Part -1 (class work) : Text Pre-processing in NLP.

Basics of Text Data Cleaning.

Instructions and Requirements:

In this Notebook we will evaluate few basic text data cleaning techniques which are modt for any NLP tasks.

This Notebook make uses of "NLTK" and "Regex" Library a lot.

Dataset: "trump_tweets.csv"

This week workshop will have two sections:

To DO:

Do - 1 - Read the code provided, understand there usages and Complete Exercise-1, which is at bottom.

Do - 2 - Based on your implementations Demonstrate the importance of Text pre - processing in NLP (one per group).

Time to Complete- 90 mins.

The first step in any Natural Language Processing task is to pre-process the text dataset. The main goal of this step is to remove noise from the data. The noise in text data can be in different form, so in this section we will look into some common datacleaning task performed before any NLP task.

Terminology Alert!!!

- Document: A distinct unit of text. This could be a sentence, paragraph or an artice.

  Example:

1. doc1==> "How are you?"
2. doc2==> "I go to school."

- Corpus: collection of documents.

Example: corpus=[doc1, doc2]

Read the data.

```
import pandas as pd
import numpy as np

df = pd.read_csv('/content/drive/MyDrive/AI and ML/week 8/trum_tweet_sentiment_analysis.csv')

df.head()
```

	text	Sentiment
0	RT @JohnLeguizamo: #trump not draining swamp b...	0
1	ICYMI: Hackers Rig FM Radio Stations To Play A...	0
2	Trump protests: LGBTQ rally in New York https:...	1
3	"Hi I'm Piers Morgan. David Beckham is awful b...	0
4	RT @GlennFranco68: Tech Firm Suing BuzzFeed fo...	0

```
df_text=df[['text']]

df_text.dropna()
```

	text
0	RT @JohnLeguizamo: #trump not draining swamp b...
1	ICYMI: Hackers Rig FM Radio Stations To Play A...
2	Trump protests: LGBTQ rally in New York https:...
3	"Hi I'm Piers Morgan. David Beckham is awful b...
4	RT @GlennFranco68: Tech Firm Suing BuzzFeed fo...
...	...
1850118	Everytime im like 'How the fuck I follow Melan...
1850119	RT @imgur: The Trump Handshake. https://t.co/R...
1850120	"Greenspan warns Trump's policies risk inflati...
1850121	RT @FasinatingLogic: We must also #INVESTIGATE...
1850122	RT @imgur: The Trump Handshake. https://t.co/R...

1850123 rows × 1 columns

Removing Unwanted Text.

Remove URLS:

In this step we will try to remove URLs.

```
import re
def remove_urls(text):
  """
  This function will try to remove URL present in out dataset and replace it with space using regex library.
  Input Args:
  text: strings of text that may contain URLs.
  Output Args:
  text: URLs replaces with text
  """
  url_pattern = re.compile(r'https?://\S+|www\.\S+')
  return url_pattern.sub(r'', text)
```

```
text = " Click on this link to open facebook https://www.facebook.com/"
text_url = remove_urls(text)
```

```
text_url
```

```
' Click on this link to open facebook '
```

```
text_no_url = df_text["text"].apply(remove_urls)
```

Remove Unwanted Characters.

This may be punctuatuion, numbers, emoji, dates etc.

[It depends on dataset and task we are performing. For example, The dataset we are using is scraped from twitter- Thus we will also try to remove @tag and #mentions from the dataset.]

sample = "Hello @gabe_flomo 👌🏾, still want us to hit that new sushi spot??? LMK when you're free cuz I can't go this or next weekend since I'll be swimming!!! #sushiBros #rawFish #🍣

Remove Emojis:

```
def remove_emoji(string):
  """
  This function will replace the emoji in string with whitespace
```

```
  """
  emoji_pattern = re.compile("["
                          u"\U0001F600-\U0001F64F"  # emoticons
                          u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                          u"\U0001F680-\U0001F6FF"  # transport & map symbols
                          u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                          u"\U00002702-\U000027B0"
                          u"\U000024C2-\U0001F251"
                          "]+", flags=re.UNICODE)
  return emoji_pattern.sub(r' ', string)


test_string = "Hello @siman 👌🏾, still on up for the movie???  #MovieNight #friday #🍱"
no_emoji = remove_emoji(test_string)
no_emoji
```

```
'Hello @siman    still on up for the movie???  #MovieNight #friday # '
```

Remove Everyunwanted characters:

We will try to compile everything into one single function to remove everthings.

```
def removeunwanted_characters(document):
  """
  This function will remove all the unwanted characters from the input dataset.
  Input Args:
  documet: A text data to be cleaned.
  Return:
  A cleaned document.
  """
  # remove user mentions
  document = re.sub("@[A-Za-z0-9_]+"," ", document)
  # remove hashtags
  document = re.sub("#[A-Za-z0-9_]+","", document)
  # remove punctuation
  document = re.sub("[^0-9A-Za-z ]", "" , document)
  #remove emojis
  document = remove_emoji(document)
  # remove double spaces
  document = document.replace('  ',"")
  return document.strip()


# Test:
cleaned_string = removeunwanted_characters(test_string)
cleaned_string
```

```
'Hellostill on up for the movie'
```

```
text_removed_unwanted = df_text["text"].apply(removeunwanted_characters)
```

Tokenizations:

Example:

IN:

"He did not try to navigate after the first bold flight, for the reaction had taken something out of his soul."

OUT:

['He', 'did', 'not', 'try', 'to', 'navigate', 'after', 'the', 'first', 'bold', 'flight', ',', 'for', 'the', 'reaction', 'had', 'taken', 'something', 'out', 'of', 'his', 'soul', '.']

We will be using NLTK library to perform tokenizations.

```
import nltk
nltk.download('punkt_tab')
from nltk import word_tokenize
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
```

```
# Test case:
IN = "He did not try to navigate after the first bold flight, for the reaction had taken something out of his soul."
OUT = word_tokenize(IN)
OUT
```

```
['He',
 'did',
 'not',
 'try',
 'to',
 'navigate',
 'after',
 'the',
 'first',
 'bold',
 'flight',
 ',',
 'for',
 'the',
 'reaction',
 'had',
 'taken',
 'something',
 'out',
 'of',
 'his',
```

```
 'soul',
 '.']
```

Remove Punctutations:

```
from nltk.tokenize import RegexpTokenizer

from nltk.tokenize import RegexpTokenizer

def remove_punct(text):
  """
  This function removes the punctutations present in our text data.
  Input Args:
  text: text data.
  Returns:
  text: cleaned text.
  """
  tokenizer = RegexpTokenizer(r"\w+")
  lst=tokenizer.tokenize(' '.join(text))
  return lst



#Test
text_punctutation = "He did not try to navigate: after the!!!! first bold flight, for,,,,, the reaction!!!!had taken??????? something out of his soul."
text_punc_token = word_tokenize(text_punctutation)
print(text_punctutation)
print("+++++++++++++++++_____________________+++++++++++++++++++++")
print(text_punc_token)
print("_____________________++++++++++++++++++++++++++_________________")
text_clean = remove_punct(text_punc_token)
print(text_clean)
```

```
He did not try to navigate: after the!!!! first bold flight, for,,,,, the reaction!!!!had taken??????? something out of his soul.
+++++++++++++++++_____________________+++++++++++++++++++++
['He', 'did', 'not', 'try', 'to', 'navigate', ':', 'after', 'the', '!', '!', '!', '!', 'first', 'bold', 'flight', ',', 'for', ',', ',', ',', ',', ',', ',', 'the', 'reaction', '!',
_____________________++++++++++++++++++++++++++_________________
['He', 'did', 'not', 'try', 'to', 'navigate', 'after', 'the', 'first', 'bold', 'flight', 'for', 'the', 'reaction', 'had', 'taken', 'something', 'out', 'of', 'his', 'soul']
```

Remove StopWord:

A majority of the words in a given text are connecting parts of a sentence rather than showing subjects, objects or intent. Word like “the” or “and” cab be removed by comparing text to a list of stopword provided by the NLTK library.

We can also define stopwords as required by our task and dataset requirement.

```
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stop_words = set(stopwords.words('english'))
custom_stopwords = ['@', 'RT']
stop_words.update(custom_stopwords)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

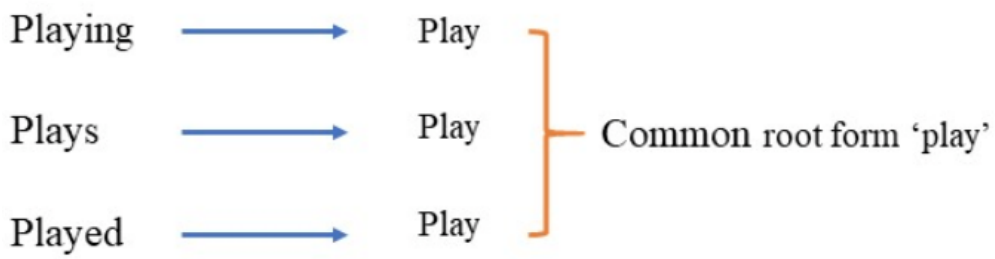
```
def remove_stopwords(text_tokens):
  """
  This function removes all the stopwords present in out text tokens.
  Input Args:
  text_tokens: tokenize input of our datasets.
  Returns:
  result_tokens: list of token without stopword.
  """

  result_tokens = []
  for token in text_tokens:
    if token not in stop_words:
       result_tokens.append(token)
  return result_tokens


test_inputs = ['He', 'did', 'not', 'try', 'to', 'navigate', 'after', 'the', 'first', 'bold', 'flight', ',', 'for', 'the', 'reaction', 'had', 'taken', 'something', 'out', 'of', 'hi
print(test_inputs)
tokens_without_stopwords = remove_stopwords(test_inputs)
print(tokens_without_stopwords)
```

```
['He', 'did', 'not', 'try', 'to', 'navigate', 'after', 'the', 'first', 'bold', 'flight', ',', 'for', 'the', 'reaction', 'had', 'taken', 'something', 'out', 'of', 'his', 'soul'
['He', 'try', 'navigate', 'first', 'bold', 'flight', ',', 'reaction', 'taken', 'something', 'soul', '.']
```

Text Normalization:

This is the idea of reducing number of words present in Corpus by the process of Lemmatization, Stemming, Capital to Lower [i.e. My -- my].

Lemmatization:




```
from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize,pos_tag
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')

def lemmatization(token_text):
  """
  This function performs the lemmatization operations as explained above.
  Input Args:
  token_text: list of tokens.
  Returns:
  lemmatized_tokens: list of lemmatized tokens.
  """
  lemma_tokens = []
  wordnet = WordNetLemmatizer()
  lemmatized_tokens = [wordnet.lemmatize(token, pos = 'v') for token in token_text]

  return lemmatized_tokens
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

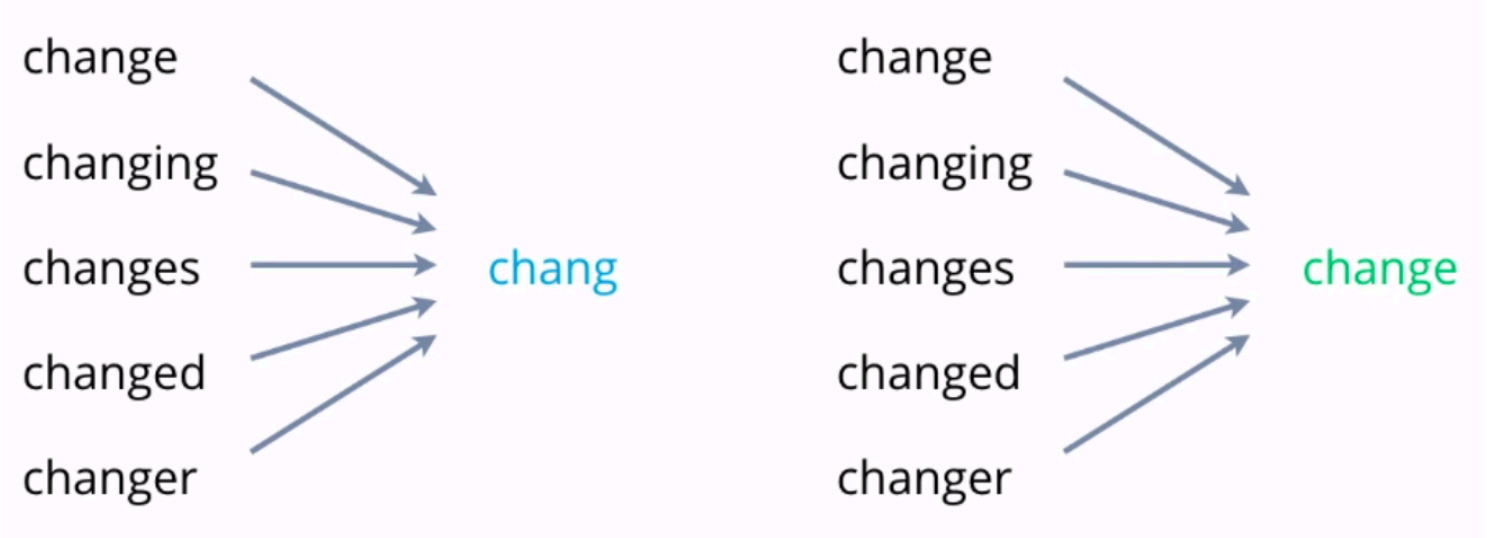
```
lemmatization("Should we go walking or swimming".split())
```

```
['Should', 'we', 'go', 'walk', 'or', 'swim']
```

Stemming:

Also a token(word) reduction techniques. This techniques tries to reduce by chopping off a part of the word at the tail end.

Stemming Vs. Lemmatization.




```
from nltk.stem import PorterStemmer

def stemming(text):
  """
  This function performs stemming operations.
  Input Args:
  token_text: list of tokenize text.
  Returns:
  stemm_tokes: list of stemmed tokens.
  """
  porter = PorterStemmer()
  stemm_tokens = []
  for word in text:
    stemm_tokens.append(porter.stem(word))
  return stemm_tokens


#Test
print("++++++++++++++++++++++++++++++++" "INPUT TOKENS" "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
token_text_test=['Connects','Connecting','Connections','Connected','Connection','Connectings','Connect']
print(token_text_test)
print("++++++++++++++++++" "LEMMATIZED TOKENS" "++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++")
lemma_tokens = lemmatization(token_text_test)
print(lemma_tokens)
print("++++++++++++++++++++" "STEMMED TOKENS" "++++++++++++++++++++++++++++++++++++")
stemmed_tokens = stemming(token_text_test)
```

```
print(stemmed_tokens)
```

```
++++++++++++++++++++++++++++++INPUT TOKENS++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
['Connects', 'Connecting', 'Connections', 'Connected', 'Connection', 'Connectings', 'Connect']
++++++++++++++++++LEMMATIZED TOKENS++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
['Connects', 'Connecting', 'Connections', 'Connected', 'Connection', 'Connectings', 'Connect']
++++++++++++++++++++STEMMED TOKENS++++++++++++++++++++++++++++++++++++
['connect', 'connect', 'connect', 'connect', 'connect', 'connect', 'connect']
```

Lower order:

```
def lower_order(text):
  """
  This function converts all the text in input text to lower order.
  Input Args:
  token_text : input text.
  Returns:
  small_order_text : text converted to small/lower order.
  """
  small_order_text = text.lower()
  return small_order_text

# Test:
sample_text = "This Is some Normalized TEXT"
sample_small = lower_order(sample_text)
print(sample_small)
```

```
this is some normalized text
```

Create Input Text Pipeline

We will compile every basic cleaning steps in following one functions and implement with our datasets.

Exercise-1:

Read the provided data "trump_tweets.csv" and complete the followiing compilin function.

Read data:

```
data = pd.read_csv("/content/drive/MyDrive/AI and ML/week 8/trum_tweet_sentiment_analysis.csv", encoding="ISO-8859-1")
data.head()
```

	text	Sentiment
0	RT @JohnLeguizamo: #trump not draining swamp b...	0
1	ICYMI: Hackers Rig FM Radio Stations To Play A...	0
2	Trump protests: LGBTQ rally in New York https:...	1
3	"Hi I'm Piers Morgan. David Beckham is awful b...	0
4	RT @GlennFranco68: Tech Firm Suing BuzzFeed fo...	0

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
data_cleaning = data["text"].dropna()
```

```
data_cleaning[0]
```

```
'RT @JohnLeguizamo: #trump not draining swamp but own taxpayer dollars on his trips to advertise his properties! @realDonaldTrumpÂ\x85 https://t.co/gFRvUkMXQz'
```

```
def text_cleaning_pipeline(dataset, rule = "lemmatize"):
  """
  This...
  """
  # Convert the input to small/lower order.
  data = lower_order(dataset)

  # Remove URLs
  data =remove_urls(dataset)
  # Remove emojis
  data = remove_emoji(dataset)
  # Remove all other unwanted characters.
  data = removeunwanted_characters(dataset)

  # Create tokens.
  tokens = data.split()
  # Remove stopwords:
  tokens = word_tokenize(data)
  if rule == "lemmatize":
    tokens = lemmatization(tokens)
  elif rule == "stem":
    tokens = stemming(tokens)
  else:
    print("Pick between lemmatize or stem")
```

```
    return " ".join(tokens)


sample = "Hello @gabe_flomo 👌🏾, I still want us to hit that new sushi spot??? LMK when you're free cuz I can't go this or next weekend since I'll be swimming!!! #sushiBros #rawFis
print(text_cleaning_pipeline(sample))
```

```
HelloI still want us to hit that new sushi spot LMK when youre free cuz I cant go this or next weekend since Ill be swim
```

```
test = data["text"][0]


print(text_cleaning_pipeline(test))
```

```
RTnot drain swamp but our taxpayer dollars on his trip to advertise his properties httpstcogFBvUkMX9z
```

```
cleaned_tokens = data["text"].apply(lambda dataset: text_cleaning_pipeline(dataset))


cleaned_tokens[0]
```

```
'RTnot drain swamp but our taxpayer dollars on his trip to advertise his properties httpstcogFBvUkMX9z'
```

```
cleaned_tokens
```

	text
0	RTnot drain swamp but our taxpayer dollars on ...
1	ICYMI Hackers Rig FM Radio Stations To Play An...
2	Trump protest LGBTQ rally in New York httpstco...
3	Hi Im Piers Morgan David Beckham be awful but ...
4	RT Tech Firm Suing BuzzFeed for Publishing Unv...
...	...
1850118	Everytime im like How the fuck I follow Melani...
1850119	RT The Trump Handshake httpstcoRI78itAbC4 http...
1850120	Greenspan warn Trumps policies risk inflation ...
1850121	RT We must also s who vote NOT to allow the re...
1850122	RT The Trump Handshake httpstcoRI78itAbC4 http...

1850123 rows × 1 columns

dtype: object

```
new_df = pd.DataFrame(cleaned_tokens)


df.head(2)
```

	text	Sentiment
0	RT @JohnLeguizamo: #trump not draining swamp b...	0
1	ICYMI: Hackers Rig FM Radio Stations To Play A...	0

```
new_df = pd.concat([new_df, df['Sentiment']], axis=1) # Use pd.concat() to combine the DataFrame and Series
# axis=1 specifies that the concatenation should happen along the columns (horizontally)


new_df.head()
```

	text	Sentiment
0	RTnot drain swamp but our taxpayer dollars on ...	0
1	ICYMI Hackers Rig FM Radio Stations To Play An...	0
2	Trump protest LGBTQ rally in New York httpstco...	1
3	Hi Im Piers Morgan David Beckham be awful but ...	0
4	RT Tech Firm Suing BuzzFeed for Publishing Unv...	0

Train-Test Split

Split the cleaned and tokenized dataset into training and testing sets using train_test_split from sklearn.model_selection

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(new_df.drop('Sentiment', axis=1), new_df['Sentiment'], test_size=0.2, random_state=42)
# Splitting your dataset into features (X) and the target variable (y). new_df.drop('Sentiment', axis=1) contains your features by dropping the 'Sentiment' column,
# and new_df['Sentiment'] contains the target variable. This ensures that train_test_split is correctly splitting the input data into the desired components.



print("new df shape : ", new_df.shape)
print("X_train shape : ", X_train.shape)
print("y_train shape : ", y_train.shape)
print("X_test shape : ", X_test.shape)
print("y_test shape : ", y_test.shape)
```

```
new df shape :  (1850123, 2)
X_train shape :  (1480098, 1)
y_train shape :  (1480098,)
```

```
X_test shape :  (370025, 1)
y_test shape :  (370025,)
```

```
X_train
```

	text
717423	she could develop a line of products that woul...
1035365	RT May hire Daily Mail spokesmanjust days afte...
1338810	Always remember fraudulent Trump University
1024006	Stand up to Trump you chickenshit spineless fu...
379844	Pizza be pizza linda be beautiful Vitor be cam...
...	...
259178	RT Trump Wants To Deport George Soros Do You A...
1414414	RT A pay MaraLago member take photos of Trump ...
131932	RT Both Donald Trump and Tom Brady be marry to...
671155	RT Leaving this here for when have to defend T...
121958	RT When the press say Putins a killer it be ad...

1480098 rows × 1 columns

TF-IDF Vectorization

Import and use the TfidfVectorizer from sklearn.feature_extraction.text to transform the training and testing texts into numerical feature vectors.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizor = TfidfVectorizer()


X_train_vector = vectorizor.fit_transform(X_train['text'])
X_test_vector = vectorizor.transform(X_test['text'])


y_train = y_train.dropna()
y_test= y_test.dropna()
```

```
print("X_train shape : ", X_train_vector.shape)
print("y_train shape : ", y_train.shape)
print("X_test shape : ", X_test_vector.shape)
print("y_test shape : ", y_test.shape)
```

```
X_train shape :  (1480098, 815139)
y_train shape :  (1480098,)
X_test shape :  (370025, 815139)
y_test shape :  (370025,)
```

Model Training

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(max_iter=1000, random_state=42)
```

```
model.fit(X_test_vector, y_test)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

```
y_pred = model.predict(X_train_vector[:1000])
```

```
y_actual = y_train[:1000]
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
```

```
print("accuracy : ", accuracy_score(y_pred, y_actual))
```

```
accuracy :  0.938
```

```
print("Confusion Matrix : ")
print(confusion_matrix(y_pred, y_actual))
```

```
Confusion Matrix :
[[683  43]
 [ 19 255]]
```