Bag of Words

```
from nltk.stem import PorterStemmer
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
import pandas as pd
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
porter_stemmer = PorterStemmer()
words = ["running","jumps","hovering","eating","baking"]
stemmed_words = [porter_stemmer.stem(word) for word in words]
```

```
print("Original words:",words)
print("Stemmed words:",stemmed_words)
```

```
Original words: ['running', 'jumps', 'hovering', 'eating', 'baking']
Stemmed words: ['run', 'jump', 'hover', 'eat', 'bake']
```

```
lemmatizer = WordNetLemmatizer()
print("rocks:",lemmatizer.lemmatize("rocks"))
print("rocks:",lemmatizer.lemmatize("corpa"))
print("better:", lemmatizer.lemmatize("better",pos="a"))
```

```
rocks: rock
rocks: corpa
better: good
```

```
from sklearn.feature_extraction.text import CountVectorizer

documents = ["This is an NLP project", "NLP is interesting",
             "The dog sat on the log",
              "Cats and dogs are pets."]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(documents)

print(X.toarray())
print(vectorizer.get_feature_names_out())
```

```
[[1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1]
 [0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 1 0 1 0 0 1 2 0]
 [0 1 1 1 0 1 0 0 0 0 0 1 0 0 0 0]]
['an' 'and' 'are' 'cats' 'dog' 'dogs' 'interesting' 'is' 'log' 'nlp' 'on'
 'pets' 'project' 'sat' 'the' 'this']
```

Double-click (or enter) to edit

```
text = """1. Natural Language Processing (NLP) is a branch of artificial intelligence.
2. It focuses on the interaction between computers and humans using language.
3. NLP enables machines to understand, interpret, and generate human language.
4. Applications of NLP include chatbots, translation, and sentiment analysis.
5. Preprocessing is a crucial step in NLP pipelines.
6. It involves cleaning and preparing text data for analysis.
7. Tokenization breaks text into individual words or sentences.
8. Stopword removal eliminates common words like "the" and "is."
9. Lemmatization reduces words to their base form.
10. TF-IDF is a statistical measure used to evaluate word importance.
11. It stands for Term Frequency-Inverse Document Frequency.
12. Higher TF-IDF values indicate important words in a document.
13. Sentence scoring helps in identifying key sentences in text.
14. Summarization extracts important information from a document.
15. NLP techniques improve search engines and recommendation systems.
16. Named Entity Recognition (NER) identifies proper nouns in text.
17. Part-of-Speech tagging classifies words into grammatical categories.
18. Sentiment analysis determines whether text expresses positive or negative emotions.
19. NLP is widely used in voice assistants like Siri and Alexa.
20. The future of NLP involves deep learning and large language models."""
```

```
with open("document.txt", "w", encoding="utf-8") as file:
    file.write(text)

print("document.txt has been created. You can now download it from the Files section.")
```

```
document.txt has been created. You can now download it from the Files section.
```

```
import nltk
nltk.download('punkt')
nltk.download('wordnet')
import nltk
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
from google.colab import files
uploaded = files.upload()
print(" File uploaded successfully!")
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
with open('document.txt', 'r', encoding='utf-8') as file:
    text = file.read()

print("\n Original Document Content:\n")
print(text)
```

```
 Original Document Content:

1. Natural Language Processing (NLP) is a branch of artificial intelligence.
2. It focuses on the interaction between computers and humans using language.
3. NLP enables machines to understand, interpret, and generate human language.
4. Applications of NLP include chatbots, translation, and sentiment analysis.
5. Preprocessing is a crucial step in NLP pipelines.
6. It involves cleaning and preparing text data for analysis.
7. Tokenization breaks text into individual words or sentences.
8. Stopword removal eliminates common words like "the" and "is."
9. Lemmatization reduces words to their base form.
10. TF-IDF is a statistical measure used to evaluate word importance.
11. It stands for Term Frequency-Inverse Document Frequency.
12. Higher TF-IDF values indicate important words in a document.
13. Sentence scoring helps in identifying key sentences in text.
14. Summarization extracts important information from a document.
15. NLP techniques improve search engines and recommendation systems.
16. Named Entity Recognition (NER) identifies proper nouns in text.
17. Part-of-Speech tagging classifies words into grammatical categories.
18. Sentiment analysis determines whether text expresses positive or negative emotions.
19. NLP is widely used in voice assistants like Siri and Alexa.
20. The future of NLP involves deep learning and large language models.
```

* Preprocessing → Lowercasing, punctuation removal

* Tokenization → Splitting into sentences & words

* Stopword Removal → Removing common words like is, the, on

* Lemmatization → Converting words to their base form

```
import nltk
import string
import numpy as np
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')

def preprocess_text(text):
    stop_words = set(stopwords.words('english'))
    lemmatizer = WordNetLemmatizer()
    sentences = sent_tokenize(text)
    cleaned_sentences = []

    for sentence in sentences:
        words = word_tokenize(sentence.lower())
        words = [word for word in words if word.isalnum()]
        words = [word for word in words if word not in stop_words]
        words = [lemmatizer.lemmatize(word) for word in words]
        cleaned_sentences.append(" ".join(words))

    return sentences, cleaned_sentences


text = """Natural Language Processing (NLP) is a branch of artificial intelligence.
It focuses on the interaction between computers and humans using language.
NLP enables machines to understand, interpret, and generate human language.
Applications of NLP include chatbots, sentiment analysis, and machine translation."""

sentences, cleaned_sentences = preprocess_text(text)

print("\n Sentence Tokenization Done! First 5 sentences:\n", sentences[:5])
print("\n Stopword Removal & Lemmatization Done! First 5 cleaned sentences:\n", cleaned_sentences[:5])
```

```
 Sentence Tokenization Done! First 5 sentences:
 ['Natural Language Processing (NLP) is a branch of artificial intelligence.', 'It focuses on the interaction between computers and

 Stopword Removal & Lemmatization Done! First 5 cleaned sentences:
 ['natural language processing nlp branch artificial intelligence', 'focus interaction computer human using language', 'nlp enables
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

TF IDF Matrix

```
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(cleaned_sentences)
feature_names = vectorizer.get_feature_names_out()


df_tfidf = pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names)

print("\n TF-IDF Matrix (First 10 columns shown):\n")
print(df_tfidf)
```

```
 TF-IDF Matrix (First 10 columns shown):

      10   11   12   13   14   15   16   17   18   19  ...  tokenization  \
0   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
1   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
2   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
3   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
4   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
5   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
6   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
7   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
8   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
9   0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
10  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
11  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
12  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
13  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.455936
14  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
15  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
16  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
17  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
18  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...      0.000000
```

```
19  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
20  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
21  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
22  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
23  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
24  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
25  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
26  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
27  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
28  0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  0.0  ...     0.000000
29  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
30  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  0.0  ...     0.000000
31  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
32  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  0.0  ...     0.000000
33  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
34  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  0.0  ...     0.000000
35  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
36  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  ...     0.000000
37  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
38  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000
39  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...     0.000000

    translation  understand      used     using     value     voice   whether  \
0      0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
1      0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
2      0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
3      0.000000    0.000000  0.000000  0.430149  0.000000  0.000000  0.000000
4      0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
5      0.000000    0.382531  0.000000  0.000000  0.000000  0.000000  0.000000
6      0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
7      0.410906    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
8      0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
9      0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
10     0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
11     0.000000    0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
```

```
def calculate_tfidf(cleaned_sentences):
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(cleaned_sentences)
    return tfidf_matrix, vectorizer.get_feature_names_out()

tfidf_matrix, feature_names = calculate_tfidf(cleaned_sentences)

def rank_sentences(tfidf_matrix, sentences):
    sentence_scores = np.sum(tfidf_matrix.toarray(), axis=1)
    ranked_sentences = [(sentences[i], sentence_scores[i]) for i in range(len(sentences))]
    ranked_sentences.sort(key=lambda x: x[1], reverse=True)
    return [sentence for sentence, score in ranked_sentences[:20]]

top_sentences = rank_sentences(tfidf_matrix, sentences)
```

```
print("\n TF-IDF Matrix Calculated!")
print("\n Vocabulary (First 10 Words):\n", feature_names[:])
print("\n TF-IDF Matrix Shape:", tfidf_matrix.shape)
```

```
 TF-IDF Matrix Calculated!

 Vocabulary (First 10 Words):
 ['10' '11' '12' '13' '14' '15' '16' '17' '18' '19' '20' 'alexa' 'analysis'
 'application' 'artificial' 'assistant' 'base' 'branch' 'break' 'category'
 'chatbots' 'classifies' 'cleaning' 'common' 'computer' 'crucial' 'data'
 'deep' 'determines' 'document' 'eliminates' 'emotion' 'enables' 'engine'
 'entity' 'evaluate' 'express' 'extract' 'focus' 'form' 'frequency'
 'future' 'generate' 'grammatical' 'help' 'higher' 'human' 'identifies'
 'identifying' 'importance' 'important' 'improve' 'include' 'indicate'
 'individual' 'information' 'intelligence' 'interaction' 'interpret'
 'involves' 'key' 'language' 'large' 'learning' 'lemmatization' 'like'
 'machine' 'measure' 'model' 'named' 'natural' 'negative' 'ner' 'nlp'
 'noun' 'pipeline' 'positive' 'preparing' 'preprocessing' 'processing'
 'proper' 'recognition' 'recommendation' 'reduces' 'removal' 'scoring'
 'search' 'sentence' 'sentiment' 'siri' 'stand' 'statistical' 'step'
 'stopword' 'summarization' 'system' 'tagging' 'technique' 'term' 'text'
 'tokenization' 'translation' 'understand' 'used' 'using' 'value' 'voice'
 'whether' 'widely' 'word']

 TF-IDF Matrix Shape: (40, 110)
```

Most Important Sentences

```
def rank_sentences(tfidf_matrix, sentences):
    sentence_scores = np.sum(tfidf_matrix.toarray(), axis=1)
    ranked_sentences = [(sentences[i], sentence_scores[i]) for i in range(len(sentences))]
    ranked_sentences.sort(key=lambda x: x[1], reverse=True)
    return [sentence for sentence, score in ranked_sentences[:20]]

top_sentences = rank_sentences(tfidf_matrix, sentences)

print("\n  Most Important Sentences:\n")
for i, sentence in enumerate(top_sentences, 1):
    print(f"{i}. {sentence}")
```

```
  Most Important Sentences:

1. Sentiment analysis determines whether text expresses positive or negative emotions.
2. Named Entity Recognition (NER) identifies proper nouns in text.
3. NLP is widely used in voice assistants like Siri and Alexa.
4. The future of NLP involves deep learning and large language models.
5. NLP enables machines to understand, interpret, and generate human language.
6. NLP techniques improve search engines and recommendation systems.
7. Applications of NLP include chatbots, translation, and sentiment analysis.
8. Natural Language Processing (NLP) is a branch of artificial intelligence.
9. It focuses on the interaction between computers and humans using language.
10. It involves cleaning and preparing text data for analysis.
11. Stopword removal eliminates common words like "the" and "is."
12. TF-IDF is a statistical measure used to evaluate word importance.
13. Higher TF-IDF values indicate important words in a document.
14. Tokenization breaks text into individual words or sentences.
15. Sentence scoring helps in identifying key sentences in text.
16. Summarization extracts important information from a document.
17. Lemmatization reduces words to their base form.
18. Part-of-Speech tagging classifies words into grammatical categories.
19. Preprocessing is a crucial step in NLP pipelines.
20. It stands for Term Frequency-Inverse Document Frequency.
```

Summary of the Document

```
print("\nSummary of the document:")
for sentence in top_sentences:
    print("-", sentence)
```

```
Summary of the document:
- Sentiment analysis determines whether text expresses positive or negative emotions.
- Named Entity Recognition (NER) identifies proper nouns in text.
- NLP is widely used in voice assistants like Siri and Alexa.
- The future of NLP involves deep learning and large language models.
- NLP enables machines to understand, interpret, and generate human language.
- NLP techniques improve search engines and recommendation systems.
- Applications of NLP include chatbots, translation, and sentiment analysis.
- Natural Language Processing (NLP) is a branch of artificial intelligence.
- It focuses on the interaction between computers and humans using language.
- It involves cleaning and preparing text data for analysis.
- Stopword removal eliminates common words like "the" and "is."
- TF-IDF is a statistical measure used to evaluate word importance.
- Higher TF-IDF values indicate important words in a document.
- Tokenization breaks text into individual words or sentences.
- Sentence scoring helps in identifying key sentences in text.
- Summarization extracts important information from a document.
- Lemmatization reduces words to their base form.
- Part-of-Speech tagging classifies words into grammatical categories.
- Preprocessing is a crucial step in NLP pipelines.
- It stands for Term Frequency-Inverse Document Frequency.
```

Start coding or generate with AI.

Start coding or generate with AI.