```
import os
import nltk
import nltk.corpus
```

```
nltk.download('punkt')
```

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

```
from nltk.tokenize import sent_tokenize
text="""Earth Day is an annual event on April 22 to demonstrate support for environmental
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['Earth Day is an annual event on April 22 to demonstrate support for environmental
```

```
from nltk.tokenize import word_tokenize
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['Earth', 'Day', 'is', 'an', 'annual', 'event', 'on', 'April', '22', 'to', 'demonstr
```

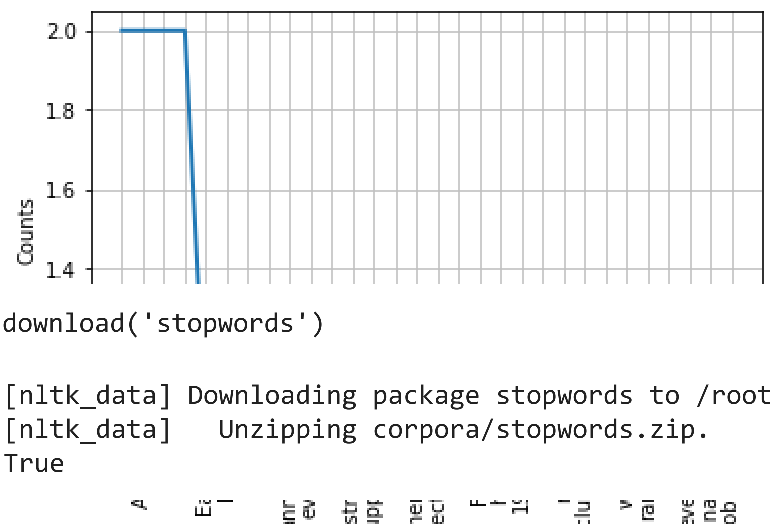
```
from nltk.probability import FreqDist
fdist = FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 41 samples and 45 outcomes>
```

```
fdist.most_common(2)
```

```
[('on', 2), ('April', 2)]
```

```
# Frequency Distribution Plot
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



```
nltk.download('stopwords')

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

```
from nltk.corpus import stopwords
stop_words=set(stopwords.words("english"))
print(stop_words)

     {'now', 'he', 've', 'but', 'whom', 'its', 'yours', 'below', 'when', 'all', 'too', 'y
```

```
filtered_sent=[]
for w in tokenized_word:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenized_word)
print("Filterd Sentence:",filtered_sent)

     Tokenized Sentence: ['Earth', 'Day', 'is', 'an', 'annual', 'event', 'on', 'April', '
     Filterd Sentence: ['Earth', 'Day', 'annual', 'event', 'April', '22', 'demonstrate',
```

```
# Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize

ps = PorterStemmer()

stemmed_words=[]
for w in filtered_sent:
    stemmed_words.append(ps.stem(w))

print("Filtered Sentence:",filtered_sent)
print("Stemmed Sentence:",stemmed_words)

     Filtered Sentence: ['Earth', 'Day', 'annual', 'event', 'April', '22', 'demonstrate',
     Stemmed Sentence: ['earth', 'day', 'annual', 'event', 'april', '22', 'demonstr', 'su
```

```
nltk.download('wordnet')

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

```
#Lexicon Normalization
#performing stemming and Lemmatization

from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()

from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()

word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))
```

```
     Lemmatized Word: fly
     Stemmed Word: fli
```

```
sent = "Hey hii how is your day?"
tokens=nltk.word_tokenize(sent)
print(tokens)
```

```
     ['Hey', 'hii', 'how', 'is', 'your', 'day', '?']
```

```
nltk.download('averaged_perceptron_tagger')
```

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

```
nltk.pos_tag(tokens)
```

```
     [('Hey', 'NNP'),
      ('hii', 'VBD'),
      ('how', 'WRB'),
      ('is', 'VBZ'),
      ('your', 'PRP$'),
      ('day', 'NN'),
      ('?', '.')]
```

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
```

Choose Files train.tsv

- **train.tsv**(n/a) - 8481022 bytes, last modified: 4/22/2021 - 100% done

```
     Saving train.tsv to train.tsv
```

```
data=pd.read_csv('train.tsv', sep='\t')
data.head()
```

| | PhraseId | SentenceId | Phrase | Sentiment |
|---|---|---|---|---|
| **0** | 1 | 1 | A series of escapades demonstrating the adage ... | 1 |
| **1** | 2 | 1 | A series of escapades demonstrating the adage ... | 2 |
| **2** | 3 | 1 | A series | 2 |
| **3** | 4 | 1 | A | 2 |
| **4** | 5 | 1 | series | 2 |

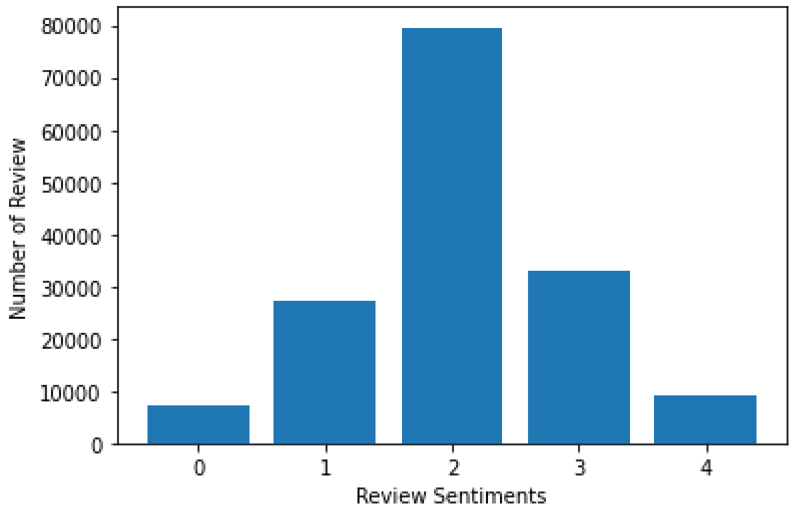
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156060 entries, 0 to 156059
Data columns (total 4 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   PhraseId    156060 non-null  int64 
 1   SentenceId  156060 non-null  int64 
 2   Phrase      156060 non-null  object
 3   Sentiment   156060 non-null  int64 
dtypes: int64(3), object(1)
memory usage: 4.8+ MB
```

```
data.Sentiment.value_counts()
```

```
2    79582
3    32927
1    27273
4     9206
0     7072
Name: Sentiment, dtype: int64
```

```
Sentiment_count=data.groupby('Sentiment').count()
plt.bar(Sentiment_count.index.values, Sentiment_count['Phrase'])
plt.xlabel('Review Sentiments')
plt.ylabel('Number of Review')
plt.show()
```



```
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
#tokenizer to remove unwanted elements from out data like symbols and numbers
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(lowercase=True,stop_words='english',ngram_range = (1,1),tokenizer = t
text_counts= cv.fit_transform(data['Phrase'])
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    text_counts, data['Sentiment'], test_size=0.3, random_state=1)
from sklearn.naive_bayes import MultinomialNB
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
MultinomialNB Accuracy: 0.6049169122986885
```

## New Section

✓ 1s completed at 12:36 PM