NLP PROGRAM-1

A program to use different Tokenization

Soundarya G_ 2048057

TOKENIZATION:

> It 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.

Import Required Modules

```
import nltk
```

```
nltk.download('all')
```

```
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package universal_treebanks_v20 to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package verbnet to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet.zip.
[nltk_data]    | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/verbnet3.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet.zip.
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package ycoe to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ycoe.zip.
[nltk_data]    | Downloading package rslp to /root/nltk_data...
[nltk_data]    |   Unzipping stemmers/rslp.zip.
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Unzipping tokenizers/punkt.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package sample_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/sample_grammars.zip.
[nltk_data]    | Downloading package spanish_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/spanish_grammars.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package large_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/large_grammars.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package word2vec_sample to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/word2vec_sample.zip.
```

```
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package mte_teip5 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/mte_teip5.zip.
```

```
pip install spacy
```

```
Requirement already satisfied: spacy in /usr/local/lib/python3.7/dist-packages (2.2.4)
Requirement already satisfied: blis<0.5.0,>=0.4.0 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: thinc==7.4.0 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: plac<1.2.0,>=0.9.6 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: importlib-metadata>=0.20 in /usr/local/lib/python3.7/dist
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/python3.7/dist
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (f
```

```
!pip install mosestokenizer
```

```
Collecting mosestokenizer
  Downloading mosestokenizer-1.1.0.tar.gz (37 kB)
Requirement already satisfied: docopt in /usr/local/lib/python3.7/dist-packages (from mo
Collecting openfile
  Downloading openfile-0.0.7-py3-none-any.whl (2.4 kB)
Collecting uctools
  Downloading uctools-1.3.0.tar.gz (4.6 kB)
Collecting toolwrapper
  Downloading toolwrapper-2.1.0.tar.gz (3.2 kB)
Building wheels for collected packages: mosestokenizer, toolwrapper, uctools
  Building wheel for mosestokenizer (setup.py) ... done
  Created wheel for mosestokenizer: filename=mosestokenizer-1.1.0-py3-none-any.whl size=
  Stored in directory: /root/.cache/pip/wheels/a7/31/94/fef279382208e85a65c1a7f5c4d00201
  Building wheel for toolwrapper (setup.py) ... done
  Created wheel for toolwrapper: filename=toolwrapper-2.1.0-py3-none-any.whl size=3354 s
  Stored in directory: /root/.cache/pip/wheels/c5/4f/33/54741ffe08e38ececb1d28068a153729
  Building wheel for uctools (setup.py) ... done
  Created wheel for uctools: filename=uctools-1.3.0-py3-none-any.whl size=6163 sha256=99
  Stored in directory: /root/.cache/pip/wheels/fb/44/e9/914cf8fa71f0141f9314f862538d1218
Successfully built mosestokenizer toolwrapper uctools
Installing collected packages: uctools, toolwrapper, openfile, mosestokenizer
Successfully installed mosestokenizer-1.1.0 openfile-0.0.7 toolwrapper-2.1.0 uctools-1.3
```

1. Regular Expression tokenizer

A RegexpTokenizer splits a string into substrings using a regular expression. The following tokenizer forms tokens out of alphabetic sequences, money expressions, and any other non-whitespace sequences

```
#Regular Expression tokenizer

def regular_exp_token(txt):
  from nltk.tokenize import RegexpTokenizer
  re_tk = RegexpTokenizer('\w+|\$[\d\.]+|\S+')
  print("Regular Expression tokenization:")
  print("================================")
  print("\t",re_tk.tokenize(txt))
```

1.1. White space tokenizer

The whitespace tokenizer breaks text into terms whenever it encounters a whitespace character.

```
# White space tokenizer
def white_space_token(txt):
  from nltk.tokenize import WhitespaceTokenizer
  ws_tk = WhitespaceTokenizer()
  print("Whitespace tokenization:")
  print("========================")
  print("\t",ws_tk.tokenize(txt))
```

1.2. Dictionary Based tokenizer

```
# Dictionary Based tokenizer

def dictionary_based_token(txt,dic_list):
  from nltk.tokenize import regexp_tokenize
  newdict = {}
  for item in dic_list:
      dk = item.replace(" ", "_")
      newdict[item] = dk

  for key, val in newdict.items():
      if key in txt:
          txt = txt.replace(key, val)

  print("Dictionary based tokenization:")
  print("==============================")
  print("\t",regexp_tokenize(txt, pattern='\S+'))
```

1.3.Word Punctuation Tokenizer

Tokenizing and removing all punctuation marks from a sentence removes all punctuation marks from each word. Another alternative word tokenizer is WordPunctTokenizer. It splits all punctuation into separate tokens

```
# Word Punctuation Tokenizer

def word_punctuation_token(txt):
  from nltk.tokenize import WordPunctTokenizer
  wp_tk = WordPunctTokenizer()
  print("Word Punctuation tokenization:")
  print("============================")
  print("\t",wp_tk.tokenize(txt))
```

2. Treebank Word / Default tokenizer

The default tokenization method in NLTK involves tokenization using regular expressions as defined in the Penn Treebank (based on English text). It assumes that the text is already split into sentences.

```
# Tree bank Word Tokenizer

def treebank_word_token(txt):
  from nltk.tokenize import TreebankWordTokenizer
  tb_tk = TreebankWordTokenizer()
  print("Treebank word tokenization:")
  print("=========================")
  print("\t",tb_tk.tokenize(txt))
```

3. Tweet tokenizer

When we want to apply tokenization in text data like tweets, the tokenizers mentioned above can't produce practical tokens. Through this issue, NLTK has a rule based tokenizer special for tweets. We can split emojis into different words if we need them for tasks like sentiment analysis.

```
# tweet tokenizer

def tweet_token(txt):
  from nltk.tokenize import TweetTokenizer
  tw_tk = TweetTokenizer()
  print("Tweet tokenization:")
  print("==================")
  print("\t",tw_tk.tokenize(txt))
```

4. MWE Tokenizer

The multi-word expression tokenizer is a rule-based, “add-on” tokenizer offered by NLTK. Once the text has been tokenized by a tokenizer of choice, some tokens can be re-grouped into multi-word expressions.

```
# MWE Tokenizer

def MWE_token(txt):
  from nltk.tokenize import MWETokenizer
  from nltk.tokenize import word_tokenize
  mwe_tk = MWETokenizer()
  print("Normal tokenization:")
  print("====================")
  print("\t",mwe_tk.tokenize(word_tokenize(txt)))

  print("\nAdd MWE")
  print("=======")
  mwe_tk = MWETokenizer()
  k1 = input("\tEnter the first word:")
  k2 = input("\tEnter the second word:")
  mwe_tk.add_mwe((k1,k2))
  print("\n\t",mwe_tk.tokenize(word_tokenize(txt)))
```

5. Spacy Tokenizer

In Spacy, the process of tokenizing a text into segments of words and punctuation is done in various steps. It processes the text from left to right. First, the tokenizer split the text on whitespace similar to the split() function. Then the tokenizer checks whether the substring matches the tokenizer exception rules, so on.

```
# Spacy Tokenizer

def spacy_token(txt):
  import spacy
  sp = spacy.load('en_core_web_sm')
  sentence = sp(txt)
  print("Spacy tokenization:")
  print("===================")
  for word in sentence:
    print("\t",word.text, word.pos_)
```

6. Word tokenizer

Word tokenization is the process of splitting a large sample of text into words.

```
# word tokenizer

def word_token(txt):
  from nltk.tokenize import word_tokenize
  print("Word tokenization:")
  print("==================")
  print("\t",word_tokenize(txt))
```

6.1.PunktWordTokenizer

An alternative word tokenizer is PunktWordTokenizer. It splits on punctuation, but keeps it with the word instead of creating separate tokens.

```
# punktword tokenizer

def punktword_token(txt):
  from nltk.tokenize import PunktWordTokenizer
  tokenizer = PunktWordTokenizer()
  print("PunktWord tokenization:")
  print("======================")
  print("\t",tokenizer.tokenize(txt))
```

7. Sentence tokenizer

Sentence tokenization is the process of splitting a large sample of text into sentence.

```
# sent tokenizer

def sent_token(txt):
  from nltk.tokenize import sent_tokenize
  print("Sentence tokenization:")
  print("======================")
  print("\t",sent_tokenize(txt))
```

8.Moses tokenizer

It separates punctuation from word

```
# Moses tokenizer
from mosestokenizer import *
def moses_token(txt):
  tokenize = MosesTokenizer('en')
```

```
  print("Moses tokenization:")
  print("==================")
  print("\t",tokenize(txt))
```

MAIN FUCTION

```
def Learner_Tokenizer():
  import matplotlib.pyplot as plt
  import matplotlib.image as mpimg
  print()

  print("TOKENIZATION")
  print("============")
  print("It is the process of tokenizing or splitting a string, text into a list o
  print("\n\n\n")
  txt = input("Enter the text for tokenization:\n")

  print("\n")
  print("Different Types of tokenizers:")
  print("\t\t 1.Regular Expression tokenizer")
  print("\t\t\t 1.1. Whitespace tokenizer")
  print("\t\t\t 1.2. Dictionary based tokenizer")
  print("\t\t\t 1.3. Word Punctuation tokenizer")
  print("\t\t 2.Treebank Word tokenizer/ Default tokenizer")
  print("\t\t 3.Tweet tokenizer")
  print("\t\t 4.MWE tokenizer")
  print("\t\t 5.Spacy tokenizer")
  print("\t\t 6.Word tokenizer")
  print("\t\t 7.Sentence tokenizer")
  print("\t\t 8.Moses tokenizer")

  op = 'yes'
  while(op == 'yes'):
    print()
    c = input("Enter your option number:")
    print()
    if c == '1':
      regular_exp_token(txt)

    elif c == '1.1':
      white_space_token(txt)

    elif c == '1.2':
      dic_list = []
      print("Dictionary words: ")
      wrd = input("Enter the word:")
      dic_list.append(wrd)
      dictionary_based_token(txt,dic_list)

    elif c == '1.3':
```

```
    elif c == '1.3':
      word_punctuation_token(txt)

    elif c == '2':
      treebank_word_token(txt)

    elif c == '3':
      tweet_token(txt)

    elif c == '4':
      MWE_token(txt)

    elif c == '5':
      spacy_token(txt)

    elif c == '6':
      word_token(txt)

    elif c == '7':
      sent_token(txt)

    elif c == '8':
      moses_token(txt)

    else:
      print("Please select your option properly")

    print()
    op = input("Do you want to continue learnig [yes/no] :")

  print("Thank You ")
```

Text: A love♡ of learning has a lot to do with learning that we are loved! ~ Fred Rogers. love of learning📚 is the most necessary #passion ... in it lies our 💯happiness. It's a sure remedy for what ails us, an unending source of pleasure. Learning is not money $1000000.

```
Learner_Tokenizer()
```

```
    TOKENIZATION
    ============
    It is the process of tokenizing or splitting a string, text into a list of tokens. On




    Enter the text for tokenization:
    A love♡ of learning has a lot to do with learning that we are loved! ~ Fred Rogers.


    Different Types of tokenizers:
```

```
                    1.Regular Expression tokenizer
                            1.1. Whitespace tokenizer
                            1.2. Dictionary based tokenizer
                            1.3. Word Punctuation tokenizer
                    2.Treebank Word tokenizer/ Default tokenizer
                    3.Tweet tokenizer
                    4.MWE tokenizer
                    5.Spacy tokenizer
                    6.Word tokenizer
                    7.Sentence tokenizer
                    8.Moses tokenizer

Enter your option number:1

Regular Expression tokenization:
================================
        ['A', 'love', '♡', 'of', 'learning', 'has', 'a', 'lot', 'to', 'do', 'with',

Do you want to continue learnig [yes/no] :yes

Enter your option number:1.1

Whitespace tokenization:
========================
        ['A', 'love♡', 'of', 'learning', 'has', 'a', 'lot', 'to', 'do', 'with', 'le

Do you want to continue learnig [yes/no] :yes

Enter your option number:1.2

Dictionary words:
Enter the word:Fred Rogers
Dictionary based tokenization:
==============================
        ['A', 'love♡', 'of', 'learning', 'has', 'a', 'lot', 'to', 'do', 'with', 'le

Do you want to continue learnig [yes/no] :yes

Enter your option number:3

Tweet tokenization:
===================
        ['A', 'love', '♥', '', 'of', 'learning', 'has', 'a', 'lot', 'to', 'do', 'wi

Do you want to continue learnig [yes/no] :no
```