

EXPERIMENT NO. 3

Aim:

To build an adaptive and contextual cognitive-based customer service application.

Theory:

Introduction:

Sentiment analysis, also known as opinion mining, is a crucial text mining technique in the field of Natural Language Processing (NLP) and Machine Learning (ML). It involves analyzing textual data to determine the sentiment or emotional tone expressed by the writer. The sentiment could be categorized as positive, negative, neutral, or even more nuanced emotions. This technology has significant applications in customer service, where understanding customer sentiments can enhance decision-making, improve customer satisfaction, and tailor responses to meet customer needs.

Objective:

The aim of this lab is to build an adaptive and cognitive-based customer service application using sentiment analysis. This application will analyze customer interactions in real-time, categorize their sentiments, and provide contextual responses based on the analysis. The goal is to enhance customer service by providing more personalized and empathetic responses, thus improving overall customer experience.

Text Mining and Sentiment Analysis:

Text mining refers to the process of deriving high-quality information from text. It involves extracting meaningful patterns, trends, and relationships from large volumes of textual data. Sentiment analysis is a specialized form of text mining focused on identifying and categorizing opinions expressed in text, particularly in terms of their sentiment polarity (positive, negative, or neutral).

Machine Learning and NLP Techniques:

Sentiment analysis leverages advanced machine learning algorithms and natural language processing techniques to understand and classify sentiments within text data. These algorithms are trained on large datasets to recognize sentiment-related patterns and subtleties in language.

Applications in Customer Service:

In a customer service context, sentiment analysis enables businesses to:

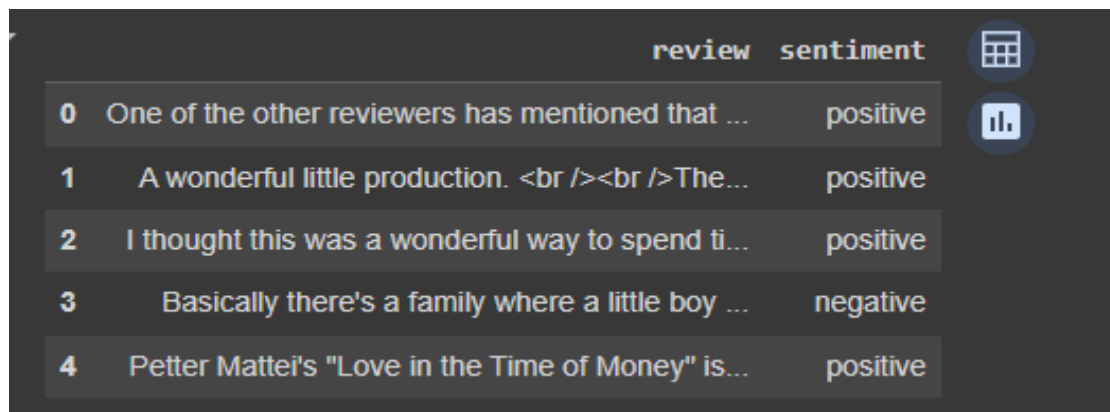
- Automatically gauge customer satisfaction and identify areas for improvement.
- Prioritize and route customer queries based on the sentiment.
- Tailor responses to match the customer's emotional tone, thereby improving the customer experience.

Code and Output:

Manually Creating a Model for Text Sentiment Analysis

1. Import necessary libraries and load the dataset:

```
from sklearn.feature_extraction.text import CountVectorizer  
  
from nltk.tokenize import RegexpTokenizer  
  
import pandas as pd  
  
data = pd.read_csv('/content/IMDB Dataset.csv')  
  
data.head()
```



	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

2. Tokenization and Vectorization:

```
token = RegexpTokenizer(r'[a-zA-Z0-9]+')  
  
cv = CountVectorizer(stop_words='english', ngram_range=(1,1), tokenizer=token.tokenize)  
  
text_counts = cv.fit_transform(data['review'])
```

3. Split the data into training and testing sets:

```
from sklearn.model_selection import train_test_split

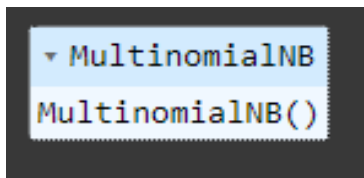
X_train, X_test, Y_train, Y_test = train_test_split(text_counts, data['sentiment'], test_size=0.25,
random_state=5)
```

4. Train a Naive Bayes classifier:

```
from sklearn.naive_bayes import MultinomialNB

MNB = MultinomialNB()

MNB.fit(X_train, Y_train)
```



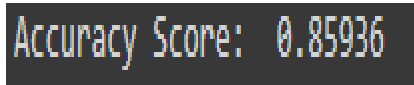
5. Evaluate the model:

```
from sklearn import metrics

predicted = MNB.predict(X_test)

accuracy_score = metrics.accuracy_score(predicted, Y_test)

print("Accuracy Score: ", accuracy_score)
```

A screenshot of a Jupyter Notebook cell with a dark background. It shows the output of the accuracy score calculation: 'Accuracy Score: 0.85936'.

6. Using TextBlob for Sentiment Analysis:

```
from textblob import TextBlob

text_1 = "It was a good idea."

text_2 = "It was not a good idea."

polarity_1 = TextBlob(text_1).sentiment.polarity
```

```
polarity_2 = TextBlob(text_2).sentiment.polarity  
  
print("Polarity of Text 1:", polarity_1)  
  
print("Polarity of Text 2:", polarity_2)  
  
subjectivity_1 = TextBlob(text_1).sentiment.subjectivity  
  
subjectivity_2 = TextBlob(text_2).sentiment.subjectivity  
  
print("Subjectivity of Text 1:", subjectivity_1)  
  
print("Subjectivity of Text 2:", subjectivity_2)
```

```
Polarity of Text 1: 0.7  
Polarity of Text 2: -0.35  
Subjectivity of Text 1: 0.6000000000000001  
Subjectivity of Text 2: 0.6000000000000001
```

7. Using VADER (Valence Aware Dictionary and sEntiment Reasoner):

```
!pip install vaderSentiment  
  
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer  
  
sentiment = SentimentIntensityAnalyzer()  
  
sent_1 = sentiment.polarity_scores(text_1)  
  
sent_2 = sentiment.polarity_scores(text_2)  
  
print("Sentiment of Text 1:", sent_1)  
  
print("Sentiment of Text 2:", sent_2)
```

```
Installing collected packages: vaderSentiment  
Successfully installed vaderSentiment-3.3.2  
Sentiment of Text 1: {'neg': 0.0, 'neu': 0.58, 'pos': 0.42, 'compound': 0.4404}  
Sentiment of Text 2: {'neg': 0.325, 'neu': 0.675, 'pos': 0.0, 'compound': -0.3412}
```

8. Using Transformer-Based Models:

```
!pip install transformers -q
```

```
from transformers import pipeline
```

```
sentiment_pipeline = pipeline("sentiment-analysis")
```

```
data = ["The weather was awesome.", "My head is paining"]
```

```
results = sentiment_pipeline(data)
```

```
print(results)
```

```
[{'label': 'POSITIVE', 'score': 0.9998689889907837}, {'label': 'NEGATIVE', 'score': 0.9986617565155029}]
```

Conclusion:

Thus, we studied an overview of what is Text Sentiment Analysis and implemented an adaptive and contextual cognitive-based customer service application.