Yash Verma
CSE-AI & ML
3RD YEAR
500069950
ROLL NO.110
B2 BATCH
LAB-2 ASSIGNMENT
COMPUTATIONAL LINGUISTICS AND NLP LAB

```
# -*- coding: utf-8 -*-
"""Untitled14.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1z1xUumdt6Rvsd6-N5epzHGU--WzoW7m8
"""
```

NATURAL LANGUAGE PROCESSING (NLP)

Traditionally, feeding statistics and models have been the method of choice for interpreting phrases. Recent advances in this area include voice recognition software, human language translation, information retrieval and artificial intelligence. There is difficulty in developing human language translation software because language is constantly changing. Natural language processing is also being developed to create human readable text and to translate between one human language and another. The ultimate goal of NLP is to build software that will analyze, understand and generate human languages naturally, enabling communication with a computer as if it were a human.

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming,

tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

The Natural Language Toolkit is an open source library for the Python programming language originally written by Steven Bird, Edward Loper and Ewan Klein for use in development and education.
It comes with a hands-on guide that introduces topics in computational linguistics as well as programming fundamentals for Python which makes it suitable for linguists who have no deep knowledge in programming, engineers and researchers that need to delve into computational linguistics, students and educators.

These are the most prominent areas where we can use Text Analytics and NLP with the help of NLTK library:

- Compare Text Analytics, NLP and Text Mining
- Text Analysis Operations using NLTK
- Tokenization
- Stopwords
- Lexicon Normalization such as Stemming and Lemmatization
- POS Tagging
- Sentiment Analysis
- Text Classification
- Performing Sentiment Analysis using Text Classification

Tokenization

Tokenization is the first step in text analytics. The process of breaking down a text paragraph into smaller chunks such as words or sentence is called Tokenization. Token is a single entity that is building blocks for sentence or paragraph.

Sentence Tokenization

Sentence tokenizer breaks text paragraph into sentences.

Word Tokenization

Breaks sentence into words

```
import nltk


text= """Monte Carlo and Las Vegas are two
Randomization algorithms developed in 90s , by
some developers AND I AM ADDING THIS
TEXT TO ONLY CHECK ABOUT THE FREQUENCY
DISTRIBUTION BY REPEATING WORDS LIKE
DISTRIBUTION AND MONTE CARLO"""


import regex
regex.split("[\s\.\,]",text)
```

```
[25] ['Monte',
      'Carlo',
      'and',
      'Las',
      'Vegas',
      'are',
      'two',
      'Randomization',
      'algorithms',
      'developed',
      'in',
      '90s',
      '',
      '',
      'by',
      'some',
      'developers',
      'AND',
      'I',
      'AM',
      'ADDING',
      'THIS',
      '',
      'TEXT',
      'TO',
      'ONLY',
      'CHECK',
      'ABOUT',
      'THE',
      'FREQUENCY',
      'DISTRIBUTION',
      'BY',
      'REPEATING',
      'WORDS',
      'LIKE',
      'DISTRIBUTION',
      'AND',
      'MONTE',
      'CARLO']
```

```
nltk.download('punkt')
nltk.word_tokenize(text)
```

```
[26]
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
['Monte',
 'Carlo',
 'and',
 'Las',
 'Vegas',
 'are',
 'two',
 'Randomization',
 'algorithms',
 'developed',
 'in',
 '90s',
 ',',
 'by',
 'some',
 'developers',
 'AND',
 'I',
 'AM',
 'ADDING',
 'THIS',
 'TEXT',
 'TO',
 'ONLY',
 'CHECK',
 'ABOUT',
 'THE',
 'FREQUENCY',
 'DISTRIBUTION',
 'BY',
 'REPEATING',
 'WORDS',
 'LIKE',
 'DISTRIBUTION',
 'AND',
 'MONTE',
 'CARLO']
```

Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and obviously in natural language processing (NLP) as well.

It is a process of linguistic normalization, which reduces words to their word root word or chops off the derivational affixes. For example, connection, connected, connecting word reduce to a common word "connect".
We will discuss few types of stemming here:

1. Porter Stemmer
2. Snowball Stemmer

"""

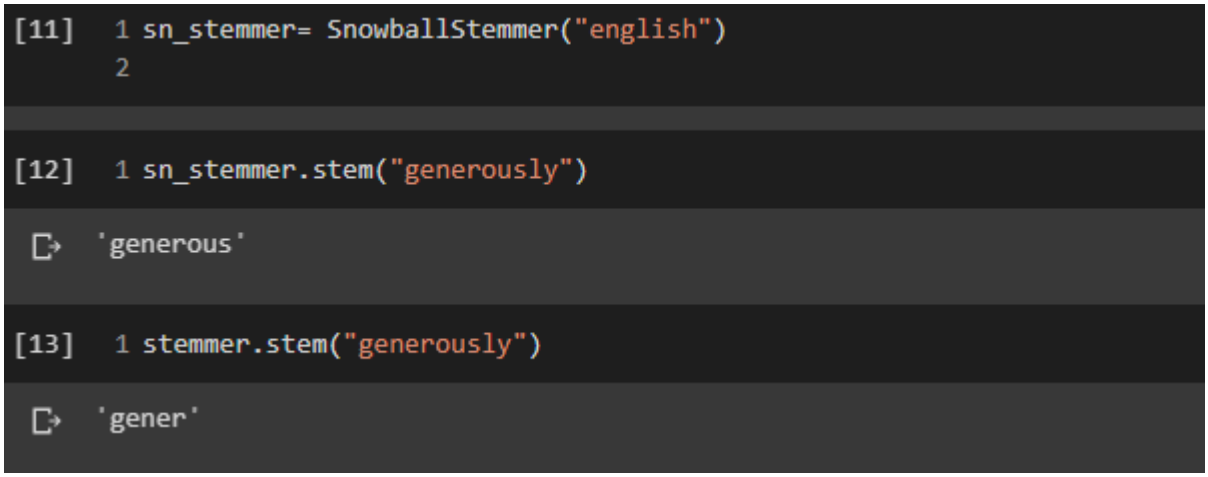
```
from nltk.stem import PorterStemmer
stemmer= PorterStemmer()
plurals=['caresses','flies','dies','mules','denied','died','agreed','owned','humbled','sized','meeting','stating','seizing','itemization',

'sensational','traditional','reference','colonizer','plotted']
for word in plurals:
  print(f"{word}>>>{stemmer.stem(word)}")
```

```
caresses>>>caress
flies>>>fli
dies>>>die
mules>>>mule
denied>>>deni
died>>>die
agreed>>>agre
owned>>>own
humbled>>>humbl
sized>>>size
meeting>>>meet
stating>>>state
seizing>>>seiz
itemization>>>item
sensational>>>sensat
traditional>>>tradit
reference>>>refer
colonizer>>>colon
plotted>>>plot
```

```
from nltk.stem.snowball import SnowballStemmer
SnowballStemmer.languages
```

```
('arabic',
 'danish',
 'dutch',
 'english',
 'finnish',
 'french',
 'german',
 'hungarian',
 'italian',
 'norwegian',
 'porter',
 'portuguese',
 'romanian',
 'russian',
 'spanish',
 'swedish')
```

```
sn_stemmer= SnowballStemmer("english")
sn_stemmer.stem("generously")
stemmer.stem("generously")
```



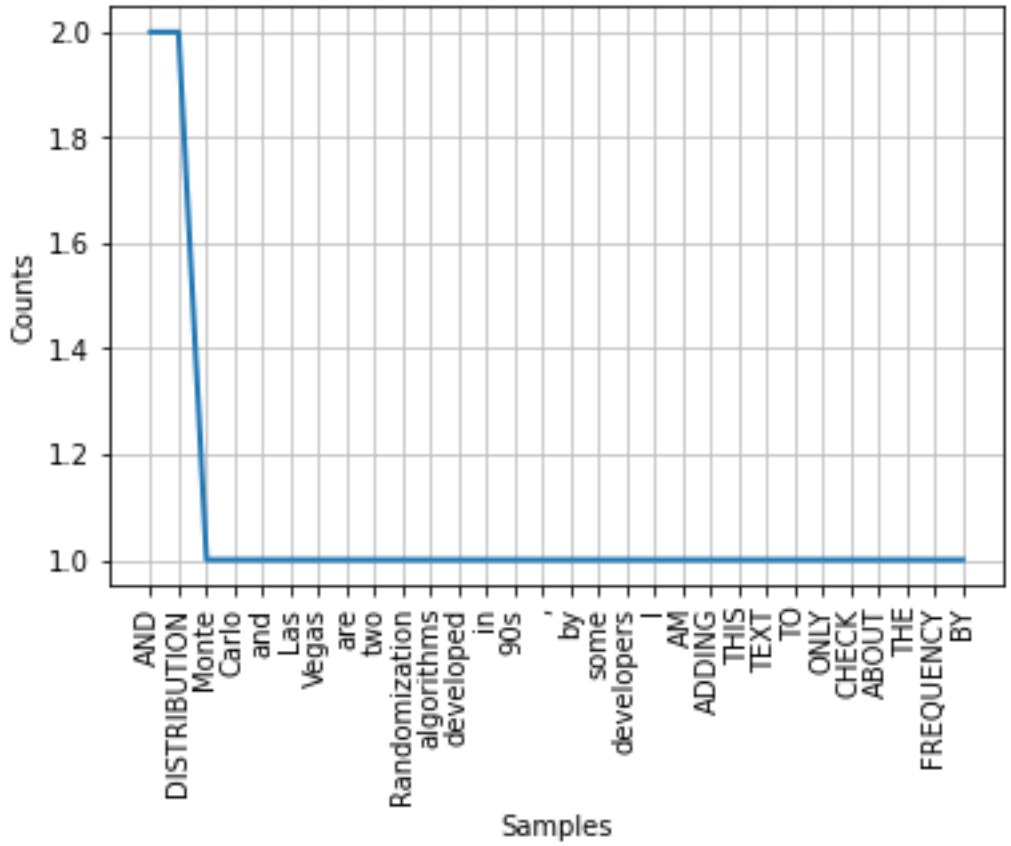
Lemmatization: retrieving the source of the word

Lemmatization reduces words to their base word, which is linguistically correct lemmas. It transforms root word with the use of vocabulary and morphological analysis. Lemmatization is usually more sophisticated than stemming. Stemmer works on an individual word without knowledge of the context. For example, The word "better" has "good" as its lemma. This thing will miss by stemming because it requires a dictionary look-up.

```
from nltk.stem import WordNetLemmatizer
lemmatizer= WordNetLemmatizer()

import nltk
nltk.download('wordnet')

for word in plurals:

print(f"{word}>>>{lemmatizer.lemmatize(word)}")
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/wordnet.zip.
caresses>>>caress
flies>>>fly
dies>>>dy
mules>>>mule
denied>>>denied
died>>>died
agreed>>>agreed
owned>>>owned
humbled>>>humbled
sized>>>sized
meeting>>>meeting
stating>>>stating
seizing>>>seizing
itemization>>>itemization
sensational>>>sensational
traditional>>>traditional
reference>>>reference
colonizer>>>colonizer
plotted>>>plotted
```

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)

from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
#<FreqDist with 25 samples and 30 outcomes>
fdist.most_common(2)
[('is', 3), (',', 2)]
# Frequency Distribution Plot
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```


2.0
1.8
1.6
1.4
1.2
1.0
Counts
AND
DISTRIBUTION
Monte
Carlo
and
Las
Vegas
are
two
Randomization
algorithms
developed
in
90s
'
by
some
developers
I
AM
ADDING
THIS
TEXT
TO
ONLY
CHECK
ABOUT
THE
FREQUENCY
BY
Samples

Stopwords

Stopwords considered as noise in the text. Text may contain stop words such as is, am, are, this, a, an, the, etc.

In NLTK for removing stopwords, you need to create a list of stopwords and filter out your list of tokens from these words.

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

from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
{'the', 'in', 'does', 'an', 'if', 'at', 'while', 'so', 'more', 'itself',
'where', 'shan', "should've", 'each', 'there', 'after', 'further', 'then',
'had', 'a', "mightn't", 'as', 're', 'myself', 'mustn', 'your', 'her',
'own', 'herself', "shan't", 'into', 'll', "she's", 'until', 'don', 'and',
'out', 'between', 'on', 'whom', "hadn't", 'i', 'y', 'needn', "that'll",
'wouldn', "you're", 'm', 'with', 'through', 'him', 'they', 'below',
'being', 'down', 'did', 'yourself', 'o', 've', "weren't", 'yours',
"wouldn't", "you'll", 'his', 'been', 'off', 'themselves', 'when', 'ain',
"aren't", 'hasn', "shouldn't", 'ours', 'for', 'by', 'from', 'same', 'am',
'd', "didn't", 'than', 'of', 'ma', 'weren', 'hers', 'won', 'aren', 'its',
"needn't", 'me', 'up', 'having', 'during', 'all', 'he', 'she',
'ourselves', 'theirs', 'were', 'do', "isn't", 'some', "doesn't",
"haven't", 'to', 'my', 'are', 'was', 'doing', 'what', 'both', 'most',
'any', 'very', "mustn't", 'our', 'yourselves', 'we', 'mightn', 'that',
'again', 'wasn', "you've", 'be', 'their', "hasn't", 'those', 'not',
'because', 'before', 'nor', 'will', "wasn't", 'but', 'over', 'under',
'you', 'which', "won't", "you'd", 'doesn', 'above', 'haven', 'them',
```

```
'has', 'himself', 'about', 'this', 'have', 'too', 'isn', 'why', 'such',
'other', 'only', "it's", 'it', 'or', 'these', 'couldn', 'who', 'hadn',
'against', 'didn', 'once', "couldn't", 'can', 's', 'now', 'should', 'is',
't', 'how', 'few', 'here', 'just', 'shouldn', "don't", 'no'}
```

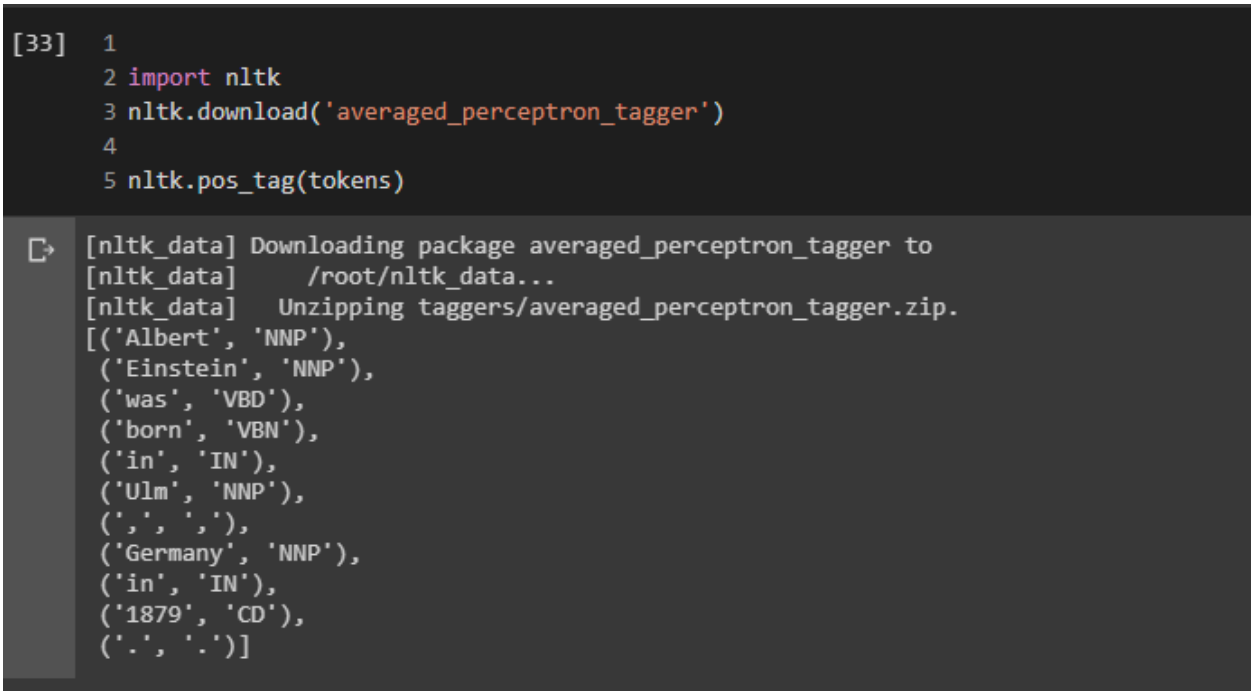
POS Tagging

The primary target of Part-of-Speech(POS) tagging is to identify the grammatical group of a given word. Whether it is a NOUN, PRONOUN, ADJECTIVE, VERB, ADVERBS, etc. based on the context. POS Tagging looks for relationships within the sentence and assigns a corresponding tag to the word.

```
sent = "Albert Einstein was born in Ulm, Germany in 1879."
tokens=nltk.word_tokenize(sent)
print(tokens)
```

```
['Albert', 'Einstein', 'was', 'born', 'in', 'Ulm', ',', 'Germany', 'in',
'1879', '.']
```

```
import nltk
nltk.download('averaged_perceptron_tagger')

nltk.pos_tag(tokens)
```



Lexicon Normalization

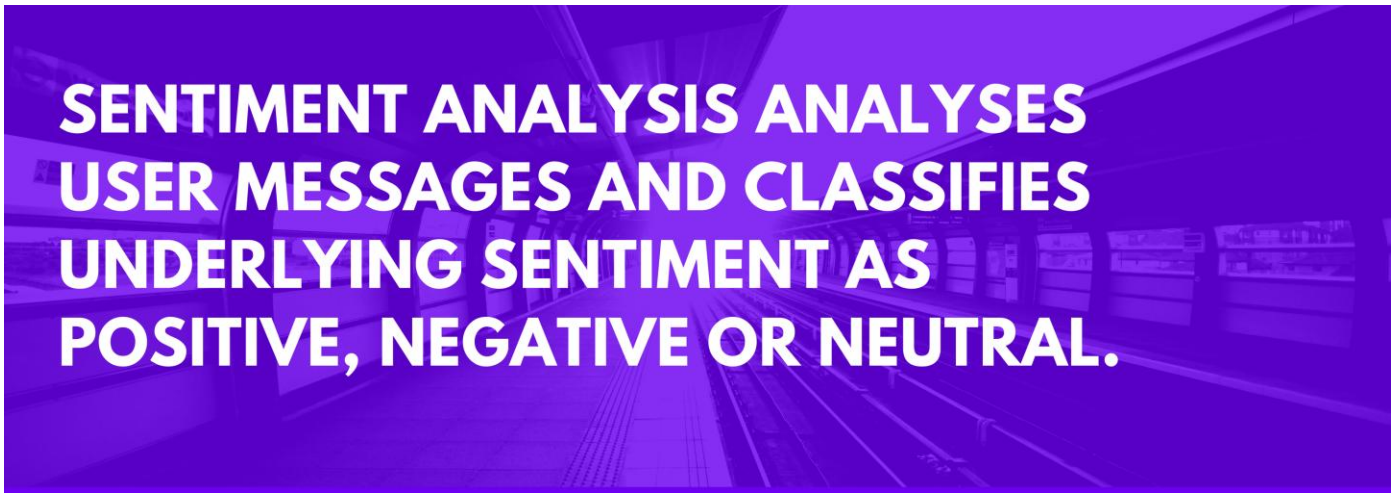
Lexicon normalization considers another type of noise in the text. For example, connection, connected, connecting word reduce to a common word "connect". It reduces derivationally related forms of a word to a common root word.

Sentiment Analysis

Nowadays companies want to understand, what went wrong with their latest products? What users and the general public think about the latest feature? You can quantify such information with reasonable accuracy using sentiment analysis.

Quantifying users content, idea, belief, and opinion is known as sentiment analysis. User's online post, blogs, tweets, feedback of product helps business people to the target audience and innovate in products and services. Sentiment analysis helps in understanding people in a better and more accurate way. It is not only limited to marketing, but it can also be utilized in politics, research, and security.

Human communication just not limited to words, it is more than words. Sentiments are combination words, tone, and writing style. As a data analyst, It is more important to understand our sentiments, what it really means?



There are mainly two approaches for performing sentiment analysis.

- Lexicon-based: count number of positive and negative words in given text and the larger count will be the sentiment of text.
- Machine learning based approach: Develop a classification model, which is trained using the pre-labeled dataset of positive, negative, and neutral.

Text Classification

Text classification is one of the important tasks of text mining. It is a supervised approach. Identifying category or class of given text such as a blog, book, web page, news

articles and tweets. It has various applications in today's computer world such as spam detection, task categorization in CRM services, categorizing products on E-retailer websites, classifying the content of websites for a search engine, sentiments of customer feedback, etc.