Term Frequency-Inverse Document Frequency (TFIDF)

The term frequency-inverse document frequency (TF-IDF) is a metric used in information retrieval (IR) and machine learning to quantify the importance or relevance of string representations (words, phrases, lemmas, etc.) in a document among a collection of documents (also known as a corpus).

What exactly is TF (term frequency)?

Term frequency works by examining the frequency of a specific term in relation to the document.There are several measures or techniques to define frequency:

- The number of occurrences of a word in a document (raw count).
- Term frequency has been modified to account for the document's length (raw count of occurrences divided by number of words in the document).
- Frequency scaled logarithmically (e.g. log(1 + raw count)).
- The frequency of a Boolean expression (e.g. 1 if the term occurs, or 0 if the term does not occur, in thedocument).

Objective TF-IDF:

1. The objective of the TF-IDF (Term Frequency-Inverse Document Frequency) algorithm is to determine the importance of a term in a document relative to its occurrence in a collection of documents.

2. TF-IDF is commonly used in information retrieval and text mining applications to rankthe relevance of documents based on a user's query.

Working of TF-IDF:

The algorithm calculates a weight for each term in a document based on two factors: term frequency (TF) and inverse document frequency (IDF). Term frequency is a measure of how frequently a term appears in a document, while inverse document frequency is a measure of how rare or unique a termis in the entire collection of documents. The TF-IDF score for a term in a document is the product of its TF and IDF values.

Goal of TF-IDF:

The goal of TF-IDF is to give higher weight to terms that appear frequently in a document but

rarelyin other documents. This helps to identify important keywords and phrases in a document that are unique to that document, making it easier to distinguish it from other documents in a collection. By using TF-IDF, search engines can rank search results based on the relevance of each document to a user's query.

What exactly is the IDF (inverse document frequency)?

Inverse document frequency examines how prevalent (or rare) a term is in the corpus. IDF is calculated as follows, where t is the term (word) to be measured and N is the number of documents (d)in the corpus (D).. The denominator is simply the number of papers that include the phrase, t.

Functionality TFIDF

The functionality of the TF-IDF algorithm is to provide a way to measure the importance of a term ina document or a collection of documents. It does this by calculating a weight for each term that takes into account both the frequency of the term in the document (TF) and the rarity of the term in the collection (IDF).

The TF-IDF algorithm is useful in a variety of natural language processing tasks, such as informationretrieval, text classification, and topic modeling. It can be used to:

1. Rank documents based on their relevance to a user's query: When a user enters a search query, thesearch engine can use the TF-IDF weights to identify documents that contain the most relevant terms.
2. Identify important terms in a document: The TF-IDF weights can be used to identify the mostimportant terms in a document, which can be useful for summarization or topic modeling.
3. Cluster similar documents: Documents that contain similar terms with high TF-IDF weights are likelyto be related to the same topic, and can be clustered together.

Overall, the TF-IDF algorithm is a powerful tool for analyzing text data and identifying importantterms and documents.

Algorithm TF-IDF:

The steps to perform TF-IDF calculation on a text document corpus are as follows:

1. Tokenization: Tokenization involves splitting the text into individual words or terms. Each term is then considered a token. This process can be done using natural language processing (NLP) techniques.
2. Removal of stop words: Stop words are common words that don't carry much meaning such as "the", "a", "an", "in", "on", etc. These words can be removed from the tokens as they don't add much valueto the analysis.

Calculate term frequency (TF): Count the number of times each token appears in a document. This gives you the frequency of each token in the document.

1. Calculate inverse document frequency (IDF): IDF measures the importance of a token in a corpus of documents. It is calculated as the logarithm of the total number of documents in the corpus divided bythe number of documents that contain the token. The formula for IDF is IDF = log(N/df), where N is the total number of documents in the corpus and df is the number of documents that contain the token.
2. Calculate TF-IDF: The final step is to calculate the TF-IDF score for each token in each document. This is done by multiplying the TF score with the IDF score. The formula for TF-IDF is TF-IDF = TF
   * IDF.
3. Repeat for all documents: Repeat steps 3-5 for all documents in the corpus.

The output of this process is a matrix of TF-IDF scores for each token in each document. This matrix can then be used for various text mining and information retrieval tasks such as document clustering, classification, and similarity analysis.

Illustration with Example:

TF-IDF (Term Frequency-Inverse Document Frequency) is a method used in natural language processing to determine the importance of a term within a document or corpus (a collection of documents).

Here's a simple example to explain how TF-IDF works:
Suppose we have two documents, Document A and Document B, which contain the following

text:

Document A: "The quick brown fox jumps over the lazy dog."

Document B: "The brown dog chased the fox and then ran away."

To calculate the **TF-IDF scores** for each word in these documents, we first need to compute the termfrequency (TF) and inverse document frequency (IDF) for each word:

Term Frequency (TF): The number of times a word appears in a document divided by the total number of words in the document.

For example, in Document A, the word **"the" appears twice**, so the TF for "the" is **2/9** (since thereare 9 total words in the document).

Inverse Document Frequency (IDF): A measure of how important a word is to the entire corpus. Itis calculated as the logarithm of the total number of documents divided by the number of documents that contain the word.

For example, in our corpus of two documents, the word "fox" appears in only one document, so the
IDF for "fox" is log(2/1) = log(2) = **0.30**.

Once we have computed the TF and IDF for each word, we can **multiply them together** to get theTF-IDF score for each word.

For example, the TF-IDF score for the word "the" in Document A is:
TF-IDF("the", Document A) = TF("the", Document A) * IDF("the") = (2/9) * log(2/2) = **0**

This means that **"the" is a very common word** in both documents and therefore has a low importance score n the other hand, the **TF-IDF score for the word "fox"** in Document A is:

TF-IDF("fox", Document A) = TF("fox", Document A) * IDF("fox") = (1/9) * log(2/1) = **0.10**
This means that **"fox" is a relatively rare** word in the corpus and therefore has a higher importancescore.
In general, words with high TF-IDF scores are considered to be more important and informative,while words with low TF-IDF scores are considered to be less important and more common.

TFIDF:

Advantages:

- Simple and effective method for representing text data
- Reduces impact of common terms
- Useful for a variety of natural language processing tasks
- Language-independent
- Computationally efficient for large collections of documents.

Disadvantages:

- Ignores context: The algorithm does not take into account the context in which a term appears, which can lead to inaccurate representations of the text.
- Requires extensive preprocessing: Before applying the algorithm, the text data must be preprocessed by removing stop words, stemming or lemmatizing the remaining words, and potentially applying other text normalization techniques. This can be time-consuming and requires careful attention to detail.
- Sensitive to document length: The algorithm may produce different results for documents of different lengths, which can make it difficult to compare or cluster documents.
- Assumes independence between terms: The algorithm assumes that the terms in a document are independent of each other, which may not always be true.
- Not effective for some types of text data: The TF-IDF algorithm may not be effective for certain typesof text data, such as short documents or highly specialized domain-specific language.

OKAPI

Okapi and BM25 have been shown to be highly effective in ranking documents for a wide range of queries, and the model is widely used in commercial search engines such as Elasticsearch and Solr. However, like any IR model, Okapi has its limitations and is not a perfect solution for all types of queries and collections.

The formula used by the Okapi model is called BM25 (Best Match 25) and it is defined as follows:

$BM25(q,d) = \sum_{i=1 \text{ to } n} IDF(qi) * ((k+1)f(qi,d))/(f(qi,d) + k(1-b+b*(DL/AVDL)))$

where:

- q: the query
- d: the document being ranked
- n: the total number of query terms
- IDF(qi): the inverse document frequency for term
- qi f(qi,d): the frequency of term qi in document d
- k: a constant that balances the contribution of term frequency and document length to the score (usually set to k=1.2)
- b: a constant that balances the effect of document length on the score (usually set to b=0.75)
- DL: the length of the document in words
- AVDL: the average length of documents in the collection being searched

The formula calculates a score for each document d based on its similarity to the query q. The score is higher for documents that contain more of the query terms, but it also takes into account factors such as document length and term frequency to ensure that longer documents and frequent terms are not given undue weight.

OKAPI BM25:

The BM25 formula combines these factors into a single score for each document, which is then used to rank the documents in order of relevance to the user's query.

To find Okapi BM25 score for a query and a document using the formula, you will need to calculate the following values:

1. ni: The number of documents containing the term i (ni = len(documents containing i))
2. fi: The frequency of term i in the document being scored
3. qfi: The frequency of term i in the query
4. N: The total number of documents in the corpus
5. K: A normalization factor that depends on the length of the document being scored and the average document length
6. k1, b, k2: Tuning parameters that control the importance of term frequency and document length in the scoring algorithm

With these values, you can calculate the score for a single term as:

((k1 + 1) * fi / (K + fi)) * ((k2 + 1) * qfi / (k2 + qfi)) * log((N - ni + 0.5) / (ni + 0.5))

To get the Okapi BM25 score for a document and a query, you need to sum the scores for all terms in the query that appear in the document.

Objectives of OKAPI:

The Okapi BM25 (Best Matching 25) model is a ranking function used for information retrieval tasks such as document retrieval and search engines. The main objectives of using the Okapi model are:

1. Improving search accuracy: The Okapi model is designed to improve search accuracy by ranking documents based on their relevance to a user's query. It uses statistical modeling to estimate the relevance of a document to a query based on the frequency of query terms in the document.

2. Handling long queries: The Okapi model is effective in handling long queries by considering the relevance of each query term to the document. It also takes into account the length of the document, which helps in identifying longer documents that contain more relevant information.

3. Providing flexibility: The Okapi model provides flexibility in weighting query terms based on their importance. It can also be customized to use different weighting schemes for different types of queries or documents.

4. Handling noisy data: The Okapi model is robust to noisy data, which is common in real-world search scenarios. It can handle misspelled words, synonyms, and other variations in

the query terms and document content.

5. Scalability: The Okapi model is scalable and can be used in large-scale search applications with millions of documents and queries.

6. Overall, the Okapi model helps in improving the relevance and effectiveness of search results and is widely used in modern search engines and information retrieval systems.

Why OKAPI?

1. Provides a ranking function for information retrieval systems.
2. Determines the relevance of documents to a specific query.
3. Takes into account various factors such as term frequency, document length, and inverse document frequency.
4. Uses the Okapi BM25 algorithm to score and rank documents in a collection.
5. Ranks documents based on their score, with the highest-scoring documents being considered the most relevant to the query.
6. Improves the accuracy and relevance of search results.
7. Enhances the user experience and increases productivity in various applications, such as web search engines, digital libraries, and document classification systems.
8. BM25 (Best Matching 25) is a ranking function used in information retrieval to score and rank documents based on their relevance to a given query. It takes into account the term frequency and document length to calculate a relevance score.

Advantages and Disadvantages:

Advantages	Disadvantages
1. Effectiveness: The Okapi model has been shown to be highly effective in ranking documents for a wide range of queries and collections, outperforming other models in many cases.	1. Need for query expansion: The Okapi model relies on the presence of relevant terms in the query to generate effective rankings.
2. Robustness: The Okapi model is robust to variations in document length and term frequency, making it well-suited for dealing with the inherent variability of natural language text.	2. Sensitivity to parameter settings: While the Okapi model is tunable, this also means that it can be sensitive to the settings of its parameters. If the parameters are not set correctly, the performance of the model may suffer.
3. Flexibility: The Okapi model is flexible enough to handle different types of queries, including long queries, short queries, and queries with multiple terms.	3. Lack of semantic understanding: The Okapi model is based on a statistical approach to information retrieval and does not have a deep understanding of the semantics of the query or the documents being searched
4. Tunability: The Okapi model is tunable, which means that it can be adapted to specific collections or query sets by adjusting the values of its parameters.	4. Limited handling of structured data: The Okapi model is primarily designed for text-based information retrieval and may not be as effective in handling structured data such as databases or spreadsheets.
5. Interpretability: The Okapi model is based on a clear mathematical formula that is easy to understand and interpret, making it easier for researchers and practitioners to analyze its behavior and performance.	5. Computational complexity: The Okapi model can be computationally complex, especially when dealing with large collections or queries with many terms.

PROBABILISTIC MODEL

The probabilistic retrieval model is a widely used approach in information retrieval, which is based on the idea of representing documents and queries as probability distributions over terms (i.e., words). This model assumes that a user's information need can be captured by a query, which is a set of terms that reflect the user's information need. The goal of information retrieval is to find the most relevant documents that satisfy the user's information need.

Objective Likelihood Ratio Approach Probabilistic Retrieval Model:

1. The objective of the likelihood ratio approach in the probabilistic retrieval model is to estimate the probability that a document is relevant to a user's query, based on the observed evidence (i.e., the words in the query and document). The likelihood ratio approach assumes that the probability of observing a set of words in a relevant document is different from the probability of observing the same set of words in an irrelevant document.

2. In the likelihood ratio approach, a document is represented as a bag of words, and a query is represented as a set of terms. The goal is to calculate the likelihood ratio of a document being relevant to a query versus being irrelevant to the query. This likelihood ratio is computed by dividing the probability of observing the words in the document given that the document is relevant, by the probability of observing the words in the document given that the document is irrelevant.

Steps (Algorithm):

1. Preprocessing: Tokenize the query and document into words or terms, and remove stop words and punctuation.

2. Building a language model for the query: Estimate the probability of each term in the query using a language model, such as the maximum likelihood estimate or the Dirichlet prior smoothing.

3. Building a language model for the document: Estimate the probability of each term in the document using a language model, such as the maximum likelihood estimate or the Dirichlet prior smoothing.

4. Computing the query likelihood ratio: Compute the likelihood ratio of the document being relevant to the query as follows:

5. QLR(d,q) = P(d|q) / P(d|~q)

6. where P(d|q) is the probability of observing the words in the document d given the query q, and P(d|~q) is the probability of observing the words in the document d given a background language model, which represents the distribution of words in the collection of all documents.

7. P(d|q) can be computed as the product of the probability of each word in the document given the query, and P(d|~q) can be computed as the product of the probability of each word in the document given the collection language model.

8. Ranking the documents: Rank the documents based on their query likelihood ratio scores, and return the top-k documents as the result of the query.

Advantages:

1. They are effective in modeling uncertainty and dealing with incomplete or noisy data.
2. They can handle a wide variety of query types, including ad hoc queries and complex queries with Boolean operators and phrase matching.
3. They are based on solid mathematical foundations and have well-defined probabilistic interpretations.
4. They are flexible and can incorporate various types of information, such as term frequencies, document lengths, and collection statistics.

Disadvantages:

1. They can be computationally expensive to implement and require significant resources for indexing and retrieval, especially for large-scale document collections.
2. They can be sensitive to the choice of smoothing parameters and other tuning parameters, and the optimal values may vary depending on the specific application.
3. They may not perform well in certain scenarios, such as when the query is very long or when the relevance of documents is highly context-dependent.
4. They may struggle to handle noisy or irrelevant data, such as documents with spam or irrelevant content, or queries with misspelled or ambiguous terms.