```
import nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from nltk. corpus import stopwords
from nltk.stem. wordnet import WordNetLemmatizer
from nltk.stem. porter import PorterStemmer
import pandas as pd


nltk.download("punkt")
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("averaged_perceptron_tagger")
```

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

```
text = "As the sun slowly set behind the majestic mountains, the cool evening breeze carried the sweet fragrance of blooming flowers, while t
example_sent = "After the thunderstorm, the mesmerizing rainbow appeared in the sky, casting a colorful reflection on the rippling surface of
sent = "Under the soft glow of the full moon, the two lovers strolled hand in hand along the deserted beach, listening to the soothing sound

words = ["Hike","Hiking", "Hiked", "Hiker", "Hikers", "Hikes", "Hikings"]


ps = PorterStemmer()
lemmatizer = WordNetLemmatizer()
```

Sentance Tokenzation

```
def sentence_tokenizer(sentance):
  return sent_tokenize(sentance)
```

Word Tokenzation

```
def word_tokenizer(sentance):
  return word_tokenize(sentance)
```

Removing Stopwords

```
def remove_stopwords(sentance):
  stop_words=set(stopwords.words('english'))
  word_tokens = word_tokenize(example_sent)
  filtered_sent = [w for w in word_tokens if not w.lower() in stop_words]
  filtered_sent = []
  for w in word_tokens:
    if w not in stop_words:
      filtered_sent.append(w)

  res = "Word Tokens: {0}, \nFiltered: {1}.".format(word_tokens, filtered_sent)

  return res
```

Stemming

```
def stemming(words):
  for w in words:
    print(w, " : ", ps.stem(w))
```

Lemmatizing

```
def lemmatizzer(word):
  res = "{0}: {1}".format(word, lemmatizer.lemmatize(word))
  return res
```

POS Tagging

```
def pos_tagger(sentance):
  tokens = nltk.word_tokenize(sentance)
  pos_tagging = nltk.pos_tag(tokens)
  return pos_tagging
```

```
print("Sentance Tokenization\n")
sentence_tokenizer(text)
```

```
Sentance Tokenization

['As the sun slowly set behind the majestic mountains, the cool evening breeze carried the sweet fragrance of blooming flowers, while
the distant sound of a waterfall echoed through the valley, creating a tranquil atmosphere for the weary travelers to rest and
rejuvenate.']
```

```
print("Word Tokenization\n")
word_tokenizer(text)
```

```
Word Tokenization

['As',
 'the',
 'sun',
 'slowly',
 'set',
 'behind',
 'the',
 'majestic',
 'mountains',
 ',',
 'the',
 'cool',
 'evening',
 'breeze',
 'carried',
 'the',
 'sweet',
 'fragrance',
 'of',
 'blooming',
 'flowers',
 ',',
 'while',
 'the',
 'distant',
 'sound',
 'of',
 'a',
 'waterfall',
 'echoed',
 'through',
 'the',
 'valley',
 ',',
 'creating',
 'a',
 'tranquil',
 'atmosphere',
 'for',
 'the',
 'weary',
 'travelers',
 'to',
 'rest',
 'and',
```

```
 'rejuvenate',
 '.']


print("Removing Stopwords\n")
remove_stopwords(example_sent)
```

```
Removing Stopwords

'Word Tokens: ['After', 'the', 'thunderstorm', ',', 'the', 'mesmerizing', 'rainbo
w', 'appeared', 'in', 'the', 'sky', ',', 'casting', 'a', 'colorful', 'reflectio
n', 'on', 'the', 'rippling', 'surface', 'of', 'the', 'lake', ',', 'as', 'the', 'b
irds', 'chirped', 'and', 'flew', 'over', 'the', 'treetops', '.'], \nFiltered: ['A
fter', 'thunderstorm', ',', 'mesmerizing', 'rainbow', 'appeared', 'sky', ',', 'ca
```

```
print("Stemming\n")
stemming(words)
```

```
Stemming

Hike  :  hike
Hiking  :  hike
Hiked  :  hike
Hiker  :  hiker
Hikers  :  hiker
Hikes  :  hike
Hikings  :  hike
```

```
print("Lemmatization\n")
lemmatizzer("Amorevole")
```

```
Lemmatization

'Amorevole: Amorevole'
```

```
print("POS Tagging\n")
pos_tagger(sent)
```

```
POS Tagging

[('Under', 'IN'),
 ('the', 'DT'),
 ('soft', 'JJ'),
 ('glow', 'NN'),
 ('of', 'IN'),
 ('the', 'DT'),
 ('full', 'JJ'),
 ('moon', 'NN'),
 (',', ','),
 ('the', 'DT'),
 ('two', 'CD'),
 ('lovers', 'NNS'),
 ('strolled', 'VBD'),
 ('hand', 'NN'),
 ('in', 'IN'),
 ('hand', 'NN'),
 ('along', 'IN'),
 ('the', 'DT'),
 ('deserted', 'JJ'),
 ('beach', 'NN'),
 (',', ','),
 ('listening', 'VBG'),
 ('to', 'TO'),
 ('the', 'DT'),
 ('soothing', 'VBG'),
 ('sound', 'NN'),
 ('of', 'IN'),
 ('waves', 'NNS'),
 ('gently', 'RB'),
 ('crashing', 'VBG'),
 ('against', 'IN'),
 ('the', 'DT'),
 ('shore', 'NN'),
 (',', ','),
 ('as', 'IN'),
 ('they', 'PRP'),
 ('shared', 'VBD'),
 ('sweet', 'JJ'),
 ('whispers', 'NNS'),
 ('and', 'CC'),
 ('stolen', 'VBN'),
```

```
      ('kisses', 'NNS'),
      (',', ','),
      ('lost', 'VBN'),
      ('in', 'IN'),
      ('the', 'DT'),
      ('blissful', 'JJ'),
      ('moment', 'NN'),
      ('of', 'IN'),
      ('their', 'PRP$'),
      ('eternal', 'JJ'),
      ('love', 'NN'),
      ('.', '.')]
```

TF-IDF

```
import math
from collections import Counter

def calculate_tf_idf(document, corpus):
    tf_idf = {}
    word_counts = Counter(document.split())
    total_documents = len(corpus)

    for word, count in word_counts.items():
        # Calculate term frequency (tf)
        tf = count / len(document.split())

        # Calculate inverse document frequency (idf)
        doc_count = sum([1 for doc in corpus if word in doc])
        if doc_count == 0:
            idf = 0
        else:
            idf = math.log(total_documents / doc_count)

        # Calculate tf-idf
        tf_idf[word] = tf * idf

    return tf_idf



document = example_sent
corpus = [
    "After the thunderstorm",
    "The mesmerizing rainbow appeared in the sky",
    "Casting a colorful reflection on the rippling surface of the lake",
    "The birds chirped and flew over the treetops"
]

tf_idf = calculate_tf_idf(document, corpus)


print(tf_idf)
```

```
{'After': 0.046209812037329684, 'the': 0.0, 'thunderstorm,': 0.0, 'mesmerizing': 0.046209812037329684, 'rainbow': 0.046209812037329684,
```