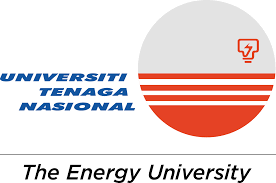
****

**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.

Lexicon-based Approach

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 |

1. 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')  ] |

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

|  |
| --- |
| table\_data = [["Text", "Actual Label", "TextBlob Polarity", "TextBlob Sentiment", "VADER Compound", "VADER Sentiment"]] |

1. 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\_polarity, tb\_label, vader\_compound, vader\_label]) |

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

|  |
| --- |
| print(tabulate(table\_data, headers="firstrow", tablefmt="plain")) |

Output:

A screenshot of a computer screen

Description automatically generated

1. 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 Polarity", "TextBlob Sentiment", "VADER Compound", "VADER Sentiment"]]  # Lexicon-based approach using TextBlob and VADER  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\_polarity, tb\_label, vader\_compound, vader\_label])  print(tabulate(table\_data, headers="firstrow", tablefmt="plain"))  **# Calculate classification report for TextBlob**  **tb\_classification\_report = classification\_report([label for \_, label, \_, tb\_label, \_, \_ in table\_data[1:]], [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, \_, \_, \_, vader\_label in table\_data[1:]], [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:

A screenshot of a computer

Description automatically generated

A screenshot of a report

Description automatically generated

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 |

1. 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')  ] |

1. 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) |

1. Extract features:

|  |
| --- |
| # Extract features (bag of words representation)  vectorizer = CountVectorizer()  X\_train = vectorizer.fit\_transform(X\_train)  X\_test = vectorizer.transform(X\_test) |

1. Initialize classifiers:

|  |
| --- |
| # Initialize classifiers  nb\_classifier = MultinomialNB()  svm\_classifier = SVC(kernel='linear') |

1. Train classifiers:

|  |
| --- |
| # Train classifiers  nb\_classifier.fit(X\_train, y\_train)  svm\_classifier.fit(X\_train, y\_train) |

1. 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] |

1. 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:

A screenshot of a computer screen

Description automatically generated

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.