Understanding TF-IDF (Term Frequency-Inverse Document Frequency)

Last Updated : 13 Aug, 2025

TF-IDF (Term Frequency–Inverse Document Frequency) is a statistical method used in natural language processing and information retrieval to evaluate how important a word is to a document in relation to a larger collection of documents. TF-IDF combines two components:

1. Term Frequency (TF): Measures how often a word appears in a document. A higher frequency suggests greater importance. If a term appears frequently in a document, it is likely relevant to the document's content.

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

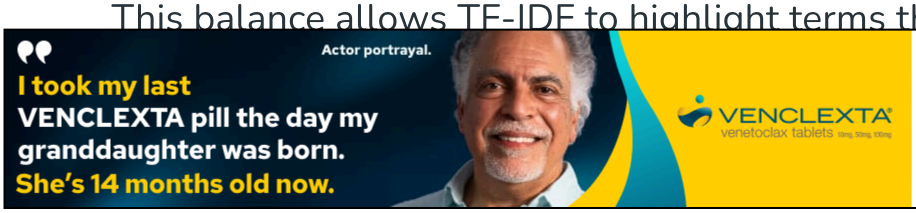
Term Frequency (TF)

2. Inverse Document Frequency (IDF): Reduces the weight of common words across multiple documents while increasing the weight of rare words. If a term appears in fewer documents, it is more likely to be meaningful and specific.

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

Inverse Document Frequency (IDF)

This balance allows TF-IDF to highlight terms that are both frequent



making it a useful tool for tasks like search ranking, text classification and keyword extraction.

Converting Text into vectors with TF-IDF

Let's take an example where we have a corpus (a collection of documents) with three documents and our goal is to calculate the TF-IDF score for specific terms in these documents.

1. **Document 1:** "The cat sat on the mat."
2. **Document 2:** "The dog played in the park."
3. **Document 3:** "Cats and dogs are great pets."

Our goal is to calculate the TF-IDF score for specific terms in these documents. Let’s focus on the word **"cat"** and see how TF-IDF evaluates its importance.

Step 1: Calculate Term Frequency (TF)

For Document 1:

- The word **"cat"** appears 1 time.
- The total number of terms in Document 1 is 6 ("the", "cat", "sat", "on", "the", "mat").
- So, TF(cat,Document 1) = 1/6

For Document 2:

- The word **"cat"** does not appear.
- So, TF(cat,Document 2)=0.

For Document 3:

- The word **"**cat" appears 1 time.
- The total number of terms in Document 3 is **6** ("cats", "and", "dogs",

In Document 1 and Document 3 the word **"cat"** has the same TF score. This means it appears with the same relative frequency in both documents. In Document 2 the TF score is 0 because the word **"cat"** does not appear.

Step 2: Calculate Inverse Document Frequency (IDF)

- **Total number of documents in the corpus (D):** 3
- **Number of documents containing the term "cat":** 2 (Document 1 and Document 3).

$$IDF(cat, D) = log\frac{3}{2} \approx 0.176$$

Step 3: Calculate TF-IDF

The TF-IDF score for "cat" is 0.029 in Document 1 and Document 3 and 0 in Document 2 that reflects both the frequency of the term in the document (TF) and its rarity across the corpus (IDF).

The TF-IDF score is the product of TF and IDF:

$$\text{TF - IDF(t, d, D) = TF(t, d)} \times \text{IDF(t, D)}$$

TF-IDF

- For Document 1: TF-IDF (cat, Document 1, D)-0.167 * 0.176 - 0.029
- For Document 2: TF-IDF(cat, Document 2, D)-0x 0.176-0
- For Document 3: TF-IDF (cat, Document 3, D)-0.167 x 0.176 ~ 0.029

Implementing TF-IDF in Python

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

Step 2: Collect strings from documents and create a corpus

```
d0 = 'Geeks for geeks'
d1 = 'Geeks'
d2 = 'r2j'
string = [d0, d1, d2]
```

Step 3: Get TF-IDF values

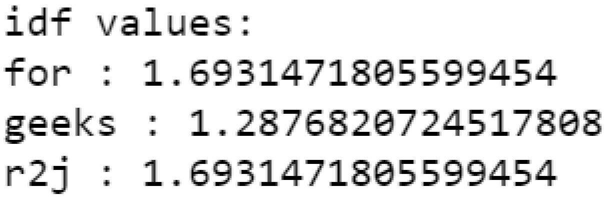
Here we are using TfidfVectorizer() from scikit learn to perform tf-idf and apply on our courpus using fit_transform.

```
tfidf = TfidfVectorizer()
result = tfidf.fit_transform(string)
```

Step 4: Display IDF values

```
print('\nidf values:')
for ele1, ele2 in zip(tfidf.get_feature_names_out(), tfidf.idf_):
    print(ele1, ':', ele2)
```

Output:

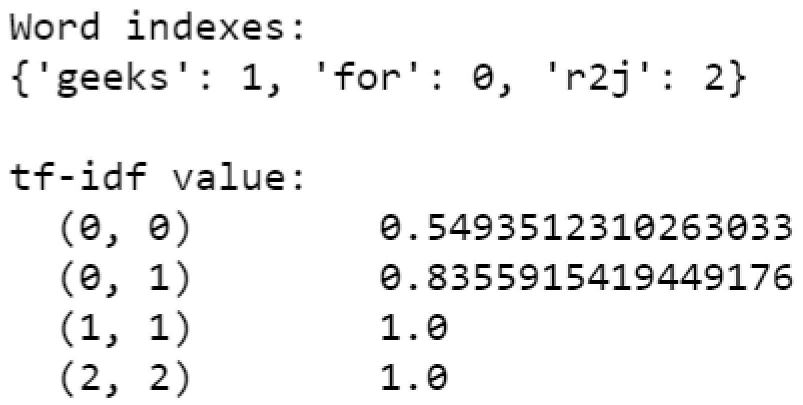


```
idf values:
for : 1.6931471805599454
geeks : 1.2876820724517808
r2j : 1.6931471805599454
```

Step 5: Display TF-IDF values along with indexing

```
print('\nWord indexes:')
print('\ntf-idf values in matrix form:')
print(result.toarray())
```

Output:

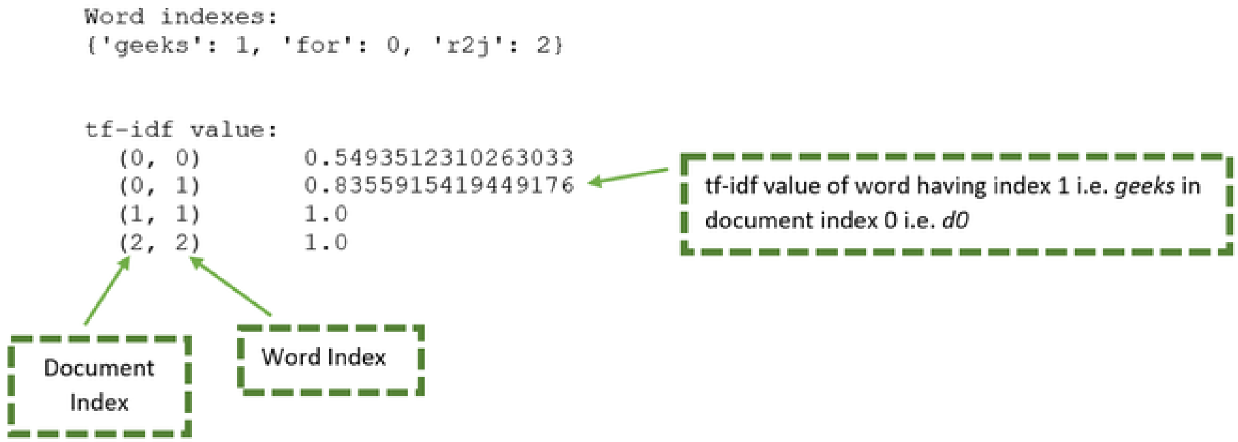


```
Word indexes:
{'geeks': 1, 'for': 0, 'r2j': 2}

tf-idf value:
  (0, 0)        0.5493512310263033
  (0, 1)        0.8355915419449176
  (1, 1)        1.0
  (2, 2)        1.0
```

Output

The result variable consists of unique words as well as the tf-if values. It can be elaborated using the below image:



```
Word indexes:
{'geeks': 1, 'for': 0, 'r2j': 2}

tf-idf value:
  (0, 0)        0.5493512310263033
  (0, 1)        0.8355915419449176
  (1, 1)        1.0
  (2, 2)        1.0
```

tf-idf value of word having index 1 i.e. *geeks* in document index 0 i.e. *d0*
Document Index
Word Index

From the above image the below table can be generated:

Document	Word	Document Index	Word Index	tf-idf value
d0	for	0	0	0.549
d0	geeks	0	1	0.8355
d1	geeks	1	1	1.000

Applications

1. **Document Similarity and Clustering:** By converting documents into numerical vectors TF-IDF enables comparison and grouping of related texts. This is valuable for clustering news articles, research papers or customer support tickets into meaningful categories.
2. **Text Classification:** It helps in identify patterns in text for spam filtering, sentiment analysis and topic classification.
3. **Keyword Extraction:** It ranks words by importance making it possible to automatically highlight key terms, generate document tags or create concise summaries.
4. **Recommendation Systems:** Through comparison of textual descriptions TF-IDF supports suggesting related articles, videos or products enhancing user engagement.

Unders
TF-IDF

TF-IDF in Action

Understanding TF-IDF

Visit Course

Article Tags :

Machine Learning Technical Scripter 2020 Python scikit-module python +1 More

Explore

Machine Learning Basics

Python for Machine Learning

Feature Engineering

Supervised Learning

Unsupervised Learning

Model Evaluation and Tuning

Advanced Techniques

Machine Learning Practice

GeeksforGeeks
Sanchhaya Education Private Limited

Corporate & Communications Address:

A-143, 7th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305)

Registered Address:

K 061, Tower K, Gulshan Vivante

Company
About Us
Legal
Privacy Policy
Contact Us
Advertise with us
GFG Corporate Solution
Campus Training Program

Explore
POTD
Job-A-Thon
Blogs
Nation Skill Up

Tutorials
Programming Languages
DSA
Web Technology
AI, ML & Data Science
DevOps
CS Core Subjects
Interview Preparation
Software and Tools

Courses
IBM Certification
DSA and Placements
Web Development
Programming Languages
DevOps & Cloud
GATE
Trending Technologies

Videos
DSA
Python
Java
C++
Web Development
Data Science
CS Subjects

Preparation Corner
Interview Corner
Aptitude
Puzzles
GfG 160
System Design