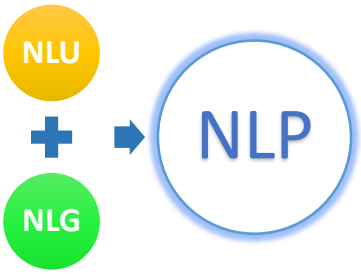
Introduction to NLP

The NLP stands for ***Natural Language Processing***. it is one of the branch of artificial intelligence, that can understand or speak or write the human language.

There are mainly two types of python libraries used for NLP which are ***NLTK*** and ***Spacy.***

1. NLP Branches:

The NLP has two branches are NLU and NLG:



1.1. NLU – (Natural Language Understanding)

- NLU is a computer program / NLP algorithm that can understand human language in the form of voice records or text.
- Here the natural language providing as input to NLU and it will understand the input and give a response in natural language according to our input.

1.2. NLG – (Natural Language Generation)

- NLG is a computer program that can Generate human language in the form of speech or text based on our request.
- NLG takes our requests as input and generates the natural language that can understandable by human.

2. NLP Stages:

NLP divided into four stages. These stages are very important to create accurate model. Let's learn one by one below.

1. Tokenization
2. Data cleaning
3. Vectorization / Word embedding
4. Model Development

3. Tokenization

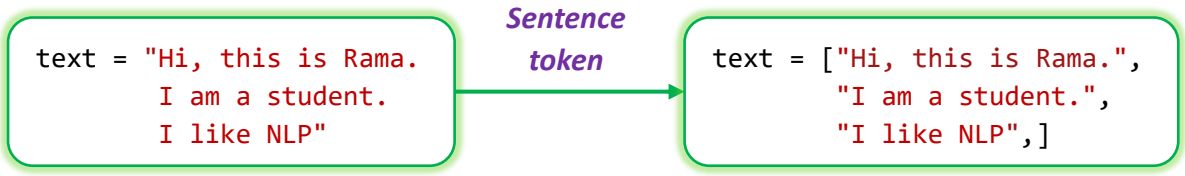
- In NLP Tokenization is the first step. In this stage the whole text will divided into meaningful units are called the token.
- The NLP or ML algorithms can understand the text in the form vector only. So, to convert a text into vectors, first we should generate the tokens of that text and will convert those tokens into vectors. This is the main use of tokenization.

There are three types of tokenization techniques are using based on the requirement:

1. Sentence token
2. Tokens of sentences
3. Tokens of whole text

3.1. Sentence token

- The *sentence token* is a technique that converts each sentence as a token from whole text.
- In this stage we can remove all non-required special characters from each sentence.



By Using NLTK:

```
1. import nltk

2. text = "Hi, this is Rama. I am a student. I like NLP" # Input text

3. # Extract sentences from the processed document
4. sent_tokens = nltk.sent_tokenize(text)
5. print(sent_tokens)

# OUTPUT: ['Hi, this is Rama.', 'I am a student.', 'I like NLP']
```

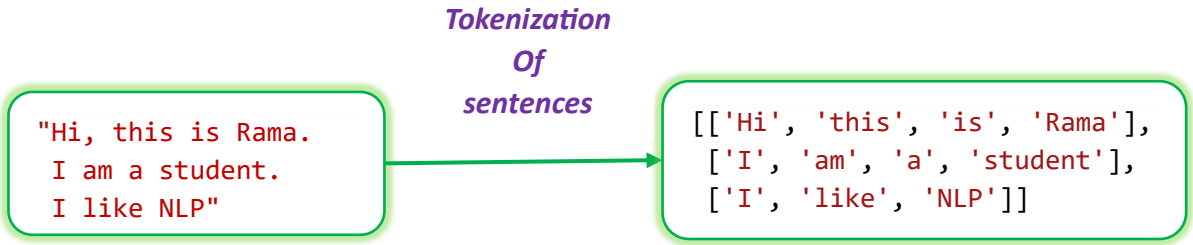
By Using SpaCy:

```
1. import spacy

2. nlp = spacy.load('en_core_web_sm') # Load the English language model
3. doc = nlp(text) # Process the text with spaCy

4. # Extract sentences from the processed document
5. sentences = [sent.text for sent in doc.sents]
```

```
6. print(sentences)

# OUTPUT: ['Hi, this is Rama.', 'I am a student.', 'I like NLP']
```

3.2. Tokens of Sentences

- The *tokens of sentences* is nothing but process converting each sentence into tokens and each sentence tokens will be a list.



By Using NLTK:

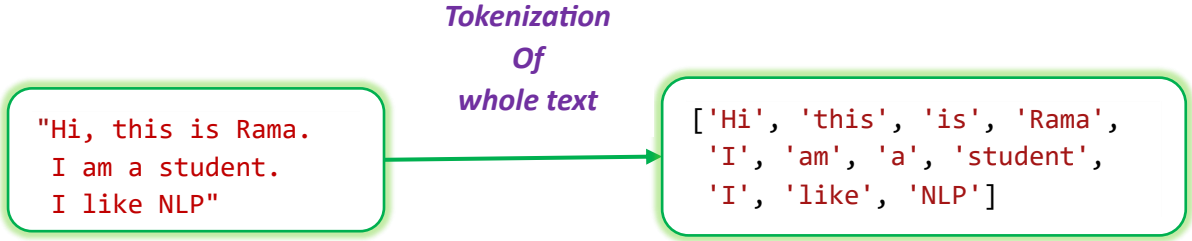
```
1. tokens_of_sents = [list(nltk.word_tokenize(sent)) for sent in
   sent_tokens]
2. print(tokens_of_sents)

# OUTPUT: [['Hi', ',', 'this', 'is', 'Rama', '.'], ['I', 'am', 'a', 'student', '.'], ['I', 'like', 'NLP']]
```

By Using SpaCy:

```
1. import spacy

2. nlp = spacy.load('en_core_web_sm') # Load the English language model
3. doc = nlp(text) # Process the text with spaCy

4. # Extract tokens of sentences from the processed document
5. tokens_of_sents = [[token for token in sent] for sent in doc.sents]
6. print(tokens_of_sents)

# OUTPUT: [[Hi, ,, this, is, Rama, .], [I, am, a, student, .], [I, like, NLP]]
```

3.3. Tokens of Whole Text

- The *tokens of whole text* is nothing but process converting whole text into a list of tokens.



By Using NLTK:

```
tokens_of_text = nltk.word_tokenize(text)
print(tokens_of_text)

# OUTPUT: [Hi, ,, this, is, Rama, ., I, am, a, student, ., I, like, NLP]
```

By Using SpaCy:

```
import spacy

nlp = spacy.load('en_core_web_sm') # Load the English language model
doc = nlp(text) # Process the text with spaCy

# Extract tokens of text from the processed document
tokens_of_text = [token for token in doc]
print(tokens_of_text)

# OUTPUT: [Hi, ,, this, is, Rama, ., I, am, a, student, ., I, like, NLP]
```

4. Data Cleaning

Data cleaning is a very important step to generate affordable text to create vectorization format. Let's start learn all data cleaning topics here.

Steps involved in data cleaning:

1. Remove stop-words
2. Convert to lower case
3. Stemming
4. Lemmatization
5. Removing special and hidden characters.

Consider the below text as example input natural language sentence to perform the data cleaning:

```
text = 'I am in America, Working as a DATA SCIENTIST and reporting @ 10:00 am every day…!'
```

4.1. Removing Stop-Words

- Stop-words are some simple and commonly used words in human languages to form a sentence. ***Ex:*** *am, in, as, a, an, and…etc.*
- These stop-words don't create any sense and don't have any extra value once the text converted into vector format.
- Sometimes there are some words not important to business case then those words are also treated as stop words.

By Using NLTK:

```
from nltk.corpus import stopwords
# Input text
```

```
text = 'I am in America, Working as a DATA SCIENTIST and urgent to reporting @10:00 am every day…!'
useless_wrds = ['urgent', 'may', 'can'] # Use-less words
# Adding useless_wrds to stop words
final_stopwords = stopwords.words('english')
final_stopwords.extend(useless_wrds)

# Removing stop words
final_text = ' '.join([word for word in text.split() if word not in final_stopwords])
print(final_text)

# OUTPUT: I America, Working DATA SCIENTIST reporting @10:00 every day…!
```

By Using SpaCy:

```
import spacy

nlp = spacy.load("en_core_web_sm") # Loading the nlp model
# Creating document from nlp model
doc = nlp('I am in America, Working as a DATA SCIENTIST and urgent to reporting @10:00 am every day…!')

useless_wrds = ['urgent', 'may', 'can'] # Use-less words

# Final text after remove stop words
print(" ".join([token.text for token in doc if (not token.is_stop) and (not token.text in useless_wrds) ]))

# OUTPUT: America , Working DATA SCIENTIST reporting @10:00 day … !
```

4.2. Convert To Lower Case

- The conversion of text to lower case is very important step to maintain the uniformity.
- ***Normalization:*** Lowercasing the text helps normalize the input by reducing the number of unique word variations. For example, "Help," "help," and "HELP" are essentially the same word, and converting them all to lowercase ensures consistency and reduces the vocabulary size
- ***Word Matching:*** Lowercasing text allows for better word matching and comparison. By converting all words to lowercase, you can treat two instances of the same word with different capitalization (e.g., "help" and "Help") as identical, which can be useful for various NLP tasks such as information retrieval, text classification, and entity recognition.
- ***Vocabulary Size:*** In many NLP applications, the size of the vocabulary has a significant impact on computational resources and model complexity. Lowercasing helps reduce the vocabulary size by collapsing words with different capitalization into a single entry.

- ***Generalization:*** Lowercasing text helps with generalization by treating different cases of the same word as equivalent. For example, if a model has learned the lowercase form of a word, it can more easily recognize and generate other variations of the same word.

```
# Input text
text = 'I am in America, Working as a DATA SCIENTIST and urgent to reporting @10:00 am every day…!'

final_text = text.lower()# Converting text into lower case.
print(final_text)

# OUTPUT: i am in america, working as a data scientist and urgent to reporting @10:00 am every day…!
```

4.3. Stemming

- Stemming is the process of cutting suffix part of the word and gives it's root word called "stem".
- The purpose of stemming is to normalize words so that different variations of the same word are treated as one, thereby it reducing the dimensionality of the vocabulary and improving the efficiency of text analysis.
  Ex: For example, consider the words "running," "runs," and "ran." These words all share the same root concept of "run." After applying stemming, they would all be reduced to their common stem, "run".

Some times stemming leads to create incorrect word by blindly remove suffixes. Stemming generates **"runn"** from "**running"** but the correct root word is **"run".** This problem can resolve buy lemmatization concept.

By Using NLTK:

```
from nltk.stem import PorterStemmer

# Input text
text = 'i am in america, working as a data scientist and urgent to reporting @10:00 am every day…!'

post_stemmer = PorterStemmer()# Generate the stemming.
final_text = ' '.join([post_stemmer.stem(word) for word in text.split()])

print(final_text)

# OUTPUT: i am in america, work as a data scientist and urgent to report @10:00 am everi day…!
```

Note: Spacy not supporting the stemming concept. We have only NLTK library to achieve this

4.4. Lemmatization

- Limitation is advanced technique of stemming; it overcomes the root word in character recognition problem.
- It generates the lemma of the word instead of blindly removing the suffix. Lemmatization is the best technique than compared to stemming.

Lemmatization doesn't work properly in spaCy after removing stop words. Lemmatization is the process of reducing words to their base or root form, and it depends on the context of the words in a sentence. Removing stop words may lead to changes in the context, and that can affect lemmatization results.

By Using NLTK:

```
1. from nltk.stem import wordnet, WordNetLemmatizer
2. from nltk.tokenize import word_tokenize

3. # Input text
4. text_data = 'i am in america, works as a data scientist and urgent to
   reporting @10:00 am every day…!'

5. lem = WordNetLemmatizer()# Assigning WordNetLemmatizer() algorithm.
6. tokens = word_tokenize(text_data.lower())# Tokenizing the input text.

7. # Performing lemmatization with lem.lemmatize(word) function.
8. print( " ".join([lem.lemmatize(word) for word in tokens]) )
```

```
# OUTPUT: i am in america , work a a data scientist and urgent to reporting @ 10:00 am every day... !
```

By Using SpaCy:

```
1. import spacy

2. nlp = spacy.load("en_core_web_sm") # Loading the nlp model

3. # Creating document from nlp model
4. stop_words_text = nlp('i am in america, working as a data scientist
   and urgent to reporting @10:00 am every day…!')

5. # Performing lemmatization with lemma_
6. print(" ".join([token.lemma_ for token in stop_words_text]))
```

```
# OUTPUT: I be in america , work as a data scientist and urgent to report @10:00 be every day ... !
```

4.5. Removing Special and Hidden Characters

Sometimes the special characters also included in the text. But special characters are not needed in NLP, so those special/hidden characters should be removed in data cleaning stage.

Special characters: “@”, “#”, “&”, “…!” …. etc.
Special characters: “/n”, “/t”, “/s” …. etc.

```
1. import nltk
2. import re

3. def remove_special_characters(text):
4. words = nltk.word_tokenize(text) # Tokenize the text into words

5. # Define a regular expression pattern to filter out special characters
6. pattern = r"[^a-zA-Z0-9:]"

7. # Use list comprehension to filter words
8. return " ".join([re.sub(pattern, '', word) for word in words])

9. text = 'I am in America, Working as a DATA SCIENTIST and reporting @
   10:00 am every day…!'
10.print( remove_special_characters(text))

# OUTPUT: I am in America  Working as a DATA SCIENTIST and reporting  10:00 am every day
```