Practical No. 07

by Kartik Deshpande

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.

In [31]:
```
import nltk
import re
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /home/kartik/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/kartik/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /home/kartik/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/kartik/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package omw-1.4 to /home/kartik/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Out[31]:
```
True
```

In [32]:
```
text = "Tokenization is the first step in text analytics."
```

In [33]:
```
from nltk.tokenize import sent_tokenize
tokenized_text = sent_tokenize(text)
print(tokenized_text)
```

```
['Tokenization is the first step in text analytics.']
```

In [34]:
```
from nltk.tokenize import word_tokenize
tokenized_word = word_tokenize(text)
print(tokenized_word)
```

```
['Tokenization', 'is', 'the', 'first', 'step', 'in', 'text', 'analytics', '.']
```

In [35]:
```
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
print(stop_words)
```

```
{"haven't", 'those', 'hers', "that'll", 'were', 'won', "don't", 'you', 'did', 'shan', 'myself', 'as', 'over', "s
hould've", 'his', "wouldn't", 'of', 'hadn', 'then', 'no', 'has', 'which', 'was', 'does', 'weren', 'both', "you'v
e", 'yourself', 'we', "needn't", 're', 'when', 'some', 'again', "won't", 'on', "hasn't", 'itself', "didn't", "sh
e's", 'up', 'a', "hadn't", 'against', 'during', "mightn't", 'shouldn', "you'll", 'why', 't', 'who', 'such', 'mus
tn', 'being', 'through', 'can', 'from', 'too', 'and', 've', "shan't", 'mightn', 'wouldn', 'this', 'each', 'whom'
, "shouldn't", 'the', 'should', 'now', 'its', 'do', 'ain', 'he', 'until', 'further', 'will', 'these', 'into', 'w
hat', 'it', 'about', 'have', 'is', 'same', 'needn', 'how', 'your', "wasn't", "aren't", 'doesn', 'yours', 'once',
'doing', 'while', 'himself', 'ourselves', 'if', 'all', 'hasn', 'above', 'before', 'most', 'o', 'any', 'just', 'a
re', 'nor', 'ma', 'between', 'in', 'didn', 'had', 'other', 'y', 'to', 'been', 'they', 'after', 'be', 'don', "wer
en't", "isn't", 'down', 'our', 'under', 'my', 'isn', 'll', 'herself', "doesn't", 'me', 'at', 'by', "you'd", 'cou
ldn', 'where', 'haven', 'but', 'them', 'that', 'theirs', 'having', 'i', 'below', 'd', 'not', "it's", "mustn't",
'yourselves', 'am', 'wasn', 's', 'ours', 'she', 'only', 'him', 'themselves', 'few', "couldn't", 'very', 'for', "
you're", 'so', 'her', 'there', 'here', 'own', 'or', 'm', 'because', 'out', 'off', 'aren', 'more', 'with', 'their
', 'than', 'an'}
```

In [36]:
```
text = "How to remove stop words with NLTK library in Python?"
text = re.sub('[^a-zA-Z]','',text)
tokens = word_tokenize(text.lower())
filtered_text = []

for w in tokens:
    if w not in stop_words:
        filtered_text.append(w)

print("Tokenized Sentence :",tokens)
print("Filtered Sentence :",filtered_text)
```

```
Tokenized Sentence : ['howtoremovestopwordswithnltklibraryinpython']
Filtered Sentence : ['howtoremovestopwordswithnltklibraryinpython']
```

In [37]:

```python
from nltk.stem import PorterStemmer
e_words = ["wait", "waiting", "waited", "waits"]
ps = PorterStemmer()
for w in e_words:
    rootWord = ps.stem(w)
print(rootWord)
```

```
wait
```

In [38]:

```python
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
text = "studies studying cries cry"
tokenization = nltk.word_tokenize(text)
for w in tokenization:
    print("Lemma for {} is {}".format(w, wordnet_lemmatizer.lemmatize(w)))
```

```
Lemma for studies is study
Lemma for studying is studying
Lemma for cries is cry
Lemma for cry is cry
```

In [39]:

```python
from nltk.tokenize import word_tokenize
data = "The pink sweater fit her perfectly"
words = word_tokenize(data)
for word in words:
    print(nltk.pos_tag([word]))
```

```
[('The', 'DT')]
[('pink', 'NN')]
[('sweater', 'NN')]
[('fit', 'NN')]
[('her', 'PRP$')]
[('perfectly', 'RB')]
```

In [40]:

```python
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer

d0 = 'Jupiter is the largest Planet'
d1 = 'Mars is the fourth planet from the sun'
string = [d0,d1]
tfidf = TfidfVectorizer()
result = tfidf.fit_transform(string)
print('Word indices:', tfidf.vocabulary_)
print('TF-IDF Values:', result)
```

```
Word indices: {'jupiter': 3, 'is': 2, 'the': 8, 'largest': 4, 'planet': 6, 'mars': 5, 'fourth': 0, 'from': 1, 's
un': 7}
TF-IDF Values:   (0, 6)	0.3793034928087496
  (0, 4)	0.5330978245262535
  (0, 8)	0.3793034928087496
  (0, 2)	0.3793034928087496
  (0, 3)	0.5330978245262535
  (1, 7)	0.37695708675831013
  (1, 1)	0.37695708675831013
  (1, 0)	0.37695708675831013
  (1, 5)	0.37695708675831013
  (1, 6)	0.2682080718928097
  (1, 8)	0.5364161437856194
  (1, 2)	0.2682080718928097
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js