

6 min read · Oct 14, 2023

304 2

**Understanding the Distinct Concepts of Likelihood and...**

Introduction

6 min read · Jan 2

73 2

See all from Everton Gomede, PhD

## Recommended from Medium

IVAN ILIN in Towards AI

**Advanced RAG Techniques: an Illustrated Overview**

A comprehensive study of the advanced retrieval augmented generation techniques...

19 min read · Dec 17, 2023

Yassine EL KHAL

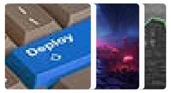
**The complete guide to string similarity algorithms**

Introduction

14 min read · Aug 21, 2023

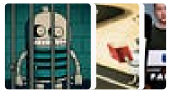
3.8K 23 281 4

## Lists



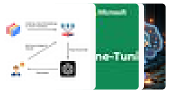
**Predictive Modeling w/ Python**

20 stories · 824 saves



**AI Regulation**

6 stories · 286 saves



**Natural Language Processing**

1118 stories · 589 saves

**Practical Guides to Machine Learning**

10 stories · 953 saves

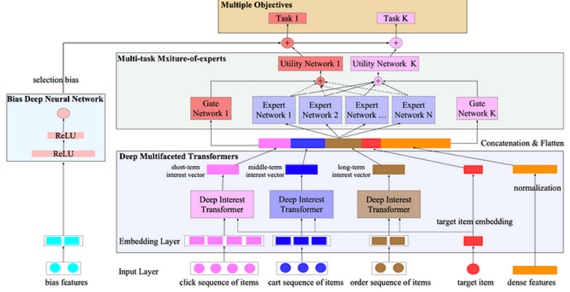
Felipe Bandeira in Python in Plain English

**Transformers vs. OCR: who can actually read better?**

A comparison between OCR-based and OCR-free approaches for Information Extraction

15 min read · Sep 1, 2023

394 4



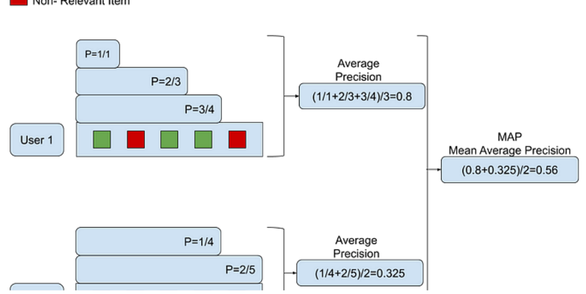
Subir Verma

**Multi-objective Ranking in Large-Scale E-commerce Recommende...**

Motivation

7 min read · Jan 10

78



Abul Aala Fareh

**Different Reranking Techniques in LlamaIndex**

What is Reranking

6 min read · Dec 12, 2023

4

Manish Chablani

**Recommendation system / Search / IR evaluation metrics**

Metrics in Recommendation Systems / Retrival / Ranking:

3 min read · Dec 28, 2023

10

See more recommendations