TEXT PROCESSING – NLP

- Mohammad Sohail
(21N31A66B4)

WHAT IS NLP ?

- NLP – Natural Language Processing
- It is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language.
- The goal of NLP is to enable computers to understand, analyze and interpret human language.

APPLICATIONS OF NLP

01.

VIRTUAL ASSISTANTS

Siri, Google Alexa

02.

CUSTOMER SUPPORT

Amazon Chatbot

03.

EMAIL FILTERING

Gmail - Spam Box

04.

TEXT PROCESSING

Tokenization, Stemming

05.

SEARCH ENGINES

Google, Bing

06.

TEXT AUTOCORRECT

Smartphone Keyboard

Mohammad Sohail

Text Processing

- Text Processing in NLP is a process of preparing and analyzing textual data to extract meaningful insights that help to build models.
- It’s main goal is to convert raw text into structured format that can be used for various NLP tasks.

Mohammad Sohail

Steps Involved In Text Processing

1. Data Collection
2. Word Tokenizaton
3. Removing Punctuations
4. Stop Words Removal
5. Conversion to Lower Case
6. Stemming
7. Lemmatization
8. TF – IDF

Step1.
Data Collection

- Data can be collected through various methods such as APIs, web scraping, and pre-existing datasets.

- Data Set : Spam.csv
- dataset: https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

CODE

```
import pandas as pd
data=pd.read_csv('spam.csv',encoding="ISO-8859-1")
```

Step2.
Removing Punctuations

- Removal of unnecessary punctuations.

CODE

```
import string
string.punctuation
def removePunctuation(text):
        punctuationfree="".join([i for i in text if i not in string.punctuation])
        return punctuationfree
data['cleaned_msg']=data['v2'].apply(lambda x:removePunctuation(x))
```

Step3.
Word Tokenization

- Splitting text into individual words.

CODE

```
from nltk.tokenize import word_tokenize
def tokenize_words(text):
        words = word_tokenize(text)
        return words
data['tokenized_msg']=data[‘cleaned_msg'].apply(lambda x:tokenize_words(x))
```

Step4.
Stop Word Removal

- Stop words are considered to be of little value in helping algorithms understand the content and meaning of the text.

CODE

```
from nltk.corpus import stopwords
stopwords = nltk.corpus.stopwords.words(‘ english ’)
def remove_stopwords(text):
                    output= [i for i in text if i not in stopwords]
                    return output
data['no_stopwords']= data['tokenized_msg'].apply(lambda x:remove_stopwords(x))
```

Step5.
Lower Case Conversion

- Converting the tokens into lower case letters.

CODE

```
data['lowerCase_msg']=data[‘no_stopwords'].apply(lambda x:x.lower())

data[[‘no_stopwords','lowerCase_msg']].set_index(data['v1'])
```

Step6.
Stemming

- It is the process of converting each word into its root form.

CODE

```
from nltk.stem import PorterStemmer
porter_stemmer = PorterStemmer()
def stemming(text):
        stem_text = [porter_stemmer.stem(word) for word in text]
        return stem_text
data['msg_stemmed']=data['lowerCase_msg'].apply(lambda x: stemming(x))
```

Step7.
Lemmatization

- It is the process of converting each word into its root form.
- It results in more accurate and meaningful reductions than stemming

CODE

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
def lemmatizer(text):
        lemm_text = [wordnet_lemmatizer.lemmatize(word) for word in text]
        return lemm_text
data['msg_lemmatized']=data['msg_stemmed'].apply(lambda x:lemmatizer(x))
```

Final Outcome

Final Data Obtained :

- Tokenized
- Cleaned
- Processed
- Meaningful
- Accurate

Final Data Usage

- The data is further converted into numerical form with the help TF-IDF method.
- The numerical data will be integrated into various models like Chatbots, virtual assistants, sentimental analysis models.

PROGRAM

- Importing DataSet

In [1]:
```
import pandas as pd
data=pd.read_csv('spam.csv',encoding="ISO-8859-1")
data.head()
```

Out[1]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [2]:
```
data.shape
```

Out[2]: (5572, 5)

PROGRAM

- Cleaning Data

Step 1 : Removing Punctuations

In [5]:
```
import string
string.punctuation
```

Out[5]:
```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [6]:
```
def removePunctuation(text):
    punctuationfree="".join([i for i in text if i not in string.punctuation])
    return punctuationfree
data['cleaned_msg']=data['v2'].apply(lambda x:removePunctuation(x))
data.head(3)
```

Out[6]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4	cleaned_msg
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN	Go until jurong point crazy Available only in ...
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN	Ok lar Joking wif u oni
2	spam	Free entry in 2 a wkly comp to win	NaN	NaN	NaN	Free entry in 2 a wkly comp to win FA

PROGRAM

Step 2 : Lowering Text

In [7]:

```
data['lowerCase_msg']=data['cleaned_msg'].apply(lambda x:x.lower())
data[['cleaned_msg','lowerCase_msg']].set_index(data['v1'])
```

Out[7]:

v1	cleaned_msg	lowerCase_msg
ham	Go until jurong point crazy Available only in ...	go until jurong point crazy available only in ...
ham	Ok lar Joking wif u oni	ok lar joking wif u oni
spam	Free entry in 2 a wkly comp to win FA Cup fina...	free entry in 2 a wkly comp to win fa cup fina...
ham	U dun say so early hor U c already then say	u dun say so early hor u c already then say
ham	Nah I dont think he goes to usf he lives aroun...	nah i dont think he goes to usf he lives aroun...

PROGRAM

Step 3 : Tokenization (Word Tokenization)

In [9]:

```
import nltk
from nltk.tokenize import word_tokenize
def tokenize_words(text):
 words = word_tokenize(text)
 return words

data['tokenized_msg']=data['lowerCase_msg'].apply(lambda x:tokenize_words(x))
```

In [11]:

```
data.head(1)
```

Out[11]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4	cleaned_msg	lowerCase_msg	tokenized_msg
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN	Go until jurong point crazy Available only in ...	go until jurong point crazy available only in ...	[go, until, jurong, point, crazy, available, o...

PROGRAM

Step 4 : Removing Stop Words

In [12]:
```
import nltk
from nltk.corpus import stopwords
stopwords = nltk.corpus.stopwords.words('english')
stopwords[0:10]
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

Out[12]:
```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

In [13]:
```
def remove_stopwords(text):
    output= [i for i in text if i not in stopwords]
    return output
data['no_stopwords']= data['tokenized_msg'].apply(lambda x:remove_stopwords(x))
```

In [14]:
```
data.head(1)
```

Out[14]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4	cleaned_msg	lowerCase_msg	tokenized_msg	no_stopwords
0	ham	Go until jurong point, crazy..	NaN	NaN	NaN	Go until jurong point crazy	go until jurong point crazy available only in	[go, until, jurong, point, crazy, available,	[go, jurong, point, crazy, available,

PROGRAM

Step 5 : Stemming

In [15]:
```
from nltk.stem import PorterStemmer
porter_stemmer = PorterStemmer()
def stemming(text):
    stem_text = [porter_stemmer.stem(word) for word in text]
    return stem_text
data['msg_stemmed']=data['no_stopwords'].apply(lambda x: stemming(x))
```

In [16]:
```
data.head(1)
```

Out[16]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4	cleaned_msg	lowerCase_msg	tokenized_msg	no_stopwords	msg
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN	Go until jurong point crazy Available only in ...	go until jurong point crazy available only in ...	[go, until, jurong, point, crazy, available, o...	[go, jurong, point, crazy, available, bugis, n...	[ ] av

PROGRAM

Step 6 : Lemmatization

In [18]:

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

def lemmatizer(text):
    lemm_text = [wordnet_lemmatizer.lemmatize(word) for word in text]
    return lemm_text
data['msg_lemmatized']=data['msg_stemmed'].apply(lambda x:lemmatizer(x))
```

In [19]:

```
data.head(1)
```

Out[19]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4	cleaned_msg	lowerCase_msg	tokenized_msg	no_stopwords	msg
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN	Go until jurong point crazy Available only in ...	go until jurong point crazy available only in ...	[go, until, jurong, point, crazy, available, o...	[go, jurong, point, crazy, available, bugis, n...	[l av

PROGRAM

In [20]:
```
list1 = []
for sublist in data['msg_lemmatized']:
    for value in sublist:
        list1.append(value.split(','))
list1
```

Out[20]:
```
[['go'],
 ['jurong'],
 ['point'],
 ['crazi'],
 ['avail'],
 ['bugi'],
 ['n'],
 ['great'],
 ['world'],
 ['la'],
 ['e'],
 ['buffet'],
 ['cine'],
 ['got'],
```

https://github.com/Sohail7861/TextProcessingNLP.git

THANK YOU

- MOHAMMAD SOHAIL (21N31A66B4)