Text Analytics

## Extract Sample document and apply following document preprocessing methods: Tokenization, Part of Speech (POS) Tagging, stop words removal, Stemming and Lemmatization.

Create representation of document by calculating Term Frequency and Inverse Document Frequency.

In [1]:

**import nltk**

**from nltk import sent\_tokenize from nltk import word\_tokenize**

In [2]:

**!pip install textblob**

/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int\_from\_bytes is deprecated, us from cryptography.utils import int\_from\_bytes

/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int\_from\_bytes is deprecated, use in from cryptography.utils import int\_from\_bytes

Defaulting to user installation because normal site-packages is not writeable

Requirement already satisfied: textblob in /home/ihack-pc/.local/lib/python3.8/site-packages (0.17.1)

Requirement already satisfied: nltk>=3.1 in /home/ihack-pc/.local/lib/python3.8/site-packages (from textblob) (3.7) Requirement already satisfied: click in /home/ihack-pc/.local/lib/python3.8/site-packages (from nltk>=3.1->textblob) (7.1.2) Requirement already satisfied: tqdm in /home/ihack-pc/.local/lib/python3.8/site-packages (from nltk>=3.1->textblob) (4.47.0) Requirement already satisfied: regex>=2021.8.3 in /home/ihack-pc/.local/lib/python3.8/site-packages (from nltk>=3.1->textblob Requirement already satisfied: joblib in /home/ihack-pc/.local/lib/python3.8/site-packages (from nltk>=3.1->textblob) (0.16.0

In [3]:

**import textblob**

**from textblob import TextBlob**

In [4]:

# text

**"Hello everyone! Welcome to my blog post on Medium. We are studying Natural Language Proces**

**tokens\_sents**

**nltk.sent\_tokenize(text)**

**print(tokens\_sents)**

**tokens\_words**

**nltk.word\_tokenize(text)**

**print(tokens\_words)**

In [5]:

**import nltk nltk.download('punkt')**

**=**

[nltk\_data] Downloading package punkt to /home/ihack-pc/nltk\_data... [nltk\_data] Package punkt is already up-to-date!

Out [5]: True

In [6]:

**import nltk nltk.download('averaged\_perceptron\_tagger')**

[nltk\_data] Downloading package averaged\_perceptron\_tagger to [nltk\_data] /home/ihack-pc/nltk\_data...

[nltk\_data] Package averaged\_perceptron\_tagger is already up-to- [nltk\_data] date!

Out [6]: True

In [7]:

**import nltk nltk.download('stopwords')**

[nltk\_data] Downloading package stopwords to /home/ihack- [nltk\_data] pc/nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

Out [7]: True

In [8]:

**TextBlob(text).words**

Out [8]: WordList(['Hello', 'everyone', 'Welcome', 'to', 'my', 'blog', 'post', 'on', 'Medium', 'We', 'are', 'studying', 'Natural', 'Language', 'Processing'])

## Tokenization

In [9]:

**=**

['Hello everyone!', 'Welcome to my blog post on Medium.', 'We are studying Natural Language Processing.']

In [10]:

**=**

['Hello', 'everyone', '!', 'Welcome', 'to', 'my', 'blog', 'post', 'on', 'Medium', '.', 'We', 'are', 'studying', 'Natural', 'L

## Part of Speech (POS) Tagging

In [11]:

**pos = nltk.pos\_tag(tokens\_words) print(pos)**

[('Hello', 'NNP'), ('everyone', 'NN'), ('!', '.'), ('Welcome', 'UH'), ('to', 'TO'), ('my', 'PRP$'), ('blog', 'NN'), ('post',

## Stop Words Removal

In [12]:

**!pip install stop-words**

/usr/lib/python3/dist-packages/secretstorage/dhcrypto.py:15: CryptographyDeprecationWarning: int\_from\_bytes is deprecated, us from cryptography.utils import int\_from\_bytes

/usr/lib/python3/dist-packages/secretstorage/util.py:19: CryptographyDeprecationWarning: int\_from\_bytes is deprecated, use in from cryptography.utils import int\_from\_bytes

Defaulting to user installation because normal site-packages is not writeable

Requirement already satisfied: stop-words in /home/ihack-pc/.local/lib/python3.8/site-packages (2018.7.23)

In [13]:

**import nltk**

**from nltk.corpus import stopwords set(stopwords.words('english'))**

Out [13]: {'a',

'about',

'above',

'after',

'again', 'against', 'ain',

'all',

'am',

'an',

'and',

'any',

'are',

'aren',

"aren't",

'as',

'at',

'be', 'because', 'been',

'before',

'being',

'below', 'between', 'both',

'but',

'by',

'can',

'couldn',

"couldn't",

'd',

'did',

'didn',

"didn't",

'do',

'does',

'doesn',

"doesn't",

'doing',

'don',

"don't",

'down',

'during',

'each',

'few',

'for',

'from', 'further', 'had',

'hadn',

"hadn't",

'has',

'hasn',

"hasn't",

'have',

'haven',

"haven't",

'having',

'he',

'her',

'here',

'hers', 'herself', 'him', 'himself', 'his',

'how',

'i',

'if',

'in',

'into',

'is',

'isn',

"isn't",

'it',

"it's",

'its',

'itself',

'just',

'll',

'm',

'ma',

'me',

'mightn',

"mightn't",

'more',

'most',

'mustn',

"mustn't",

'my',

'myself',

'needn',

"needn't",

'no',

'nor',

'not',

'now',

'o',

'of',

'off',

'on',

'once',

'only',

'or',

'other',

'our',

'ours', 'ourselves', 'out',

'over',

'own',

're',

's',

'same',

'shan',

"shan't",

'she',

"she's",

'should',

"should've", 'shouldn', "shouldn't",

'so',

'some',

'such',

't',

'than',

'that',

"that'll",

'the',

'their',

'theirs',

'them', 'themselves', 'then',

'there',

'these',

'they',

'this',

'those', 'through', 'to',

'too',

'under',

'until',

'up',

've',

'very',

'was',

'wasn',

"wasn't",

'we',

'were',

'weren',

"weren't",

'what',

'when',

'where',

'which',

'while',

'who',

'whom',

'why',

'will',

'with',

'won',

"won't",

'wouldn',

"wouldn't",

'y',

'you',

"you'd",

"you'll",

"you're",

"you've",

'your',

'yours', 'yourself', 'yourselves'}

**import pandas as pd import numpy as np**

**n\_docs = len(corpus)**

**#·Number of documents in the corpus**

**n\_words\_set = len(words\_set) #·Number of unique words in the**

**df\_tf = pd.DataFrame(np.zeros((n\_docs, n\_words\_set)), columns=words\_set) # Compute Term Frequency (TF)**

**for i in range(n\_docs):**

**words = corpus[i].split(' ') # Words in the document for w in words:**

**df\_tf[w][i] = df\_tf[w][i] + (1 / len(words))**

**df\_tf**

## Stemming and Lemmatization

In [15]:

**from nltk.stem import PorterStemmer from nltk.stem import LancasterStemmer**

In [16]:

**#create an object of class PorterStemmer porter = PorterStemmer() lancaster=LancasterStemmer()**

**#proide a word to be stemmed print("Porter Stemmer") print(porter.stem("cats")) print(porter.stem("trouble")) print(porter.stem("troubling")) print(porter.stem("troubled")) print("Lancaster Stemmer") print(lancaster.stem("cats")) print(lancaster.stem("trouble")) print(lancaster.stem("troubling")) print(lancaster.stem("troubled"))**

Porter Stemmer cat

troubl troubl troubl

Lancaster Stemmer cat

troubl troubl troubl

Create representation of document by calculating Term Frequency and Inverse Document Frequency.

In [17]:

**corpus = ['data science is one of the most important fields of science', 'this is one of the best data science courses',**

**'data scientists analyze data' ]**

In [18]:

**words\_set = set()**

**for doc in corpus:**

**words = doc.split(' ')**

**words\_set = words\_set.union(set(words))**

**print('Number of words in the corpus:',len(words\_set)) print('The words in the corpus: \n', words\_set)**

Number of words in the corpus: 14 The words in the corpus:

{'most', 'analyze', 'the', 'data', 'of', 'one', 'is', 'important', 'best', 'courses', 'fields', 'scientists', 'this', 'scien

In [21]:

## s

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Out [21]:** | most | | analyze | the | data | of | one | is | important | best | courses | ﬁelds | scienti |
| 0 | | 0.090909 | 0.00 | 0.090909 | 0.090909 | 0.181818 | 0.090909 | 0.090909 | 0.090909 | 0.000000 | 0.000000 | 0.090909 | 0.00 |
| 1 | | 0.000000 | 0.00 | 0.111111 | 0.111111 | 0.111111 | 0.111111 | 0.111111 | 0.000000 | 0.111111 | 0.111111 | 0.000000 | 0.00 |
| 2 | | 0.000000 | 0.25 | 0.000000 | 0.500000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.25 |