In [1]:
```
#!/usr/bin/env python
# coding: utf-8

# In[172]:


# Importing Libraries
import numpy as np
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

from Naive_Bayes_functions import *

# # Dataset creating and level 1 preprocessing
#
# Here we just load the data into python environment and create a pandas DataFrame
with required columns.
# We also store the final outcome as CSV inorder to just load the data for model bu
ilding and experiments


#Uncomment the line below to do this one time process
#If preprocessed_data_lvl_1.csv is already present no need to do it

lines = []
for the_text in open('naive_bayes_data.txt', encoding='utf8'):
    lines.append(the_text.strip('\n'))
the_data = pd.DataFrame(columns=["Id","Category","Text","Sentiment"],index=range(0,
len(lines)))
for idx,line in enumerate(lines):
    tokens = line.split(" ")
    the_data.loc[idx,"Category"] = tokens[0]
    the_data.loc[idx,"Sentiment"] = tokens[1]
    the_data.loc[idx,"Id"] = tokens[2].split(".")[0]
    the_data.loc[idx,"Text"] = ' '.join(word for word in tokens[3:])

the_data.to_csv("preprocessed_data_lvl_1.csv", index = False)

##############################################################################
############################################################

# # Text classification Preprocessing
#
# Here we do the the usual preprocessing required for text classification. This inc
ludes
# 1. Tokenisation
# 2. Punctuation removal (if required)
# 3. Stop words removal (if required)
# 4. Transforming text into features for building any Classification model (X)
#
# Note: Tokenisation could have been done in level 1 prepocessing itself but to mak
e the process in well strucuted manner for any given CSV file with a text columns,
we are doing it here


#loading the preprocessed data
the_data = pd.read_csv("preprocessed_data_lvl_1.csv")


#Tokenising
```

```
Confusion Matrix
Predicted   neg  pos  Total
Actual
neg         834  349   1183
pos         565  635   1200
Total      1399  984   2383



Accuracy: 61.64 %
Precision: 64.53 %
Recall: 52.92 %
f1-score: 58.15 %
```

In [ ]: