TF-IDF (Term Frequency Time Inverse Document Frequency) Concept

Posted on August 24, 2019

Introduction

TF-IDF is an information-retrieval technique used to represent the importance of words in documents. The abbreviation stands for *term frequency–inverse document frequency*. The current post title contains the word “Time,” but the established technical term is “term frequency.” TF-IDF is useful when a system must compare text, rank documents for a query, identify characteristic vocabulary, classify documents, cluster similar items, or build a content-based recommender. It does not understand meaning in the human sense; it converts text into weighted numerical features that can be compared efficiently.

The original exercise combines two different recommendation ideas. The first is a demographic or popularity-based movie ranking that uses vote averages and vote counts. The second is a content-based system that uses plot descriptions and TF-IDF. These methods can complement one another, but they solve different problems. A weighted rating identifies broadly well-regarded movies, while TF-IDF identifies movies whose textual descriptions are similar.

Term Frequency

Term frequency measures how often a term appears in a document. The simplest definition is the raw count of a word. If the term “volcano” appears five times in one article, its raw term frequency is five. Raw counts can overemphasize long documents because longer texts naturally contain more words. Implementations therefore may divide the count by document length, use binary presence or absence, or apply sublinear scaling such as $1 + \log(tf)$. The Stanford *Introduction to Information Retrieval* explains that repeated occurrence is evidence of relevance, but twenty occurrences do not necessarily make a term twenty times more informative than one occurrence. (Manning et al., 2008)

Inverse Document Frequency

Term frequency alone treats every word as equally useful. Common words such as “the,” “is,” and “movie” may appear frequently but do little to distinguish one document from another. Inverse document frequency reduces the weight of terms that occur in many documents and increases the relative importance of terms that occur in fewer documents.

A common form is:

idf(t) = log(N / df(t))

Here, *N* is the total number of documents and *df(t)* is the number of documents that contain term *t*. A rare term receives a larger inverse-document-frequency value. A term that appears in nearly every document receives a small value because it has little discriminating power. Practical libraries often add smoothing so that division by zero is impossible and previously unseen terms can be handled consistently.

Combining the Two Measures

The TF-IDF weight of a term in a document is the product of its term-frequency component and inverse-document-frequency component:

tf-idf(t,d) = tf(t,d) × idf(t)

A term receives a high weight when it appears repeatedly in one document but is uncommon across the document collection. It receives a low weight when it is absent, occurs only weakly in the document, or is common throughout the collection. Once every document is represented by a vector of these weights, mathematical similarity measures can compare the documents.

Text Preparation

Before computing TF-IDF, the system defines what counts as a feature. Typical preprocessing includes lowercasing, tokenization, removal or control of punctuation, and decisions about stop words. Word-level features may be individual terms or n-grams such as two-word phrases. Character n-grams can be useful for spelling variation, short texts, and languages where word boundaries are difficult to identify. Stemming or lemmatization may combine related word forms, although aggressive normalization can remove distinctions that matter.

Feature choices depend on the task. In movie plots, the phrase “serial killer” may be more informative than either word alone, so bigrams can improve representation. Extremely rare features may be removed with a minimum document-frequency threshold, while extremely common features may be removed with a maximum document-frequency threshold. Scikit-learn’s `TfidfVectorizer` combines vocabulary construction, counting, inverse-document-frequency weighting, and optional normalization. (Scikit-learn developers, 2026)

From Documents to Vectors

Suppose a corpus contains thousands of movie descriptions. Each distinct feature becomes a dimension. A movie document is then represented by a sparse vector because it contains only a small fraction of all possible terms. Sparse matrices make the method practical even when the vocabulary contains tens of thousands of features.

Rows are commonly normalized to unit length. With L2 normalization, cosine similarity between two TF-IDF vectors can be computed through their dot product. Cosine similarity focuses on the direction of the vectors rather than raw document length. A score near one indicates strongly aligned term patterns, while a score near zero indicates little overlap. Negative scores do not normally occur because standard TF-IDF vectors contain nonnegative values.

Loading and Inspecting the Movie Dataset

The original case begins by loading movie metadata and asking how films should be ranked. The dataset may include titles, overviews, average ratings, vote counts, genres, release dates, and other fields.

Data cleaning should occur before ranking or text vectorization. Missing overviews may be replaced with empty strings for the text pipeline, but missing ratings and vote counts require different treatment. Duplicate records, inconsistent titles, language differences, and adult-content or age-rating constraints may also affect recommendations. Training and evaluation data should be separated when the system is tested so that the developer does not measure performance on the same information used to tune the model.

Popularity Ranking Is Not TF-IDF

A movie with a perfect average rating from two voters should not automatically outrank a movie with a slightly lower average from thousands of voters. The original exercise addresses this problem with a weighted rating similar to the formula historically used in IMDb-style rankings:

WR = (v / (v + m))R + (m / (v + m))C

- *v* is the number of votes for the movie;
- *m* is the minimum vote count required for consideration;
- *R* is the movie's average rating; and

- C is the mean rating across the collection.

This formula shrinks the score of films with few votes toward the overall mean. It is a Bayesian-style reliability adjustment, not a TF-IDF calculation. The percentile used for m is a policy choice. Selecting the 95th percentile produces a short list of widely rated films, but it can disadvantage recent, independent, foreign-language, or niche movies that have not accumulated large audiences.

The minimum number of votes can be calculated with a quantile function:

The qualified dataset is then filtered so that only films meeting the selected threshold are included:

Calculating the Weighted Ranking

A copied DataFrame prevents later operations from unintentionally changing the source metadata. A function can compute the weighted rating for every qualified movie, and the resulting score can be sorted in descending order.

The final table may display the title, vote average, vote count, and weighted score:

This popularity list is useful as a default recommendation for new users because it does not require personal history. However, it gives nearly the same recommendations to everyone and can reinforce existing popularity.

Building the Content-Based Recommender

The content-based stage uses a movie's overview or plot description. Missing descriptions are filled, and a TF-IDF vectorizer is fitted to the complete collection. The result is a document-term matrix in which every movie has a weighted representation.

To recommend films similar to a selected title, the system identifies that movie's row, computes

cosine similarity against all other rows, sorts the scores, excludes the same movie, and returns the highest-scoring titles. A film about extraterrestrial invasion will tend to retrieve descriptions containing distinctive terms such as “alien,” “spacecraft,” “planet,” or related phrases. A zombie film may retrieve plots containing “undead,” “infection,” “survivors,” and “apocalypse.”

The content-based score may be combined with the weighted rating. One approach first retrieves the most textually similar candidates and then reranks them using a mixture of similarity, rating reliability, recency, language, or user constraints. This prevents a highly similar but very poorly received film from always appearing first. The weights should be tested rather than selected only by intuition.

Limitations of TF-IDF

TF-IDF uses lexical overlap and a bag-of-words representation. It usually ignores word order beyond the chosen n-grams, has difficulty with synonyms, and cannot naturally recognize that “automobile” and “car” are related unless the corpus or preprocessing connects them. It can also mistake shared generic vocabulary for thematic similarity. Short plot descriptions may not provide enough evidence, while promotional descriptions may emphasize marketable phrases rather than the actual content.

The method is corpus-dependent. A term can be rare in one collection and common in another. Updating the corpus changes document frequencies and therefore changes weights. The system also inherits bias from the metadata: poorly described films, minority-language films, and older titles with limited summaries may receive weaker representations.

Modern embedding models can represent semantic relationships more effectively, but TF-IDF remains valuable because it is fast, transparent, inexpensive, and easy to debug. Its feature weights can be inspected directly, and it performs strongly in many classification and retrieval problems, especially when the dataset is moderate and vocabulary is informative.

Evaluation and Responsible Use

A recommender should be evaluated with more than a few attractive examples. Offline measures may include precision, recall, ranking quality, coverage, novelty, and diversity. User testing can determine whether recommendations are understandable and useful. A system should also avoid exposing sensitive viewing history, provide controls for removing or resetting profiles, and explain when recommendations are based on popularity rather than personal preference.

Conclusion

TF-IDF assigns high weight to terms that are frequent in one document and uncommon across the corpus. It converts text into sparse vectors that can be compared with cosine similarity, making it suitable for a content-based movie recommender. The movie weighted-rating formula in the original exercise addresses a separate reliability problem involving vote averages and vote counts. A stronger recommendation pipeline uses each method for its proper purpose: weighted ratings for broad quality or popularity, and TF-IDF for textual similarity. Clear preprocessing, evaluation, and awareness of limitations are necessary for useful results.

References

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. https://nlp.stanford.edu/IR-book/

Scikit-learn developers. (2026). *TfidfVectorizer documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html