```
#Name-Suraj Sawant
#Roll no-TEB38


# DSBDA Practical No A-7: Text Analytics
# 1. Extract Sample document and apply following document preprocessing methods:
# Tokenization, POS Tagging, stop words removal, Stemming and Lemmatization.
# 2. Create representation of document by calculating Term Frequency and Inverse Document Frequency.


pip install nltk scikit-learn
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.1.8)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (2.0.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
```

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


# Download necessary NLTK datasets
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
document = "Natural language processing is a field of artificial intelligence that"


nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

```
# Tokenize the document
tokens = word_tokenize(document)
print("Tokens:", tokens)
```

```
Tokens: ['Natural', 'language', 'processing', 'is', 'a', 'field', 'of', 'artificial', 'intelligence', 'that']
```

```
nltk.download('averaged_perceptron_tagger_eng')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
True
```

```
# Perform POS tagging
pos_tags = pos_tag(tokens)
print("POS Tags:", pos_tags)
```

```
POS Tags: [('Natural', 'JJ'), ('language', 'NN'), ('processing', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('field', 'NN'), ('of', 'IN'),
```

```
# Remove stop words
stop_words = set(stopwords.words('english'))
filtered_tokens = [word for word in tokens if word.lower() not in stop_words]
print("Filtered Tokens (Stop Words Removed):", filtered_tokens)
```

```
Filtered Tokens (Stop Words Removed): ['Natural', 'language', 'processing', 'field', 'artificial', 'intelligence']
```

```
# Initialize the stemmer
stemmer = PorterStemmer()
# Perform stemming
stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]
print("Stemmed Tokens:", stemmed_tokens)
```

```
Stemmed Tokens: ['natur', 'languag', 'process', 'field', 'artifici', 'intellig']
```

```
# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()
# Perform lemmatization
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]
print("Lemmatized Tokens:", lemmatized_tokens)
```

```
Lemmatized Tokens: ['Natural', 'language', 'processing', 'field', 'artificial', 'intelligence']
```

```
# Example corpus of documents
corpus = [
"Natural language processing is a field of artificial intelligence that focuses on the interaction"
"It's an essential part of data science and machine learning.",
"Machine learning and deep learning are subfields of artificial intelligence.",
"Natural language processing helps computers understand human languages."]
```

```
# Initialize TfidfVectorizer to compute TF-IDF
vectorizer = TfidfVectorizer()
```

```
# Fit and transform the corpus to get the TF-IDF representation
tfidf_matrix = vectorizer.fit_transform(corpus)
```

```
# Convert the TF-IDF matrix to a dense matrix
dense_matrix = tfidf_matrix.todense()
```

```
# Get feature names (words in the corpus)
feature_names = vectorizer.get_feature_names_out()
```

```
# Display the TF-IDF values
print("TF-IDF Matrix:")
for i, doc in enumerate(dense_matrix):
  print(f"Document {i+1}:")
  doc_tfidf = doc.tolist()[0]
  for word, score in zip(feature_names, doc_tfidf):
      if score > 0:
        print(f" {word}: {score:.4f}")
```

```
TF-IDF Matrix:
Document 1:
 an: 0.2298
 and: 0.1747
 artificial: 0.1747
 data: 0.2298
 essential: 0.2298
 field: 0.2298
 focuses: 0.2298
 intelligence: 0.1747
 interactionit: 0.2298
 is: 0.2298
 language: 0.1747
 learning: 0.1747
 machine: 0.1747
 natural: 0.1747
 of: 0.3495
 on: 0.2298
 part: 0.2298
 processing: 0.1747
 science: 0.2298
 that: 0.2298
 the: 0.2298
Document 2:
 and: 0.2655
 are: 0.3491
 artificial: 0.2655
 deep: 0.3491
 intelligence: 0.2655
```

```
 learning: 0.5310
 machine: 0.2655
 of: 0.2655
 subfields: 0.3491
Document 3:
 computers: 0.3853
 helps: 0.3853
 human: 0.3853
 language: 0.2930
 languages: 0.3853
 natural: 0.2930
 processing: 0.2930
 understand: 0.3853
```

Start coding or generate with AI.