Ain Shams University
Faculty of Computers & Information Sciences
Computer Science Department

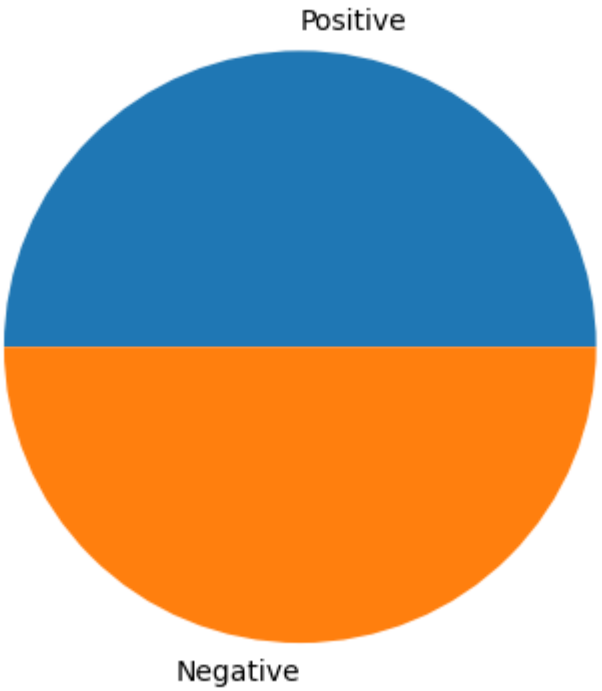
Sentiment Analysis of Movie Reviews

Natural Language Processing

Team ID: T30

Reading & Exploring Data:

The movies review dataset consists of two classes (positive - negative), each containing 1000 reviews.



The distribution of the classes indicates that the data is evenly split between the positive and the negative reviews which is advantageous to the training model as it prevents bias and is crucial for building a robust sentiment analysis model.

Preprocessing:

- Remove Non-Alphanumeric Characters:

  Non-Alphanumeric characters were removed while maintaining the whitespace between words of each review and making sure to preserve context and facilitate analysis.

  Regex: `[^a-zA-Z0-9\s]+`

- Tokenization:

  The reviews contain multiple sentiments and arguments within a single paragraph, which makes word tokenization a great candidate as it provides fine-grained information about sentiment expressed on the word level.

- Remove Stopwords:

  Stopwords were identified using the NLTK library to access a predefined list of English stopwords, and they were removed as they are irrelevant and commonly occurring words that do not contribute to the overall meaning of the sentiment or text.

- Lemmatization:

  Lemmatization is the step where words are converted to their base forms (root), which allows for better generalization and reducing dimensionality of data.

- Join Tokens:

  The last step of the preprocessing is to reconstruct the movie review as a string by joining the preprocessed tokens together, to extract features from them.

Feature Extraction:

- Methodology:

  We used the TfidfVectorizer from scikit-learn for feature extraction. We processed the text data by removing stopwords and converting it into a TF-IDF representation. We limited the number of features to 32000 using the max_features parameter.

- Results

  After applying TF-IDF, we obtained a feature matrix with dimensions (2000, 5000). Each element in the matrix represents the importance of a term in a document based on its frequency and rarity.

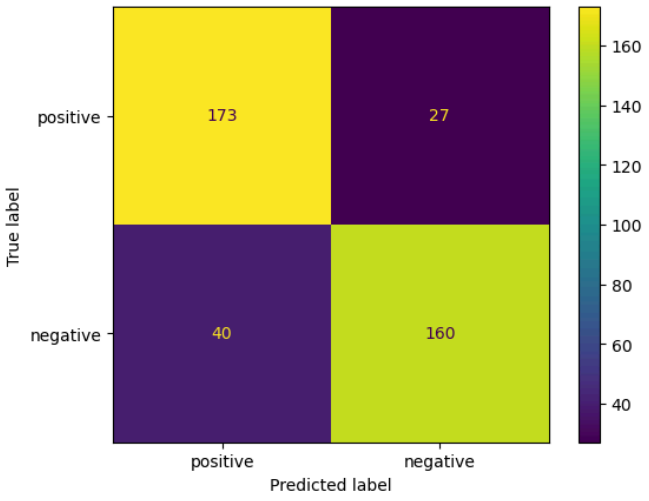
Modeling:

- Naive Bayes:
  - Methodology:

    We applied the MultinomialNB classifier from scikit-learn for text classification. After training the model on the TF-IDF transformed training data (X_train, y_train), we made predictions on the test data (X_test) and assessed its performance.

  - Results:

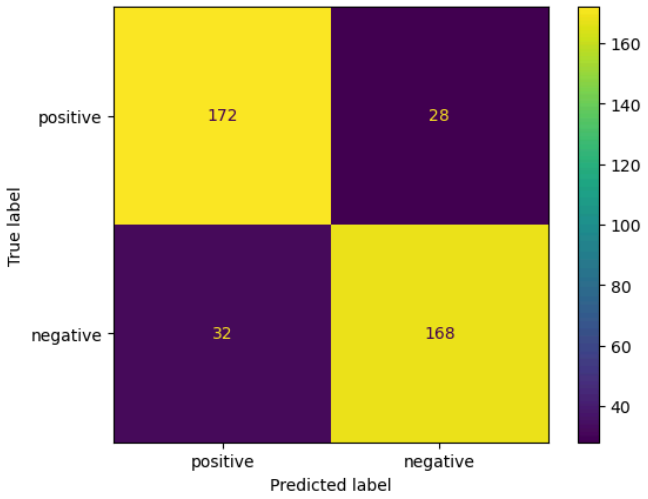
    The Naive Bayes model achieved an accuracy of 81%
    on the test set.



- Support Vector Machine (SVM):
    - Methodology:

      We utilized the SVC classifier from scikit-learn with a linear kernel and regularization parameter C=1.0. Similarly, we trained the SVM model on the TF-IDF transformed data (X_train, y_train) and evaluated its performance.
    - Results:

      The SVM model outperformed Naive Bayes, achieving an accuracy of 84% on the test set. We used classification_report to provide a detailed performance breakdown.



Conclusion:

Both Naive Bayes and SVM proved effective in classifying text data. However, SVM demonstrated superior performance, likely due to its ability to find optimal hyperplanes in high-dimensional space.