Experiment No. 1

Title: Implementation of Tokenization with different python libraries

Batch: 1 **Roll No.: 16010420008** **Experiment No.:1**

Aim: To implement Tokenization using different python libraries

---

Resources needed: Text Editor, Python Interpreter

Theory:

Tokenization

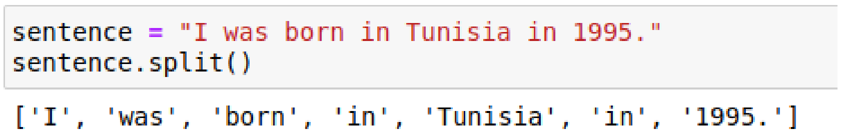
Tokenization is the first step in any NLP pipeline. It has an important effect on the rest of pipeline. Tokenization can separate sentences, words, characters, or subwords._When the text is split into words it is called as „word tokenization" and when the text is split into sentences it is called as „sentence tokenization". A tokenizer breaks unstructured data and natural language text into chunks of information that can be considered as discrete elements.

Tokenization using different python libraries

1. split ( ) function in python

The easiest way is to use split ( ) function for tokenization. By default it uses whitespace as delimiter for tokenization. The default delimiter can be changed as well.

Example:



```
sentence = "I was born in Tunisia in 1995."
sentence.split()
```

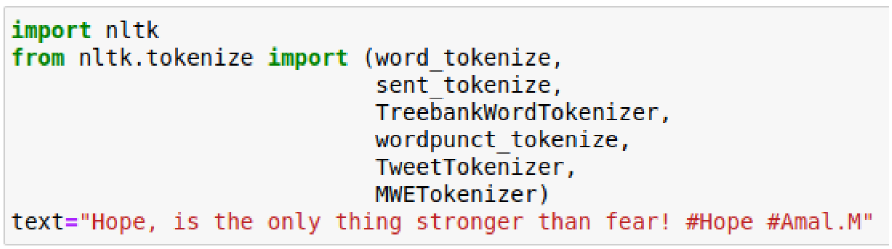
```
['I', 'was', 'born', 'in', 'Tunisia', 'in', '1995.']
```

2. NLTK (Natural Language Toolkit) library

NLTK is an open-source Python library for Natural Language Processing. It has easy-to-use interfaces for over 50 corpora and lexical resources such as WordNet, along with a set of text processing libraries for tokenization, stop word removal, stemming, lemmatization, classification and so on.

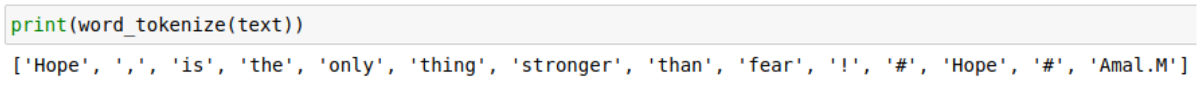
Example:

Install NLTK – pip install nltk

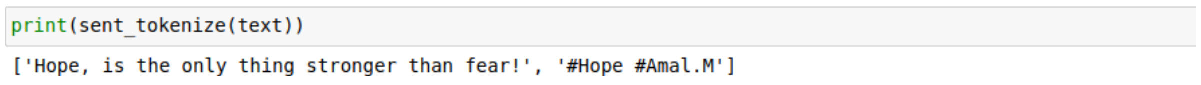


```
import nltk
from nltk.tokenize import (word_tokenize,
                           sent_tokenize,
                           TreebankWordTokenizer,
                           wordpunct_tokenize,
                           TweetTokenizer,
                           MWETokenizer)
text="Hope, is the only thing stronger than fear! #Hope #Amal.M"
```

- Word and Sentence tokenizer



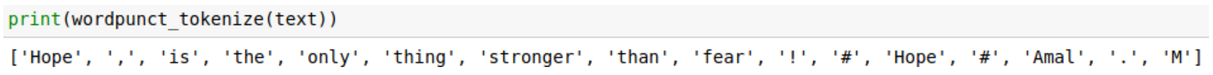
```
print(word_tokenize(text))
['Hope', ',', 'is', 'the', 'only', 'thing', 'stronger', 'than', 'fear', '!', '#', 'Hope', '#', 'Amal.M']
```



```
print(sent_tokenize(text))
['Hope, is the only thing stronger than fear!', '#Hope #Amal.M']
```

- Punctuation-based tokenizer

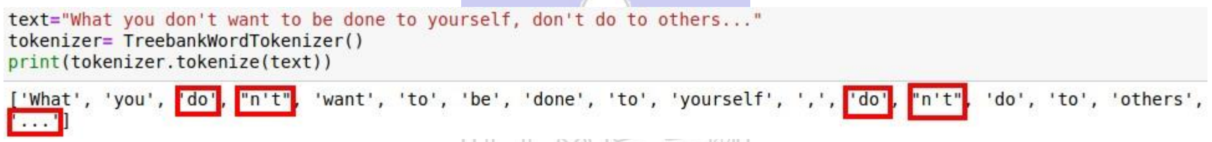
This tokenizer splits the sentences into words based on whitespaces and punctuations.



```
print(wordpunct_tokenize(text))
['Hope', ',', 'is', 'the', 'only', 'thing', 'stronger', 'than', 'fear', '!', '#', 'Hope', '#', 'Amal', '.', 'M']
```

- Treebank Word tokenizer

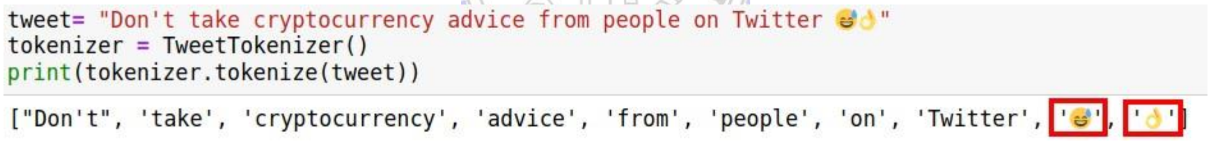
This tokenizer incorporates a variety of common rules for english word tokenization. It separates phrase-terminating punctuation like (?!.;,) from adjacent tokens and retains decimal numbers as a single token. Besides, it contains rules for English contractions such as "don"t" is tokenized as ["do", "n"t"].



```
text="What you don't want to be done to yourself, don't do to others..."
tokenizer= TreebankWordTokenizer()
print(tokenizer.tokenize(text))
['What', 'you', 'do', "n't", 'want', 'to', 'be', 'done', 'to', 'yourself', ',', 'do', "n't", 'do', 'to', 'others',
'...']
```

- Tweet tokenizer

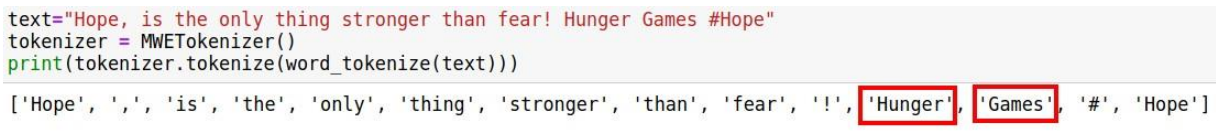
NLTK has a rule based tokenizer special for tweets. It can split emojis into different words if needed for tasks like sentiment analysis.



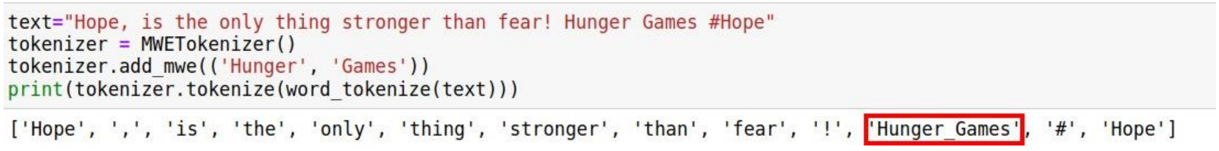
```
tweet= "Don't take cryptocurrency advice from people on Twitter 😅👌"
tokenizer = TweetTokenizer()
print(tokenizer.tokenize(tweet))
["Don't", 'take', 'cryptocurrency', 'advice', 'from', 'people', 'on', 'Twitter', '😅', '👌']
```

- MWET tokenizer

NLTK"s multi-word expression tokenizer (MWETokenizer) provides a function add_mwe() that allows the user to enter multiple word expressions before using the tokenizer on the text. More simply, it can merge multi-word expressions into single tokens.



```
text="Hope, is the only thing stronger than fear! Hunger Games #Hope"
tokenizer = MWETokenizer()
print(tokenizer.tokenize(word_tokenize(text)))
['Hope', ',', 'is', 'the', 'only', 'thing', 'stronger', 'than', 'fear', '!', 'Hunger', 'Games', '#', 'Hope']
```



```
text="Hope, is the only thing stronger than fear! Hunger Games #Hope"
tokenizer = MWETokenizer()
tokenizer.add_mwe(('Hunger', 'Games'))
print(tokenizer.tokenize(word_tokenize(text)))
['Hope', ',', 'is', 'the', 'only', 'thing', 'stronger', 'than', 'fear', '!', 'Hunger_Games', '#', 'Hope']
```

More NLTK tokenization modules can be studied from https://www.nltk.org/api/nltk.tokenize.html

3. TextBlob library

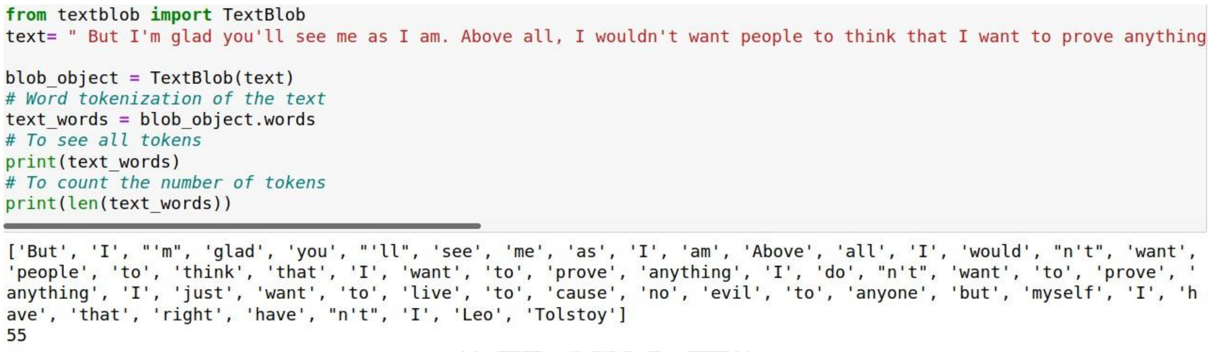
TextBlob is a Python library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and so on.

Example:

Install TextBlob – pip install textblob

from textblob import TextBlob

text=" But I'm glad you'll see me as I am. Above all, I wouldn't want people to think that I want to prove anything. I don't want to prove anything, I just want to live; to cause no evil to anyone but myself. I have that right, haven't I Leo Tolstoy"



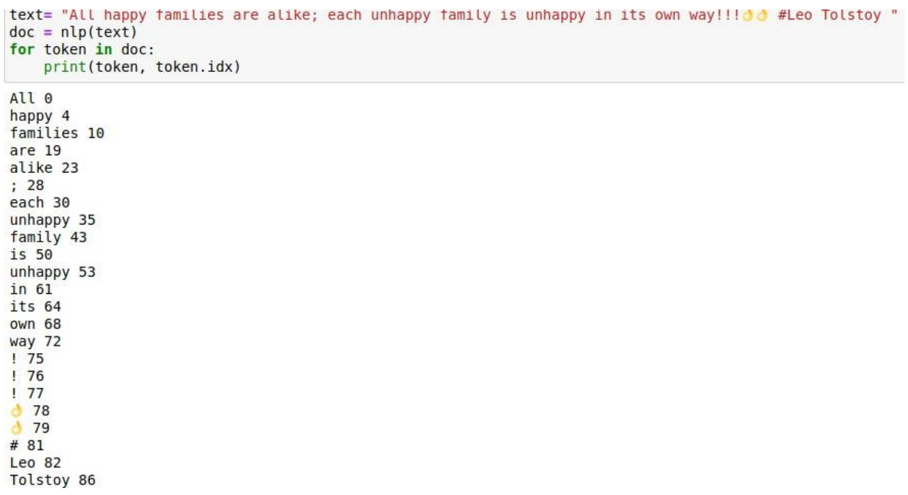
```
from textblob import TextBlob
text= " But I'm glad you'll see me as I am. Above all, I wouldn't want people to think that I want to prove anything

blob_object = TextBlob(text)
# Word tokenization of the text
text_words = blob_object.words
# To see all tokens
print(text_words)
# To count the number of tokens
print(len(text_words))
```

```
['But', 'I', "'m", 'glad', 'you', "'ll", 'see', 'me', 'as', 'I', 'am', 'Above', 'all', 'I', 'would', "n't", 'want',
'people', 'to', 'think', 'that', 'I', 'want', 'to', 'prove', 'anything', 'I', 'do', "n't", 'want', 'to', 'prove', '
anything', 'I', 'just', 'want', 'to', 'live', 'to', 'cause', 'no', 'evil', 'to', 'anyone', 'but', 'myself', 'I', 'h
ave', 'that', 'right', 'have', "n't", 'I', 'Leo', 'Tolstoy']
55
```

4. spaCy library

SpaCy is an open-source Python library that parses and understands large volumes of text. spaCy tokenizer provides the flexibility to specify special tokens that don‟t need to be segmented, or need to be segmented using special rules for each language, for example punctuation at the end of a sentence should be split off – whereas "U.K." should remain one token.

Example: Install SpaCy – pip install spacy

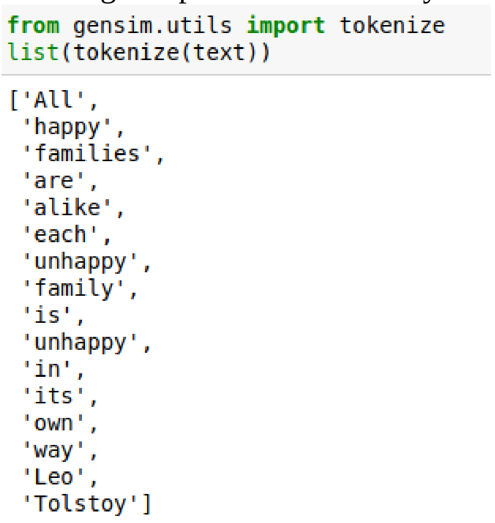


```
text= "All happy families are alike; each unhappy family is unhappy in its own way!!!👌👌 #Leo Tolstoy "
doc = nlp(text)
for token in doc:
    print(token, token.idx)
```

```
All 0
happy 4
families 10
are 19
alike 23
; 28
each 30
unhappy 35
family 43
is 50
unhappy 53
in 61
its 64
own 68
way 72
! 75
! 76
! 77
👌 78
👌 79
# 81
Leo 82
Tolstoy 86
```

5. Gensim library

Gensim is a Python library for topic modeling, document indexing, and similarity retrieval with large corpora. It offers utility functions for tokenization.



```
from gensim.utils import tokenize
list(tokenize(text))
```

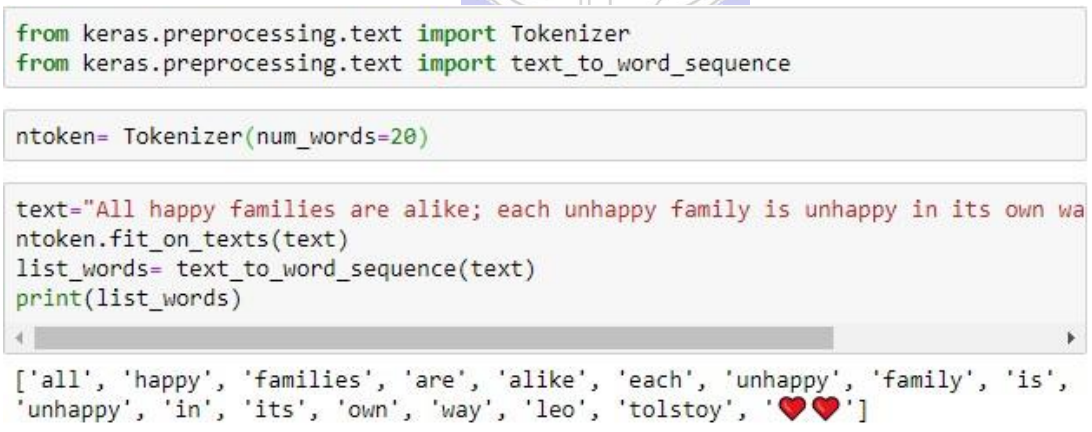
```
['All',
 'happy',
 'families',
 'are',
 'alike',
 'each',
 'unhappy',
 'family',
 'is',
 'unhappy',
 'in',
 'its',
 'own',
 'way',
 'Leo',
 'Tolstoy']
```

6. Keras library

Keras is an open-source library in python and is one of the most reliable deep learning frameworks. To perform tokenization, text_to_word_sequence method from the Class Keras.preprocessing.text class is used. The great thing about Keras is converting the alphabet in a lower case before tokenizing it, which saves time.

Example:

Install Keras – pip install keras



```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.text import text_to_word_sequence
```

```
ntoken= Tokenizer(num_words=20)
```

```
text="All happy families are alike; each unhappy family is unhappy in its own wa
ntoken.fit_on_texts(text)
list_words= text_to_word_sequence(text)
print(list_words)
```

```
['all', 'happy', 'families', 'are', 'alike', 'each', 'unhappy', 'family', 'is',
'unhappy', 'in', 'its', 'own', 'way', 'leo', 'tolstoy', '❤❤']
```

Activity:

1. Apply tokenization using different python libraries with different text inputs such as tweets, news article and so on.
2. Write your observances regarding using different methods of tokenization with different input text data.

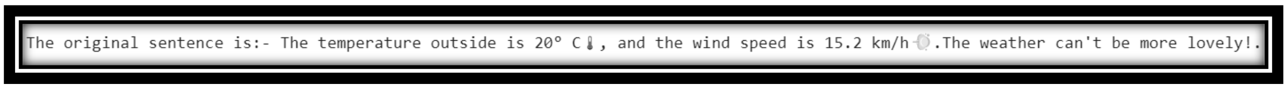
Results: (Program with snapshot of output)

Code:

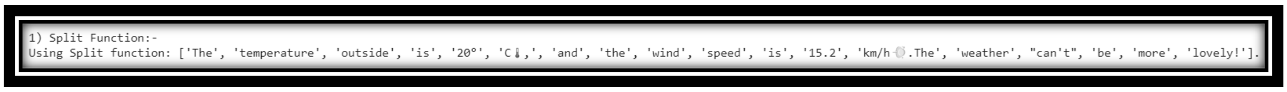
```
import nltk
nltk.download('punkt')
import nltk.tokenize as nltktoken
from textblob import TextBlob as textblob
import spacy
from gensim.utils import tokenize
from keras.preprocessing.text import text_to_word_sequence

unique_sentence = f"The temperature outside is 20{chr(176)} C 🌡, and the wind
speed is 15.2 km/h 💨.The weather can't be more lovely!"
print(f"The original sentence is:- {unique_sentence}.")

print(f"1) Split Function:- ")
split_unique_sentence = unique_sentence.split()
print(f"Using Split function: {split_unique_sentence}.")

print(f"2) Natural Language Toolkit (NLTK) Library:- ")
word_unique_sentence = nltktoken.word_tokenize(unique_sentence)
print(f"Using Word Tokenize Function: {word_unique_sentence}.")

sentence_unique_sentence = nltktoken.sent_tokenize(unique_sentence)
print(f"Using Sentence Tokenize Function: {sentence_unique_sentence}.")

tweet_unique_sentence = nltktoken.TweetTokenizer().tokenize(unique_sentence)
print(f"Using Tweet Tokenize Function: {tweet_unique_sentence}.")

wordpunct_unique_sentence = nltktoken.wordpunct_tokenize(unique_sentence)
print(f"Using Word Punct Function: {wordpunct_unique_sentence}.")

multitoken_unique_sentence = nltktoken.MWETokenizer([('29', '°', 'C', '🌡'),
('15.2', 'km', '/', 'h', '💨')]).tokenize(unique_sentence.split())
print(f"Using Multi Word Expression Function:
```

```
{multitoken_unique_sentence}.")

 print(f"3) Textblob Library:- ")
 textblob_words_unique_sentence = textblob(unique_sentence).words
 print(f"Using Textblob Words Function: {textblob_words_unique_sentence}.")

 print(f"4) Spacy Library:- ")
 spacy_unique_sentence = list(spacy.blank("en")(unique_sentence))
 print(f"Using Spacy Function: {spacy_unique_sentence}")

 print(f"5) Gensim Library:- ")
 gensim_unique_sentence = list(tokenize(unique_sentence))
 print(f"Using Gensim Tokenize Function: {gensim_unique_sentence}")

 print(f"6) Keras Library:- ")
 keras_unique_sentence = text_to_word_sequence(unique_sentence)
 print(f"Using Keras Tokenize Function: {keras_unique_sentence}")
```

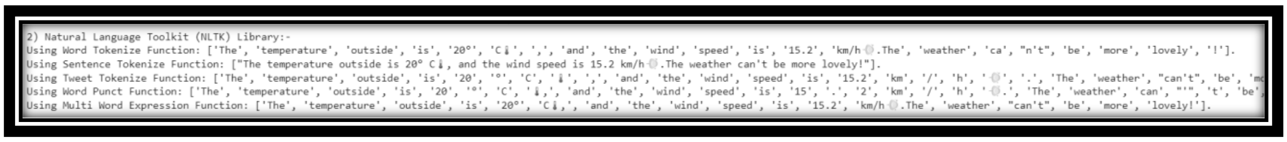
Output:



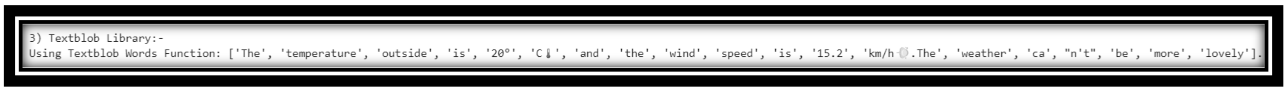
```
The original sentence is:- The temperature outside is 20° C🌡, and the wind speed is 15.2 km/h🌬.The weather can't be more lovely!.
```



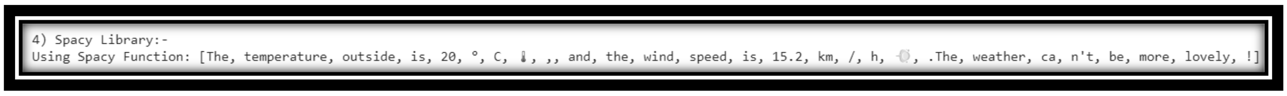
```
1) Split Function:-
Using Split function: ['The', 'temperature', 'outside', 'is', '20°', 'C🌡,', 'and', 'the', 'wind', 'speed', 'is', '15.2', 'km/h🌬.The', 'weather', "can't", 'be', 'more', 'lovely!'].
```



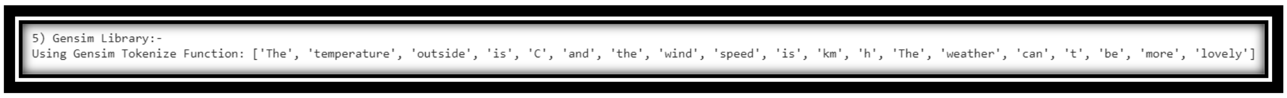
```
2) Natural Language Toolkit (NLTK) Library:-
Using Word Tokenize Function: ['The', 'temperature', 'outside', 'is', '20°', 'C🌡', ',', 'and', 'the', 'wind', 'speed', 'is', '15.2', 'km/h🌬.The', 'weather', 'ca', "n't", 'be', 'more', 'lovely', '!'].
Using Sentence Tokenize Function: ["The temperature outside is 20° C🌡, and the wind speed is 15.2 km/h🌬.The weather can't be more lovely!"].
Using Tweet Tokenize Function: ['The', 'temperature', 'outside', 'is', '20', '°', 'C', '🌡', ',', 'and', 'the', 'wind', 'speed', 'is', '15.2', 'km', '/', 'h', '🌬', '.', 'The', 'weather', "can't", 'be', 'mo
Using Word Punct Function: ['The', 'temperature', 'outside', 'is', '20', '°', 'C', '🌡,', 'and', 'the', 'wind', 'speed', 'is', '15', '.', '2', 'km', '/', 'h', '🌬.', 'The', 'weather', 'can', "'", 't', 'be',
Using Multi Word Expression Function: ['The', 'temperature', 'outside', 'is', '20°', 'C🌡,', 'and', 'the', 'wind', 'speed', 'is', '15.2', 'km/h🌬.The', 'weather', "can't", 'be', 'more', 'lovely!'].
```



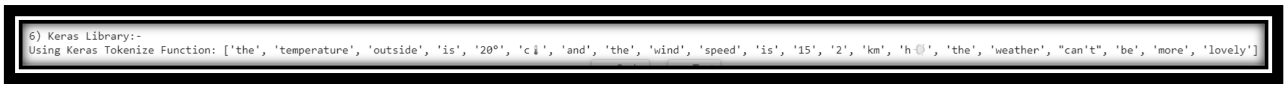
```
3) Textblob Library:-
Using Textblob Words Function: ['The', 'temperature', 'outside', 'is', '20°', 'C🌡', 'and', 'the', 'wind', 'speed', 'is', '15.2', 'km/h🌬.The', 'weather', 'ca', "n't", 'be', 'more', 'lovely'].
```



```
4) Spacy Library:-
Using Spacy Function: [The, temperature, outside, is, 20, °, C, 🌡, ,, and, the, wind, speed, is, 15.2, km, /, h, 🌬, .The, weather, ca, n't, be, more, lovely, !]
```



```
5) Gensim Library:-
Using Gensim Tokenize Function: ['The', 'temperature', 'outside', 'is', 'C', 'and', 'the', 'wind', 'speed', 'is', 'km', 'h', 'The', 'weather', 'can', 't', 'be', 'more', 'lovely']
```



```
6) Keras Library:-
Using Keras Tokenize Function: ['the', 'temperature', 'outside', 'is', '20°', 'c🌡', 'and', 'the', 'wind', 'speed', 'is', '15', '2', 'km', 'h🌬', 'the', 'weather', "can't", 'be', 'more', 'lovely']
```

Observations:

For word_tokenize (NLTK): word_tokenize function typically splits the sentence into words based on spaces and punctuation and doesn't handle emojis as separate tokens unless you preprocess the text to separate them.

For sent_tokenize (NLTK): sent_tokenize divides the text into sentences based on sentence-ending punctuation like periods, question marks, and exclamation marks.

For TweetTokenizer (NLTK): TweetTokenizer is designed to handle Twitter-specific tokens, including usernames, hashtags, URLs, and emoji and may tokenize emojis as separate tokens.

For wordpunct_tokenize (NLTK): wordpunct_tokenize tokenizes text based on whitespace and punctuation and tends to split emojis and special characters into separate tokens.

For MWETokenizer (NLTK): Multi-Word Expression (MWE) Tokenization is used for identifying multi-word expressions and might be useful when you have specific phrases or expressions that you want to treat as single tokens.

TextBlob: TextBlob uses NLTK under the hood and is similar to NLTK's word_tokenize and wordpunct_tokenize functions where emojis may be treated as separate tokens.

spaCy: spaCy is a modern NLP library that can handle, special characters, and punctuation effectively and often treats emojis as separate tokens and provides detailed linguistic information for each token.

gensim: Gensim is primarily used for topic modeling and doesn't provide tokenization as a primary feature.

Keras (with Tokenizer): Keras is a deep learning library, and tokenization is usually performed using the Tokenizer class as it tokenizes text based on spaces and punctuation where emojis are treated as separate tokens.

Questions:

1. Explain the tokenization of regular expression using Regexp tokenizer from NLTK with the help of an example

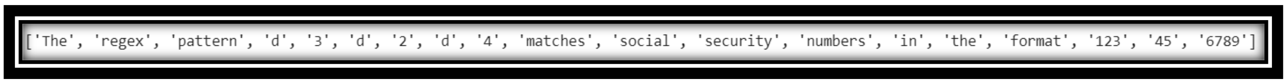
A) Tokenization is the process of breaking down text into smaller units, typically called tokens, which might be words or subwords. The RegexpTokenizer from the Natural Language Toolkit (NLTK) library allows for tokenization based on regular expression patterns.

Working:
There are primarily two modes to use the RegexpTokenizer:
Tokenize based on matching patterns.
Tokenize based on gaps between patterns.
By default, it tokenizes based on matching patterns. If you want to tokenize based on gaps, you should set the gaps flag to True.

Code:

```
import nltk
from nltk.tokenize import RegexpTokenizer
text = "The regex pattern \d{3}-\d{2}-\d{4}, matches social security numbers in the format 123-45-
6789."
tokenizer = RegexpTokenizer(r'\w+')
tokens = tokenizer.tokenize(text)
print(tokens)
```

Output:



```
['The', 'regex', 'pattern', 'd', '3', 'd', '2', 'd', '4', 'matches', 'social', 'security', 'numbers', 'in', 'the', 'format', '123', '45', '6789']
```

Outcomes:

CO1: Understand fundamentals of NLP

Conclusion: (Conclusion to be based on the outcomes achieved)

We understood the concept of tokenization and the various tokenization along with different modules. We implemented them for a sentence with alphabets, numbers, special characters, punctuations, emojis and noted our observations.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Allen.James, Natural Language Understanding, Benjamin Cumming, Second Edition,1995
2. Jurafsky, Dan and Martin, James, Speech and Language Processing, Prentice Hall,2008
3. Palash Goyal, Karan Jain, Sumit Pandey,Deep Learning for Natural Language