Tokenization,Stemming, Lemmatization, Stop Words

1. Tokenization

- Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. One can think of token as parts like a word is a token in a sentence, and a sentence is a token in a paragraph.

Code #1: Sentence Tokenization – Splitting sentences in the paragraph

In [1]:

```
from nltk.tokenize import sent_tokenize

text = "Hello Mr. Smith, how are you doing today. ?"
sent_tokenize(text)
```

Out[1]:

```
['Hello Mr. Smith, how are you doing today.', '?']
```

How sent_tokenize works ?

The sent_tokenize function uses an instance of PunktSentenceTokenizer from the nltk.tokenize.punkt module, which is already been trained and thus very well knows to mark the end and beginning of sentence at what characters and punctuation.

Code #: Word Tokenization – Splitting words in a sentence.

In [2]:

```
from nltk.tokenize import word_tokenize

word_tokenize(text)
```

Out[2]:

```
['Hello', 'Mr.', 'Smith', ',', 'how', 'are', 'you', 'doing', 'today', '.',
'?']
```

How word_tokenize works?

word_tokenize() function is a wrapper function that calls tokenize() on an instance of the TreebankWordTokenizer class.

2. Stemming

- Refers to the process of slicing the end or the beginning of words with the intention of removing affixes (lexical additions to the root of the word).

- Affixes that are attached at the beginning of the word are called prefixes (e.g. “astro” in the word “astrobiology”) and the ones attached at the end of the word are called suffixes (e.g. “ful” in the word “helpful”).
- A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.
- Stemming is an important part of the pipelining process in Natural language processing. The input to the stemmer is tokenized words.

```
 Some more example of stemming for root word "like" include:
->"likes"
->"liked"
->"likely"
->"liking"
```

Errors in Stemming:

- There are mainly two errors in stemming –

```
  1. over-stemming:- occurs when two words are stemmed from the same root that
 are of different stems. Over-stemming can also be regarded as false-positives.
  2. under-stemming :- occurs when two words are stemmed from the same root tha
t are not of different stems. Under-stemming can be interpreted as false-negati
ves.
```

Some Stemming algorithms are:

```
1. Porter’s Stemmer algorithm
```

Advantage: It produces the best output as compared to other stemmers and it has less error rate.

Limitation: Morphological variants produced are not always real words.

```
2. Snowball Stemmer:
```

- When compared to the Porter Stemmer, the Snowball Stemmer can map non-English words too.
- Since it supports other languages the Snowball Stemmers can be called a multi-lingual stemmer.
- The Snowball stemmers are also imported from the nltk package. This stemmer is based on a programming language called ‘Snowball’ that processes small strings and is the most widely used stemmer.
- The Snowball stemmer is way more aggressive than Porter Stemmer and is also referred to as Porter2 Stemmer. Because of the improvements added when compared to the Porter Stemmer, the Snowball stemmer is having greater computational speed.

In [3]:

```python
# import these modules
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

# choose some words to be stemmed
words = ["program", "programs", "programer", "programing", "programers"]

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

```
program  :  program
programs  :  program
programer  :  program
programing  :  program
programers  :  program
```

In [4]:

```python
## Stemming words from sentences
#importing modules
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize  #tokenization

ps = PorterStemmer()

sentence = "Programers program with programing languages"
words = word_tokenize(sentence)

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

```
Programers  :  program
program  :  program
with  :  with
programing  :  program
languages  :  languag
```

In [5]:

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

#the stemmer requires a language parameter
snow_stemmer = SnowballStemmer(language='english')
porter_stem = PorterStemmer()

#list of tokenized words
words = ['cared','university','fairly','easily','singing','sings','sung','singer','sporting

#stem's of each word
print('SnowballStemmer \n')
for i in words:
    print(i, " : ", snow_stemmer.stem(i))
print('\nPorterStemmer\n')
for j in words:
    print(j, " : ", porter_stem.stem(j))
```

```
SnowballStemmer 

cared  :  care
university  :  univers
fairly  :  fair
easily  :  easili
singing  :  sing
sings  :  sing
sung  :  sung
singer  :  singer
sportingly  :  sport

PorterStemmer

cared  :  care
university  :  univers
fairly  :  fairli
easily  :  easili
singing  :  sing
sings  :  sing
sung  :  sung
singer  :  singer
sportingly  :  sportingli
```

3. Lemmatization

- Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meaning to one word.
- Text preprocessing includes both Stemming as well as Lemmatization. Many times people find these two terms confusing. Some treat these two as same. Actually, lemmatization is preferred over Stemming because lemmatization does morphological analysis of the words.

```
Examples of lemmatization:

-> rocks : rock
-> corpora : corpus
-> better : good
```

In [6]:

```python
# import these modules
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

words = ['rocks','corpora','better']

for j in words:
    print(j, " : ", lemmatizer.lemmatize(j))

print('\nPorterStemmer\n')

for k in words:
    print(k, " : ", porter_stem.stem(k))
```

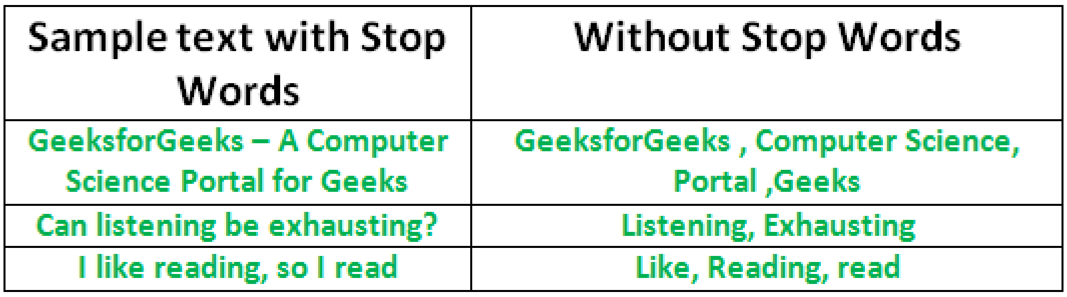
```
rocks  :  rock
corpora  :  corpus
better  :  better

PorterStemmer

rocks  :  rock
corpora  :  corpora
better  :  better
```

Different Types Lemmatization :- [https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/ (https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/)](https://www.geeksforgeeks.org/python-lemmatization-approaches-with-examples/)

4. Stop Words



Sample text with Stop Words	Without Stop Words
GeeksforGeeks – A Computer Science Portal for Geeks	GeeksforGeeks , Computer Science, Portal ,Geeks
Can listening be exhausting?	Listening, Exhausting
I like reading, so I read	Like, Reading, read

- Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

In [7]:

```
import nltk
from nltk.corpus import stopwords
print("These Are All Stops Word That We Have To Remove\n")
print(stopwords.words('english'))
```

```
These Are All Stops Word That We Have To Remove

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

Removing stop words

In [8]:

```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

example_sent = """This is a sample sentence,
                  showing off the stop words filtration."""

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)

filtered_sentence = [w for w in word_tokens if not w in stop_words]

filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print("With Stop Word: ",word_tokens)
print("After Remove Stops Word: ",filtered_sentence)
```

```
With Stop Word:  ['This', 'is', 'a', 'sample', 'sentence', ',', 'showing',
'off', 'the', 'stop', 'words', 'filtration', '.']
After Remove Stops Word:  ['This', 'sample', 'sentence', ',', 'showing', 'st
op', 'words', 'filtration', '.']
```

Now All Put It on to One Paragraph

In [9]:

```python
import nltk

paragraph = """Thank you all so very much. Thank you to the Academy.
            Thank you to all of you in this room. I have to congratulate
               the other incredible nominees this year. The Revenant was
               the product of the tireless efforts of an unbelievable cast
               and crew. First off, to my brother in this endeavor, Mr. Tom
               Hardy. Tom, your talent on screen can only be surpassed by
               your friendship off screen … thank you for creating a t
               ranscendent cinematic experience. Thank you to everybody at
               Fox and New Regency … my entire team. I have to thank
               everyone from the very onset of my career … To my parents;
               none of this would be possible without you. And to my
               friends, I love you dearly; you know who you are. And lastly,
               I just want to say this: Making The Revenant was about
               man's relationship to the natural world. A world that we
               collectively felt in 2015 as the hottest year in recorded
               history. Our production needed to move to the southern
               tip of this planet just to be able to find snow. Climate
               change is real, it is happening right now. It is the most
               urgent threat facing our entire species, and we need to work
               collectively together and stop procrastinating. We need to
               support leaders around the world who do not speak for the
               big polluters, but who speak for all of humanity, for the
               indigenous people of the world, for the billions and
               billions of underprivileged people out there who would be
               most affected by this. For our children’s children, and
               for those people out there whose voices have been drowned
               out by the politics of greed. I thank you all for this
               amazing award tonight. Let us not take this planet for
               granted. I do not take tonight for granted. Thank you so very much."""


# text Preprocessing
import re
from nltk.corpus import stopwords #stopword
from nltk.stem.porter import PorterStemmer #for Stemming
from nltk.stem import WordNetLemmatizer #for Lemmatization

ps = PorterStemmer() #inialize stemmer
wordnet=WordNetLemmatizer() #inialize Lemmatize

## You can use any one but Lemmatize is better than stemmer

#step- 1
sentences = nltk.sent_tokenize(paragraph) #sentence Tokenize

corpus = []

for i in range(len(sentences)):
    # step 2 using some preprocess like remove unnecessary word,lower the sentence etc
    review = re.sub('[^a-zA-Z]', ' ', sentences[i])
    review = review.lower()
    review = review.split()

    #step3 use stemmer and remove stopwords in sentence
#     review = [ps.stem(word) for word in review if not word in set(stopwords.words('englis
    #step3 use Lemmatize and remove stopwords in sentence Use any one of these.
    review = [wordnet.lemmatize(word) for word in review if word not in set(stopwords.words

    review = ' '.join(review)
    corpus.append(review)
```

In [10]:

```
corpus
```

Out[10]:

```
['thank much',
 'thank academy',
 'thank room',
 'congratulate incredible nominee year',
 'revenant product tireless effort unbelievable cast crew',
 'first brother endeavor mr tom hardy',
 'tom talent screen surpassed friendship screen thank creating ranscendent c
inematic experience',
 'thank everybody fox new regency entire team',
 'thank everyone onset career parent none would possible without',
 'friend love dearly know',
 'lastly want say making revenant man relationship natural world',
 'world collectively felt hottest year recorded history',
 'production needed move southern tip planet able find snow',
 'climate change real happening right',
 'urgent threat facing entire specie need work collectively together stop pr
ocrastinating',
 'need support leader around world speak big polluter speak humanity indigen
ous people world billion billion underprivileged people would affected',
 'child child people whose voice drowned politics greed',
 'thank amazing award tonight',
 'let u take planet granted',
 'take tonight granted',
 'thank much']
```

Now Our Data Cleaned So We Go towards The text vectorization using Bag_Of_Words(Count Vectorizer) ,TFIDF, Unigram,BiGram,n-Gram,Word2vc and Average Word2vec