CISB5123 Text Analytics
Lab 7
Sentiment Analysis

Sentiment Analysis is the process of classifying the content of **documents** as **positive, negative** and/or **neutral**.

Sentiment analysis encompasses a variety of methods and techniques, which can be broadly categorized into lexicon-based and machine-learning based approaches.

<u>Lexicon-based Approach</u>

Text Blob and VADER are among the popular lexicons for sentiment analysis.

1. Import the required libraries for sentiment analysis (TextBlob and SentimentIntensityAnalyzer from VADER), and the tabulate library for displaying data in a table format.

```
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from tabulate import tabulate
```

2. Create sample data consisting of text samples along with their corresponding actual sentiment labels.

```
data = [
    ("I love this product, it's amazing!", 'positive'),
    ("This product is terrible, I hate it.", 'negative'),
    ("It's okay, not bad but not great either.", 'neutral'),
```

```
    ("Best product ever, highly recommended!", 'positive'),
    ("I'm really disappointed with the quality.", 'negative'),
    ("So-so product, nothing special about it.", 'neutral'),
    ("The customer service was excellent!", 'positive'),
    ("I wasted my money on this useless product.", 'negative'),
    ("It's not the worst, but certainly not the best.", 'neutral'),
    ("I can't live without this product, it's a lifesaver!", 'positive'),
    ("The product arrived damaged and unusable.", 'negative'),
    ("It's average, neither good nor bad.", 'neutral'),
    ("Highly disappointed with the purchase.", 'negative'),
    ("The product exceeded my expectations.", 'positive'),
    ("It's just okay, nothing extraordinary.", 'neutral'),
    ("This product is excellent, it exceeded all my expectations!", 'positive'),
    ("I regret purchasing this product, it's a waste of money.", 'negative'),
    ("It's neither good nor bad, just average.", 'neutral'),
    ("Outstanding customer service, highly recommended!", 'positive'),
    ("I'm very disappointed with the quality of this item.", 'negative'),
    ("It's not the best product, but it gets the job done.", 'neutral'),
    ("This product is a game-changer, I can't imagine life without it!", 'positive'),
    ("I received a defective product, very dissatisfied.", 'negative'),
    ("It's neither great nor terrible, just okay.", 'neutral'),
    ("Fantastic product, I would buy it again in a heartbeat!", 'positive'),
    ("Avoid this product at all costs, complete waste of money.", 'negative'),
    ("It's decent, but nothing extraordinary.", 'neutral'),
    ("Impressive quality, exceeded my expectations!", 'positive'),
    ("I'm very unhappy with this purchase, total disappointment.", 'negative'),
    ("It's neither amazing nor terrible, somewhere in between.", 'neutral')
]
```

3. Initialize an empty list to store the data in tabular format.

```
table_data = [["Text", "Actual Label", "TextBlob Sentiment", "VADER Sentiment"]]
```

4. Loop through each text in the sample data and analyze its sentiment using both TextBlob and VADER. Determine the sentiment label based on the sentiment score obtained.

```
for text, actual_label in data:
    # TextBlob
    blob = TextBlob(text)
```

```
    tb_polarity = blob.sentiment.polarity

    # Determine label based on polarity score from TextBlob
    if tb_polarity > 0:
        tb_label = 'positive'
    elif tb_polarity < 0:
        tb_label = 'negative'
    else:
        tb_label = 'neutral'

    # VADER
    analyzer = SentimentIntensityAnalyzer()
    vs = analyzer.polarity_scores(text)
    vader_compound = vs['compound']

    # Determine label based on compound score from VADER
    if vader_compound > 0.05:
        vader_label = 'positive'
    elif vader_compound < -0.05:
        vader_label = 'negative'
    else:
        vader_label = 'neutral'

    table_data.append([text, actual_label, tb_label, vader_label])
```

5. Print the sentiment analysis results in a table format using the tabulate library.

```
print(tabulate(table_data, headers="firstrow"))
```

Output:

```
Text                                        Actual Label    TextBlob Sentiment    VADER Sentiment
------------------------------------------  --------------  --------------------  -----------------
I love this product, it's amazing!          positive        positive              positive
This product is terrible, I hate it.        negative        negative              negative
It's okay, not bad but not great either.    neutral         positive              negative
Best product ever, highly recommended!      positive        positive              positive
I'm really disappointed with the quality.   negative        negative              negative
So-so product, nothing special about it.    neutral         positive              negative
I love this product, it's amazing!          positive        positive              positive
This product is terrible, I hate it.        negative        negative              negative
It's okay, not bad but not great either.    neutral         positive              negative
Best product ever, highly recommended!      positive        positive              positive
I'm really disappointed with the quality.   negative        negative              negative
So-so product, nothing special about it.    neutral         positive              negative
```

6. Display the classification report for both Text Blob and VADER. Modify the code as follows:

```
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from sklearn.metrics import classification_report
from tabulate import tabulate

# Sample data for demonstration
data = [
    ("I love this product, it's amazing!", 'positive'),
    ("This product is terrible, I hate it.", 'negative'),
    ("It's okay, not bad but not great either.", 'neutral'),
    ("Best product ever, highly recommended!", 'positive'),
    ("I'm really disappointed with the quality.", 'negative'),
    ("So-so product, nothing special about it.", 'neutral'),
    ("The customer service was excellent!", 'positive'),
    ("I wasted my money on this useless product.", 'negative'),
    ("It's not the worst, but certainly not the best.", 'neutral'),
    ("I can't live without this product, it's a lifesaver!", 'positive'),
    ("The product arrived damaged and unusable.", 'negative'),
    ("It's average, neither good nor bad.", 'neutral'),
    ("Highly disappointed with the purchase.", 'negative'),
    ("The product exceeded my expectations.", 'positive'),
    ("It's just okay, nothing extraordinary.", 'neutral'),
    ("This product is excellent, it exceeded all my expectations!", 'positive'),
    ("I regret purchasing this product, it's a waste of money.", 'negative'),
    ("It's neither good nor bad, just average.", 'neutral'),
    ("Outstanding customer service, highly recommended!", 'positive'),
    ("I'm very disappointed with the quality of this item.", 'negative'),
    ("It's not the best product, but it gets the job done.", 'neutral'),
    ("This product is a game-changer, I can't imagine life without it!", 'positive'),
    ("I received a defective product, very dissatisfied.", 'negative'),
    ("It's neither great nor terrible, just okay.", 'neutral'),
    ("Fantastic product, I would buy it again in a heartbeat!", 'positive'),
    ("Avoid this product at all costs, complete waste of money.", 'negative'),
    ("It's decent, but nothing extraordinary.", 'neutral'),
    ("Impressive quality, exceeded my expectations!", 'positive'),
    ("I'm very unhappy with this purchase, total disappointment.", 'negative'),
    ("It's neither amazing nor terrible, somewhere in between.", 'neutral')
```

```
]

# Initialize an empty list to store the data in tabular format
table_data = [["Text", "Actual Label", "TextBlob Sentiment", "VADER Sentiment"]]

# Lexicon-based approach using TextBlob and VADER
print("Lexicon-based approach:")
for text, actual_label in data:
    # TextBlob
    blob = TextBlob(text)
    tb_polarity = blob.sentiment.polarity

    # Determine label based on polarity score from TextBlob
    if tb_polarity > 0:
        tb_label = 'positive'
    elif tb_polarity < 0:
        tb_label = 'negative'
    else:
        tb_label = 'neutral'

    # VADER
    analyzer = SentimentIntensityAnalyzer()
    vs = analyzer.polarity_scores(text)
    vader_compound = vs['compound']

    # Determine label based on compound score from VADER
    if vader_compound > 0.05:
        vader_label = 'positive'
    elif vader_compound < -0.05:
        vader_label = 'negative'
    else:
        vader_label = 'neutral'

    table_data.append([text, actual_label, tb_label, vader_label])

# Print the sentiment analysis results in a table format
print(tabulate(table_data, headers="firstrow"))

# Calculate classification report for TextBlob
```

```
tb_classification_report = classification_report([label for _, label in data], [tb_label for _, _, tb_label, _ in table_data[1:]], target_names=['negative', 'neutral', 'positive'])

# Calculate classification report for VADER
vader_classification_report = classification_report([label for _, label in data], [vader_label for _, _, _, vader_label in table_data[1:]], target_names=['negative', 'neutral', 'positive'])

# Print classification report for TextBlob
print("\nClassification Report for TextBlob:")
print(tb_classification_report)

# Print classification report for VADER
print("\nClassification Report for VADER:")
print(vader_classification_report)
```

Output:

```
Classification Report for TextBlob:
              precision    recall  f1-score   support

    negative       1.00      1.00      1.00         2
     neutral       0.00      0.00      0.00         2
    positive       0.50      1.00      0.67         2

    accuracy                           0.67         6
   macro avg       0.50      0.67      0.56         6
weighted avg       0.50      0.67      0.56         6


Classification Report for VADER:
              precision    recall  f1-score   support

    negative       0.50      1.00      0.67         2
     neutral       0.00      0.00      0.00         2
    positive       1.00      1.00      1.00         2

    accuracy                           0.67         6
   macro avg       0.50      0.67      0.56         6
weighted avg       0.50      0.67      0.56         6
```

Machine learning-based Approach

1. Import necessary libraries:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report
```

2. Create sample data:

```
data = [
    ("I love this product, it's amazing!", 'positive'),
    ("This product is terrible, I hate it.", 'negative'),
    ("It's okay, not bad but not great either.", 'neutral'),
    ("Best product ever, highly recommended!", 'positive'),
    ("I'm really disappointed with the quality.", 'negative'),
    ("So-so product, nothing special about it.", 'neutral'),
    ("The customer service was excellent!", 'positive'),
    ("I wasted my money on this useless product.", 'negative'),
    ("It's not the worst, but certainly not the best.", 'neutral'),
    ("I can't live without this product, it's a lifesaver!", 'positive'),
    ("The product arrived damaged and unusable.", 'negative'),
    ("It's average, neither good nor bad.", 'neutral'),
    ("Highly disappointed with the purchase.", 'negative'),
    ("The product exceeded my expectations.", 'positive'),
    ("It's just okay, nothing extraordinary.", 'neutral'),
    ("This product is excellent, it exceeded all my expectations!", 'positive'),
    ("I regret purchasing this product, it's a waste of money.", 'negative'),
    ("It's neither good nor bad, just average.", 'neutral'),
    ("Outstanding customer service, highly recommended!", 'positive'),
    ("I'm very disappointed with the quality of this item.", 'negative'),
    ("It's not the best product, but it gets the job done.", 'neutral'),
    ("This product is a game-changer, I can't imagine life without it!", 'positive'),
    ("I received a defective product, very dissatisfied.", 'negative'),
    ("It's neither great nor terrible, just okay.", 'neutral'),
    ("Fantastic product, I would buy it again in a heartbeat!", 'positive'),
    ("Avoid this product at all costs, complete waste of money.", 'negative'),
    ("It's decent, but nothing extraordinary.", 'neutral'),
```

```
    ("Impressive quality, exceeded my expectations!", 'positive'),
    ("I'm very unhappy with this purchase, total disappointment.", 'negative'),
    ("It's neither amazing nor terrible, somewhere in between.", 'neutral')
]
```

3. Split data:

```
# Split data into training and testing sets
texts = [text for text, _ in data]
labels = [label for _, label in data]
X_train, X_test, y_train, y_test = train_test_split(texts, labels, test_size=0.4,
random_state=42)
```

4. Extract features:

```
# Extract features (bag of words representation)
vectorizer = CountVectorizer()
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

5. Initialize classifiers:

```
# Initialize classifiers
nb_classifier = MultinomialNB()
svm_classifier = SVC(kernel='linear')
```

6. Train classifiers:

```
# Train classifiers
nb_classifier.fit(X_train, y_train)
svm_classifier.fit(X_train, y_train)
```

7. Predict sentiment on test data:

```
# Predict sentiment using classifiers
for text, actual_label in zip(X_test, y_test):
    # Predict sentiment using Naive Bayes
    nb_prediction = nb_classifier.predict(text)[0]
```

```
# Predict sentiment using SVM
svm_prediction = svm_classifier.predict(text)[0]
```

8. Calculate and display classification report:

```
# Calculate classification report for Naive Bayes
nb_classification_report = classification_report(y_test, nb_classifier.predict(X_test), target_names=['negative', 'neutral', 'positive'])

# Calculate classification report for SVM
svm_classification_report = classification_report(y_test, svm_classifier.predict(X_test), target_names=['negative', 'neutral', 'positive'])

# Print classification report for Naive Bayes
print("\nClassification Report for Naive Bayes:")
print(nb_classification_report)

# Print classification report for SVM
print("\nClassification Report for SVM:")
print(svm_classification_report)
```

Output:

```
Classification Report for Naive Bayes:
              precision    recall  f1-score   support

    negative       0.80      1.00      0.89         4
     neutral       0.75      1.00      0.86         3
    positive       1.00      0.60      0.75         5

    accuracy                           0.83        12
   macro avg       0.85      0.87      0.83        12
weighted avg       0.87      0.83      0.82        12


Classification Report for SVM:
              precision    recall  f1-score   support

    negative       0.75      0.75      0.75         4
     neutral       0.75      1.00      0.86         3
    positive       0.75      0.60      0.67         5

    accuracy                           0.75        12
   macro avg       0.75      0.78      0.76        12
weighted avg       0.75      0.75      0.74        12
```

Interpretation of the classification reports for both lexicon-based and machine learning-based approaches.

- Text Blob: TextBlob has relatively higher precision for negative (0.67) and positive (0.53) sentiments compared to neutral. It also shows good recall for negative (0.80) and positive (0.80) sentiments.
- VADER: VADER has higher precision for negative (0.56) and positive (0.89) sentiments compared to neutral (0.33). It also shows high recall for negative (1.00) and positive (0.80) sentiments.
- Naïve Bayes: Naive Bayes shows high precision, recall, and F1-score for all three sentiment classes (negative, neutral, positive).
- Support Vector Machine: SVM also demonstrates good precision, recall, and F1-score for all sentiment classes.
- Overall, machine-learning-based approaches perform better compared to lexicon-based approaches (TextBlob and VADER) in this specific dataset. Naive Bayes shows the highest accuracy among all classifiers, while SVM performs slightly lower but still reasonably well.