

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/AIandML/Tutorial8/trumptweets_small.csv')

df_text=df[['content']]

df_text.dropna()
```

↻

	content	📊
0	Be sure to tune in and watch Donald Trump on L...	📊
1	Donald Trump will be appearing on The View tom...	
2	Donald Trump reads Top Ten Financial Tips on L...	
3	New Blog Post: Celebrity Apprentice Finale and...	
4	"My persona will never be that of a wallflower...	
...	...	
41117	I have never seen the Republican Party as Stro...	
41118	Now Mini Mike Bloomberg is critical of Jack Wi...	
41119	I was thrilled to be back in the Great State o...	
41120	"In the House, the President got less due proc...	
41121	A great show! Check it out tonight at 9pm. @ F...	

41122 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 our dataset and replace it with space using regex library.
    Input Args:
    text: strings of text that may contain URLs.
    Output Args:
    text: URLs replaced with text
    """
    url_pattern = re.compile(r'https?://\S+|www\.\S+')
    return url_pattern.sub(' ', 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["content"].apply(remove_urls)
```

✕ Remove Unwanted Characters.

This may be punctuation, 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:
    document: 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

```

🔍 'Hello still on up for the movie'

```
text_removed_unwanted = df_text["content"].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')
nltk.download('punkt_tab')
from nltk import word_tokenize

```

```

[ nltk_data] Downloading package punkt to /root/nltk_data...
[ nltk_data] Unzipping tokenizers/punkt.zip.
[ 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 Punctuations:

```

from nltk.tokenize import RegexpTokenizer

```

```

from nltk.tokenize import RegexpTokenizer

```

```

def remove_punct(text):
    """
    This function removes the punctuations 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_punctuation = "He did not try to navigate: after the!!!! first bold flight, for,,,,, the reaction!!!!had taken??????? something out o
text_punc_token = word_tokenize(text_punctuation)
print(text_punctuation)
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', ',', ',', ',
+++++++
[ 'He', 'did', 'not', 'try', 'to', 'navigate', 'after', 'the', 'first', 'bold', 'flight', 'for', 'the', 'reaction', 'had', 'taken', 'som

```

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 stopwords 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 our text tokens.
    Input Args:
    text_tokens: tokenize input of our datasets.
    Returns:
    result_tokens: list of token without stopwords.
    """

    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']
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',
['He', 'try', 'navigate', 'first', 'bold', 'flight', ',', 'reaction', 'taken', 'something', 'soul', '.']
```

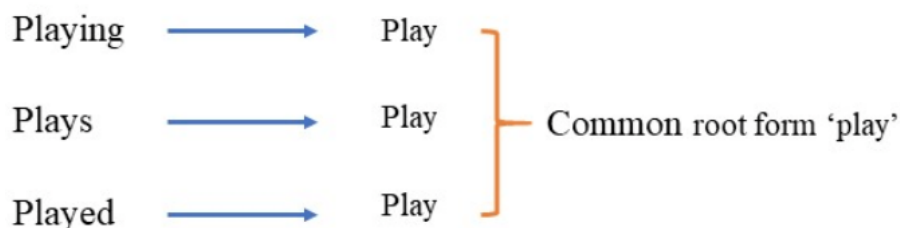
✓ Text Normalization:

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:

It is a common NLP technique used to reduce number of tokens(words) in dataset, this is achieved by replacing the word with its root words.

Example:



```
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...

```

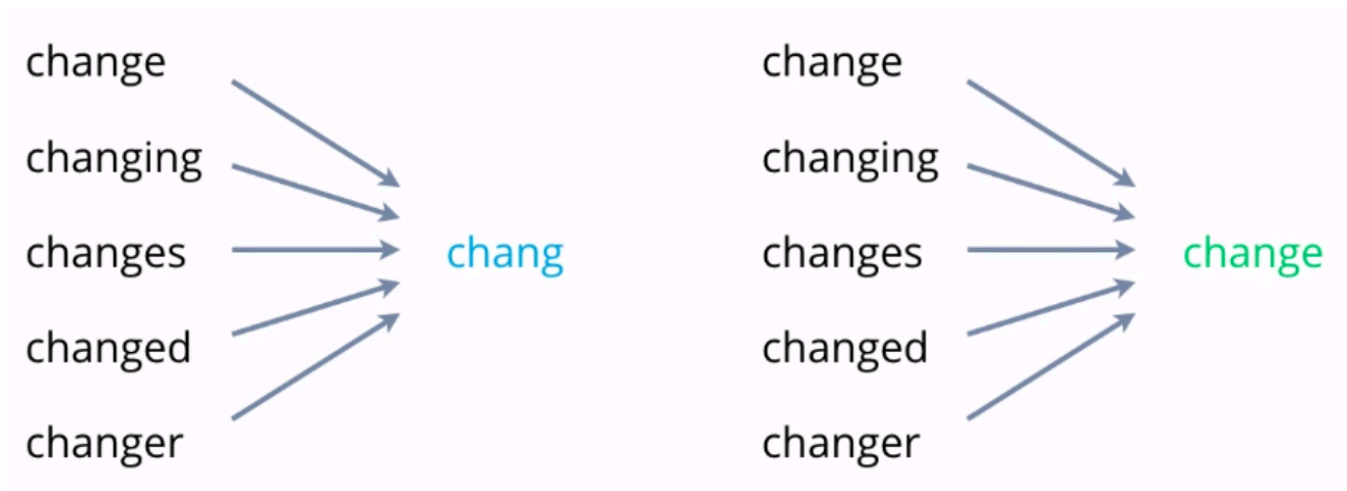
```
lemmatization("Should we go walking or swimming".split())
```

```
['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 following compilin function.

Read data:

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

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
data = pd.read_csv("/content/drive/MyDrive/AIandML/Tutorial8/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	

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

```
data_cleaning[0]
```

```
'RT @JohnLeguizamo: #trump not draining swamp but our taxpayer dollars on his trips to advertise his properties! @realDonaldTrump'
https://t.co/aFRvILkMYQz'
```

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	

```
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 = remove_unwanted_characters(dataset)
```

```
    # Create tokens.
```

```
    tokens = data.split()
```

```
    # Remove stopwords:
```

```
    tokens = remove_stopwords(tokens)
```

```
    # Stemming or Lemmatization:
```

```
    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 sinc
print(text_cleaning_pipeline(sample))
```

```
➦ HelloI still want us hit new sushi spot LMK youre free cuz I cant go next weekend since Ill swim
```

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

```
print(test)
```

```
➦ RT @JohnLeguizamo: #trump not draining swamp but our taxpayer dollars on his trips to advertise his properties! @realDonaldTrump http
```

```
print(text_cleaning_pipeline(test))
```

```
➦ RTnot drain swamp taxpayer dollars trip advertise properties httpstcogFBvUkMX9z
```


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

Double-click (or enter) to edit

```
cleaned_tokens[0]
```

```
➦ 'RTnot drain swamp taxpayer dollars trip advertise properties httpstcogFBvUkMX9z'
```

```
cleaned_tokens
```

	text
0	RTnot drain swamp taxpayer dollars trip advert...
1	ICYMI Hackers Rig FM Radio Stations To Play An...
2	Trump protest LGBTQ rally New York httpstcoLfH...
3	Hi Im Piers Morgan David Beckham awful Donald ...
4	Tech Firm Suing BuzzFeed Publishing Unverified...
...	...
1850118	Everytime im like How fuck I follow Melania Tr...
1850119	The Trump Handshake httpstcoRI78itAbC4 httpstc...
1850120	Greenspan warn Trumps policies risk inflation ...
1850121	We must also vote NOT allow release trump taxe...
1850122	The Trump Handshake httpstcoRI78itAbC4 httpstc...


1850123 rows × 1 columns

dataset object

✓ Creating new dataframe


```
new_df = pd.DataFrame(cleaned_tokens)
```

```
new_df.head(2)
```



	text
0	RTnot drain swamp taxpayer dollars trip advert...
1	ICYMI Hackers Rig FM Radio Stations To Play An...


```
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

```
new_dataset = pd.concat([new_df, data['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_dataset.head()
```



	text	Sentiment
0	RTnot drain swamp taxpayer dollars trip advert...	0
1	ICYMI Hackers Rig FM Radio Stations To Play An...	0
2	Trump protest LGBTQ rally New York httpstcoLfH...	1
3	Hi Im Piers Morgan David Beckham awful Donald ...	0
4	Tech Firm Suing BuzzFeed Publishing Unverified...	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

# Assuming 'new_dataset' is your DataFrame and 'Sentiment' is your target column
X = new_dataset.drop('Sentiment', axis=1) # Features
y = new_dataset['Sentiment'] # Target variable


# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42) # Adjust test_size as needed




# Now you have X_train, X_test, y_train, and y_test
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("y_train shape:", y_train.shape)
print("y_test shape:", y_test.shape)

```

 X_train shape: (1480098, 1)
X_test shape: (370025, 1)
y_train shape: (1480098,)
y_test shape: (370025,)

X_train



	text	
717423	could develop line products would appeal Trump...	
1035365	May hire Daily Mail spokesmanjust days Wikiped...	
1338810	Always remember fraudulent Trump University	
1024006	Stand Trump chickenshit spineless fuckheads ht...	
379844	Pizza pizza linda beautiful Vitor camren fuck ...	
...	...	
259178	Trump Wants To Deport George Soros Do You Agre...	
1414414	A pay MaraLago member take photos Trump brief ...	
131932	Both Donald Trump Tom Brady marry foreign wome...	
671155	Leaving defend Trump knock McCain httpstcoiSQt...	
121958	When press say Putins killer adorable see Trum...	

1480098 rows × 1 columns

y_train



	Sentiment
717423	1
1035365	0
1338810	0
1024006	0
379844	0
...	...
259178	1
1414414	0
131932	1
671155	1
121958	1

1480098 rows × 1 columns

dtype: int64

✓ 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

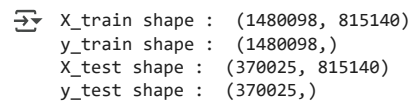
# Initialize TfidfVectorizer
vectorizer = TfidfVectorizer()

# Fit and transform the training data
X_train_tfidf = vectorizer.fit_transform(X_train['text'])

# Transform the testing data (using the vocabulary learned from the training data)
X_test_tfidf = vectorizer.transform(X_test['text'])

Y_test = y_test.dropna()
Y_train = y_train.dropna()

print("X_train shape : ", X_train_tfidf.shape)
print("y_train shape : ", y_train.shape)
print("X_test shape : ", X_test_tfidf.shape)
print("y_test shape : ", y_test.shape)
```

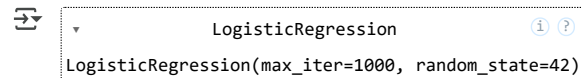


```
X_train shape : (1480098, 815140)
y_train shape : (1480098,)
X_test shape : (370025, 815140)
y_test shape : (370025,)
```

✓ Model Training

```
from sklearn.linear_model import LogisticRegression

# Initialize and train the Logistic Regression model
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train_tfidf, Y_train)
```

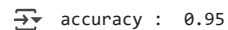


```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

```
y_pred = model.predict(X_test_tfidf[:1000])
y_actual = y_test[:1000]
```

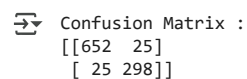
```
from sklearn.metrics import accuracy_score, confusion_matrix
```

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



```
accuracy : 0.95
```

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



```
Confusion Matrix :
[[652  25]
 [ 25 298]]
```