DAY – 8

7 August 2025

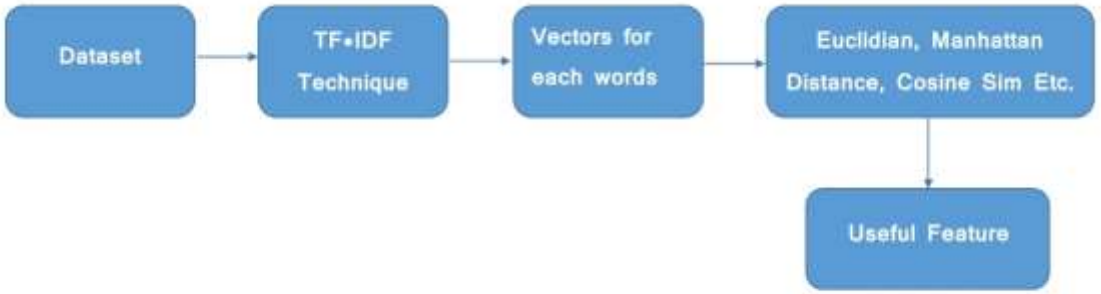
TF-IDF (Term Frequency-Inverse Document Frequency) is a core machine learning technique in Natural Language Processing (NLP) that numerically scores word importance in a document relative to a collection (corpus), highlighting unique keywords by multiplying how often a word appears (TF) by how rare it is across all documents (IDF) to convert text into meaningful vectors for tasks like search, classification, and recommendations. It helps models understand keywords, overcoming simple word counts by down-weighting common words (like 'the') and boosting specific terms.

How it works (The two parts):

- Term Frequency (TF): Measures how often a word appears in a single document. More appearances suggest higher relevance within that document.
- Formula: (Count of word in doc) / (Total words in doc).
- Inverse Document Frequency (IDF): Measures how rare a word is across the entire collection of documents? Words in fewer documents get a higher score.
- Formula: log (Total documents / Documents containing the word).

Why it's used in Machine Learning:

- Text Vectorization: Converts unstructured text into numerical vectors that ML models can process.
- Feature Extraction: Identifies important keywords (features) for tasks like topic modelling.
- Information Retrieval: Ranks search results based on keyword relevance.
- Recommendation Systems: Suggests content based on user's preferred keywords.



What is TF-IDF?

TF-IDF is a numerical statistic that reflects the significance of a word within a document relative to a collection of documents, known as a corpus. The idea behind TF-IDF is to quantify the importance of a term in a document with respect to its frequency in the document and its rarity across multiple documents.

Term Frequency (TF)

TF measures the frequency of a term within a document. It is calculated as the ratio of the number of times a term occurs in a document to the total number of terms in that document. The goal is to emphasize words that are frequent within a document.

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

Inverse Document Frequency (IDF)

IDF measures the rarity of a term across a collection of documents. It is calculated as the logarithm of the ratio of the total number of documents to the number of documents containing the term. The goal is to penalize words that are common across all documents.

$$IDF(t,D) = \log\left(\frac{\text{Total number of documents in the corpus } N}{\text{Number of documents containing term } t}\right)$$

TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical measure that evaluates a word's importance in a document relative to a collection (corpus), transforming text into numerical vectors for machine learning, used heavily in search engines for relevance ranking and keyword extraction. It combines Term Frequency (TF), showing how often a word appears in a document, and Inverse Document Frequency (IDF), which down-weights common words and highlights rare, meaningful terms. A simple Python program uses sklearn.feature_extraction.text.TfidfVectorizer to easily calculate these scores, making text data usable for algorithms.

Introduction to TF-IDF

What it is: A core concept in Natural Language Processing (NLP) and Information Retrieval (IR) that scores words based on their significance within a document and across a corpus.

Why it's used: Converts unstructured text into structured numerical data (vectors) that ML models can process, helping identify important keywords and rank search results.

Core Components:

Term Frequency (TF): (Number of times a term appears in a document) / (Total terms in the document).

Inverse Document Frequency (IDF): log(Total number of documents / Number of documents containing the term).

TF-IDF Score: TF * IDF.

Python Program Example (using scikit-learn)

This example shows how to implement TF-IDF in Python.

```
from sklearn.feature_extraction.text import TfidfVectorizer

# Sample corpus (collection of documents)
corpus = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?"
]

# 1. Initialize the TfidfVectorizer
vectorizer = TfidfVectorizer()

# 2. Fit the vectorizer to the corpus and transform it into TF-IDF matrix
tfidf_matrix = vectorizer.fit_transform(corpus)

# 3. Get feature names (the words)
feature_names = vectorizer.get_feature_names_out()

# 4. Print the results (a sparse matrix of scores)
print("TF-IDF Matrix:")
print(tfidf_matrix)
```

```
print("\nFeature Names (Words):", feature_names)


# To see scores for a specific document (e.g., the first one)
print("\nScores for Document 1:")
# Convert sparse matrix row to array for easy viewing
doc1_scores = tfidf_matrix[0].toarray()[0]
for word, score in zip(feature_names, doc1_scores):
    if score > 0:
        print (f" {word}: {score:.4f}")
```