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

In [1]:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

In [2]:

```
# download and load the text6 corpus from NLTK
nltk.download("nps_chat")
text6 = nltk.corpus.nps_chat.words()
```

```
[nltk_data] Downloading package nps_chat to
[nltk_data]     C:\Users\bhavy\AppData\Roaming\nltk_data...
[nltk_data]   Package nps_chat is already up-to-date!
```

In [4]:

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

In [7]:

```
# Example documents
doc1 = "The quick brown fox jumped over the lazy dog."
doc2 = "The dog was lazy so the fox had to jump over it."
doc3 = "The cat was sleeping in the sun."

# Create a list of documents
docs = [doc1, doc2, doc3]

# Create the TF-IDF vectorizer object
vectorizer = TfidfVectorizer()

# Fit the vectorizer to the documents
tfidf_matrix = vectorizer.fit_transform(docs)

# Get the vocabulary of the documents
vocab = vectorizer.get_feature_names()

print(tfidf_matrix.toarray())
print(vocab)
```

```
[[0.38607715 0.         0.29362163 0.29362163 0.         0.
  0.         0.         0.38607715 0.29362163 0.29362163 0.38607715
  0.         0.         0.         0.45604677 0.         0.        ]
 [0.         0.         0.24955659 0.24955659 0.32813692 0.
  0.32813692 0.32813692 0.         0.24955659 0.24955659 0.
  0.         0.32813692 0.         0.38760591 0.32813692 0.24955659]
 [0.         0.40914568 0.         0.         0.         0.40914568
  0.         0.         0.         0.         0.         0.
  0.40914568 0.         0.40914568 0.48329606 0.         0.31116583]]
['brown', 'cat', 'dog', 'fox', 'had', 'in', 'it', 'jump', 'jumped', 'laz
y', 'over', 'quick', 'sleeping', 'so', 'sun', 'the', 'to', 'was']
```

In [8]:

```
import pandas as pd

# Sample data
corpus = ['This is the first document.',
          'This document is the second document.',
          'And this is the third one.',
          'Is this the first document?']

# Calculate TF-IDF
tfidf_vectorizer = TfidfVectorizer()
tfidf = tfidf_vectorizer.fit_transform(corpus)

# Convert to pandas dataframe
df = pd.DataFrame(tfidf[0].T.todense(), index=tfidf_vectorizer.get_feature_names(), co
df = df.sort_values('TF-IDF', ascending=False)
print(df)
```

```
            TF-IDF
first     0.580286
document  0.469791
is        0.384085
the       0.384085
this      0.384085
and       0.000000
one       0.000000
second    0.000000
third     0.000000
```