Natural Language Processing: Lab 1 - Preprocessing

In this Notebook, we are going to exam the first lab in Natural Language Processing DeepLearning.AI Course.

Objectives and Layouts:

- Jupyter - Lite Notebook Familiarization.
- Natural Language Processing Python Libraries.
- Preprocessing of given twitter dataset.

Part A: Natural Language Processing Python Library

We will be using the Natural Language Toolkit (NLTK) package, an open-source Python library for natural language processing. It has modules for collecting, handling, and processing Twitter data. For this exercise, we will use a Twitter dataset that comes with NLTK.

NLTK Library

In [ ]:
```
import nltk                                # Python Library for Natural Language Processing
from nltk.corpus import twitter_samples # NLTK twitter dataset
```

General Helping Libraries

In [ ]:
```
import matplotlib.pyplot as plt         # Matlab Python Library for visualization purposes
import random                           # Pseudo Random Number Generator
```

Twitter DataSet Overview

The sample dataset from NLTK is separated into positive and negative tweets. It contains 5000 positive tweets and 5000 negative tweets exactly. The exact match between these classes is not a coincidence. The intention is to have a balanced dataset. That **does not** reflect the real

distributions of positive and negative classes in live Twitter streams. It is just because balanced datasets simplify the design of most computational methods that are required for sentiment analysis. However, it is better to be aware that this balance of classes is **artificial**.

In [ ]:
```
import nltk                                 # Python Library for Natural Language Processing
from nltk.corpus import twitter_samples # NLTK twitter dataset
#-----------------------------------------------------------------------------------------
import matplotlib.pyplot as plt         # Matlab Python Library for visualization purposes
import random                               # Pseudo Random Number Generator
#-----------------------------------------------------------------------------------------
# select the set of positive and negative tweets
print("Nope")
# downloads sample twitter dataset.
nltk.download('twitter_samples')
```

```
Nope
[nltk_data] Downloading package twitter_samples to
[nltk_data]     /home/julian/nltk_data...
[nltk_data]   Unzipping corpora/twitter_samples.zip.
```

Out[ ]:
```
True
```

In [ ]:
```
#select the set of positive and negative tweets
all_positive_tweets = twitter_samples.strings('positive_tweets.json')
all_negative_tweets = twitter_samples.strings('negative_tweets.json')
```

We will now display the number of all positive as well as all negative tweets:

In [10]:
```
# display the length of all positive tweets
print('Number of positive tweets: ', len(all_positive_tweets))
# display the length of all negative tweets
print('Number of negative tweets: ', len(all_negative_tweets))
```

```
Number of positive tweets:  5000
Number of negative tweets:  5000
```

One important question may occur as what information do we have about the structure of the data under consideration. Let us first review the type of "all positive tweets".

In [11]:
```
print(type(all_positive_tweets))
```

```
<class 'list'>
```

Therefore we understand that we are dealing with a list, meaning that we have to acess any of the given 5,000 tweets by a numerical index 0 - 4,999. Let us now review the type of a tweet (or a list entry):

In [13]:
```
tweet_entry = all_positive_tweets[0]
print(tweet_entry)
```

```
#FollowFriday @France_Inte @PKuchly57 @Milipol_Paris for being top engaged members in my community this week :)
```

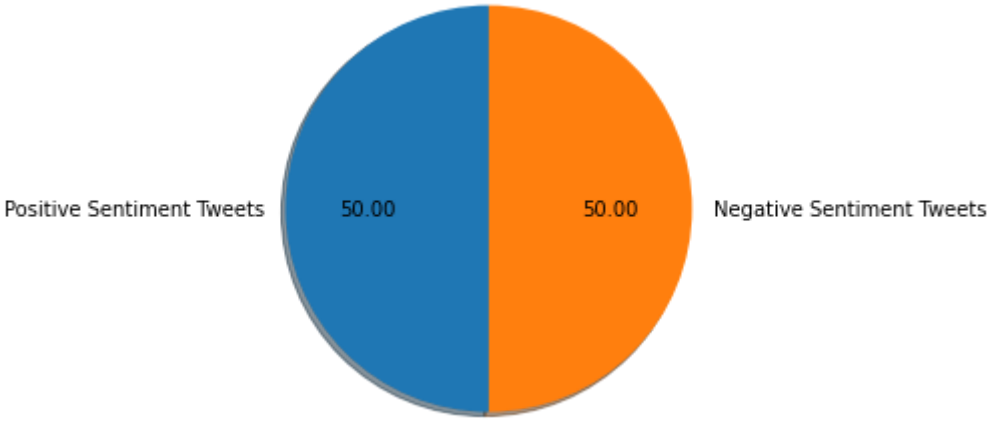
By the prementioned example it is clear that we are refering to a straight-forward string: namely, the tweet itself.

Matplotlib Visualization : Pie Chart

By the following example, we are going to examine a code snipet that is going to present to us, a graphical representation of the distribution of the data at hand (via the matplotlib library).

In [25]:
```
# Declare a figure with a custom size
# Input : (float,float) representing width and height in inches (1 inch = 2.54 cm)
fig = plt.figure(figsize=(4, 4))

# Labels for the two classes
labels = 'Positive Sentiment Tweets', 'Negative Sentiment Tweets'

# Sizes for each slide
sizes = [len(all_positive_tweets), len(all_negative_tweets)]

# Declare pie chart, where the slices will be ordered and plotted counter-clockwise:
# Input:
#     sizes: the wedge sizes
#     labels: the labels
#     autopct: A string or function used to label the wedges with their numeric value.
#              The label will be placed inside the wedge.
#              If it is a format string, the label will be fmt % pct.
#              If it is a function, it will be called.
#     shadow: Draw a shadow beneath the pie.
#     startangle: The angle by which the start of the pie is rotated, counterclockwise from the x-axis.
plt.pie(sizes, labels=labels, autopct='%.2f', shadow=True, startangle=90)

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')
```

```
# Display the chart
plt.show()
```



Data Initial Overview

Before starting the actual preprocessing functionality, it is constructive to initially observe the data at hand, in order to make observations regarding some common structural components of them. It is estimated, that understanding the data is responsible for 80% of the success or failure in data science projects. We can use this time to observe aspects we'd like to consider when preprocessing our data.

In [ ]:
```
#List of ASCII common color codes
Red = '\033[91m'
Green = '\033[92m'
Blue = '\033[94m'
Cyan = '\033[96m'
White = '\033[97m'
Yellow = '\033[93m'
Magenta = '\033[95m'
Grey = '\033[90m'
Black = '\033[90m'
Default = '\033[99m'

# Print a random integer in range [0-4999]
# random.randint(0,5000)
```

In [31]:
```
# Print positive in greeen
print('\033[92m' + all_positive_tweets[random.randint(0,4999)])
```

```
print('\033[92m' + all_positive_tweets[random.randint(0,4999)])
print('\033[92m' + all_positive_tweets[random.randint(0,4999)])
print('\033[92m' + all_positive_tweets[random.randint(0,4999)])
print('\033[92m' + all_positive_tweets[random.randint(0,4999)] + '\n')

# print negative in red
print('\033[91m' + all_negative_tweets[random.randint(0,4999)])
print('\033[91m' + all_negative_tweets[random.randint(0,4999)])
print('\033[91m' + all_negative_tweets[random.randint(0,4999)])
print('\033[91m' + all_negative_tweets[random.randint(0,4999)])
print('\033[91m' + all_negative_tweets[random.randint(0,4999)])
```

```
@BsquaredGaming like :-)
@pascale_blakey Great, thanks lovely :-). Almost 40 responses in the first hour!!! So delighted. Some really great comments on w
hat next.
Great start #SE100Leaders @hiSbe_Food , shining a light on doing good in business :) @LabelledShop
@luckybsmith I'm dead...thank you for doing this to me :))
@cyrenity I know he is, only saying what I know :)

Boohoo back to work tonight, another weekend on nights :-(:-(:-(
@HomeXpertsLeyla fyi the url on your profile doesn't work :(
@OhHeyItsAJ Bow down bitches to the queen of the world!!!! :(((
fnaf 4 coming after my holiday now :(
It's 3am and I can't sleep :(
```

One observation you may have is the presence of emoticons and URLs in many of the tweets. This info will come in handy in the next steps.