Sentiment Analysis

Main goal of sentiment analysis is to preserve the meaning of the word. previously we were only concerned with the structure of the sentence but that will not solve our problem in NLP.

Part 1: Simple analyser

Part 2: LSTM and Naive Bayes

Lexicon

It is a mechanism or method that we use to classify your sentence, every lexicon has threshold values and based upon those it classfies sentences to positive, negative and neutral.

VADER: It is a rule based lexicon which is used in our model to evaluate and check where the sentence is +ve, -ve or neutral. The value ranges from (-1 to +1). If a sentence is passed to a vader and score is above 0.05 then it is treated as a +ve sentence. Else, it is a negative sentence.

Steps:

Step 1: Data Cleaning

1. Identify Noise
2. Noise Removal
3. Character Normalisation: transforming your data (text). Lowercasing (convert all text to lowercase, due to ASCII codes), Converting your special characters (punctuation, emoji), Solving or handling the encoding issue
4. Data Masking: Hiding your private information that is being transmitted

character normalisation, why? - to increase your recall value

After this you get clean text

Step 2: Linguistic Processing

1. Tokenization
2. POS Tagging
3. Lemmatization
4. Named Entity Recognition

After this you get prepared text that is fed to VADER

Question 1: Create a sentiment analysis model to classify text as positive, negative or neutral

In [1]:
```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.sentiment import SentimentIntensityAnalyzer
```

In [2]:
```
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\ahana\AppData\Roaming\nltk_data...
```

Out[2]:
```
True
```

In [3]:
```
def preprocess(text):
    tokens = word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [token.lower() for token in tokens if token.isalpha() and token.lower() not in stop_words]
    return filtered_tokens

def analyze_sentiment(text):
    preprocessed_text = preprocess(text)

    #Join tokens back into string
    preprocessed_text = ' '.join(preprocessed_text)

    #Initialise the sentiment analyzer
    sia = SentimentIntensityAnalyzer()

    #Analyze sentiment
    sentiment_scores = sia.polarity_scores(preprocessed_text)

    #Determine sentiment label
    if sentiment_scores['compound'] >= 0.05:
        sentiment = 'Positive'
    elif sentiment_scores['compound'] <= -0.05:
        sentiment = 'Negative'
    else:
        sentiment = "Neutral"
    return sentiment, sentiment_scores

#Sample Text
text = "I love this product! It's amazing and works perfectly!"
sentiment, sentiment_scores = analyze_sentiment(text)
print("Sentiment: ",sentiment)
print("Sentiment Score: ", sentiment_scores)
```

```
Sentiment:  Positive
Sentiment Score:  {'neg': 0.0, 'neu': 0.141, 'pos': 0.859, 'compound': 0.9217}
```

Here compound refers to the total score of the sentence

In [11]:
```python
import pandas as pd
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
```

In [12]:
```python
df = pd.read_csv('text.csv')
print(df)
```

```
                                              reviewText  Positive
0      This is a one of the best apps acording to a b...         1
1      This is a pretty good version of the game for ...         1
2      this is a really cool game. there are a bunch ...         1
3      This is a silly game and can be frustrating, b...         1
4      This is a terrific game on any pad. Hrs of fun...         1
...                                                  ...       ...
19995  this app is fricken stupid.it froze on the kin...         0
19996  Please add me!!!!! I need neighbors! Ginger101...         1
19997  love it!  this game. is awesome. wish it had m...         1
19998  I love love love this app on my side of fashio...         1
19999  This game is a rip off. Here is a list of thin...         0

[20000 rows x 2 columns]
```

In [13]:
```python
#Initialise NLTK Sentiment Analyzer
analyzer = SentimentIntensityAnalyzer()

#Create get_sentiment function
def get_sentiment(text):
    scores = analyzer.polarity_scores(text)
    sentiment = 1 if scores['pos'] > 0 else 0
    return sentiment

#apply function
df['sentiment'] = df['reviewText'].apply(get_sentiment)
df
```

Out[13]:

	reviewText	Positive	sentiment
0	This is a one of the best apps acording to a b...	1	1
1	This is a pretty good version of the game for ...	1	1
2	this is a really cool game. there are a bunch ...	1	1
3	This is a silly game and can be frustrating, b...	1	1
4	This is a terrific game on any pad. Hrs of fun...	1	1
...	...	...	...
19995	this app is fricken stupid.it froze on the kin...	0	0
19996	Please add me!!!!! I need neighbors! Ginger101...	1	1
19997	love it! this game. is awesome. wish it had m...	1	1
19998	I love love love this app on my side of fashio...	1	1
19999	This game is a rip off. Here is a list of thin...	0	1

20000 rows × 3 columns

In [14]:
```python
from sklearn.metrics import confusion_matrix
print(confusion_matrix(df['Positive'], df['sentiment']))
```

```
[[ 1377  3390]
 [  620 14613]]
```

In [16]:
```python
from sklearn.metrics import classification_report
print(classification_report(df['Positive'],df['sentiment']))
```

```
              precision    recall  f1-score   support

           0       0.69      0.29      0.41      4767
           1       0.81      0.96      0.88     15233

    accuracy                           0.80     20000
   macro avg       0.75      0.62      0.64     20000
weighted avg       0.78      0.80      0.77     20000
```

Finding the degree of Accuracy

In [17]:
```
!pip install vaderSentiment
```

```
Collecting vaderSentiment
  Downloading vaderSentiment-3.3.2-py2.py3-none-any.whl (125 kB)
     ------------------------------------ 126.0/126.0 kB 672.7 kB/s eta 0:00:00
Requirement already satisfied: requests in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from vaderSentimen
t) (2.28.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (fr
om requests->vaderSentiment) (3.0.1)
Requirement already satisfied: idna<4,>=2.5 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from requests-
>vaderSentiment) (3.4)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from
requests->vaderSentiment) (1.26.14)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\ahana\appdata\local\programs\python\python311\lib\site-packages (from req
uests->vaderSentiment) (2022.12.7)
Installing collected packages: vaderSentiment
Successfully installed vaderSentiment-3.3.2


[notice] A new release of pip available: 22.3.1 -> 24.0
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [18]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

In [22]:
```
#function to print sentiments of the sentence
def sentiment_scores(sentence):

    #Create a sentimentintensityanalyzer object
    sia_obj = SentimentIntensityAnalyzer()

    #polarity scores method of SIA object gives a sentiment dictionary which contains pos,neg,neu and compound scores
    sentiment_dict = sia_obj.polarity_scores(sentence)

    print("Overall sentiment dictionary is: ",sentiment_dict)
    print("Sentence was rated as: ", sentiment_dict['neg']*100, "% Negavtive")
    print("Sentence was rated as: ", sentiment_dict['neu']*100, "% Neutral")
    print("Sentence was rated as: ", sentiment_dict['pos']*100, "% Positive")
    print("Sentence Overall rated as: ",end = " ")

    if sentiment_dict['compound'] >= 0.05:
        print("Positive")
    elif sentiment_dict['compound']<= -0.05:
        print("negative")
    else:
        print("Neutral")

#Driver Code
if __name__ == "__main__":
    print("\n1st Statement: ")
    sentence = "AHANA SADH IS EXTREMELY HAPPY TODAY"
    sentiment_scores(sentence)

    print("\n2nd Statement: ")
    sentence = "She is attending class as usual"
    sentiment_scores(sentence)

    print("\3rd Statement: ")
    sentence = "Result is declared and she is extremely sad today"
    sentiment_scores(sentence)
```

```
1st Statement:
Overall sentiment dictionary is:  {'neg': 0.0, 'neu': 0.556, 'pos': 0.444, 'compound': 0.6115}
Sentence was rated as:  0.0 % Negavtive
Sentence was rated as:  55.60000000000001 % Neutral
Sentence was rated as:  44.4 % Positive
Sentence Overall rated as:  Positive

2nd Statement:
Overall sentiment dictionary is:  {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Sentence was rated as:  0.0 % Negavtive
Sentence was rated as:  100.0 % Neutral
Sentence was rated as:  0.0 % Positive
Sentence Overall rated as:  Neutral
␃rd Statement:
Overall sentiment dictionary is:  {'neg': 0.298, 'neu': 0.702, 'pos': 0.0, 'compound': -0.5256}
Sentence was rated as:  29.799999999999997 % Negavtive
Sentence was rated as:  70.19999999999999 % Neutral
Sentence was rated as:  0.0 % Positive
Sentence Overall rated as:  negative
```

Assignment: Use different algorithms (LSTM and Naive Bayes) and compare their performance

In [23]:

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv("Dataset.csv")

texts = df['review'].astype(str).tolist()
labels = df['sentiment'].tolist()

X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.2, random_state=42)


vectorizer = CountVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)



nb_classifier = MultinomialNB()
nb_classifier.fit(X_train_vectorized, y_train)

y_pred = nb_classifier.predict(X_test_vectorized)

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")
print(classification_report(y_test, y_pred))


new_texts = ["Terrible experience", "Hated it."]
new_texts_vectorized = vectorizer.transform(new_texts)
new_predictions = nb_classifier.predict(new_texts_vectorized)

for i, text in enumerate(new_texts):
    print(f"Text: {text}\nPredicted Sentiment: {'Positive' if new_predictions[i] == 1 else 'Negative'}\n")
```

```
Accuracy: 0.85
Classification Report:
              precision    recall  f1-score   support

    negative       0.83      0.88      0.85      4961
    positive       0.87      0.82      0.85      5039

    accuracy                           0.85     10000
   macro avg       0.85      0.85      0.85     10000
weighted avg       0.85      0.85      0.85     10000

Text: Terrible experience
Predicted Sentiment: Negative

Text: Hated it.
Predicted Sentiment: Negative
```

In [ ]:

```
```