Sentiment Analysis Implementation and Experimentation

Introduction

This report describes the implementation of various machine learning models for sentiment analysis on textual data. The datasets used here are taken from Kaggle. The datasets (both training and testing), undergo necessary preprocessing before model experimentation.

Dataset Info-

Content

The dataset is stored in a CSV file with six fields, and emoticons have been removed to prevent label leakage during model training. It captures data from tweets made over a certain time period, primarily focusing on the sentiment labels associated with each tweet. It has a total of 2000 entries out of which 228 can be considered null values. Below is a breakdown of the columns in the dataset:

- **Polarity**: Indicates the sentiment of each tweet:
  - Negative sentiment
  - Neutral sentiment
  - Positive sentiment
- **Tweet ID**: A unique identifier for each tweet.
- **Date**: The date and time when the tweet was posted, in UTC format.
- **Query**: A term associated with the tweet; if there is no query, this field contains "NO_QUERY."
- **User**: The username of the individual who posted the tweet.
- **Text**: The actual content of the tweet.

---

1. Data Preprocessing and Visualization

1.1 Library Imports

The libraries included are pandas, numpy, matplotlib.pylot, seaborn, warnings, nltk, re, emoji, sklearn.

1.2 Dataset Loading

The training and testing datasets are loaded, and the 'Id' column is dropped, as it does not provide useful information for sentiment analysis. Removing redundant columns helps simplifying the dataset, focusing only on relevant features.

1.3 Text Preprocessing

Natural language processing (NLP) libraries like nltk and emoji are imported to prepare the text data. The preprocessing steps include:

1. **Emoji Conversion**: Emojis are converted into descriptive text to ensure meaningful representation, as models often struggle with emojis.
2. **Text Normalization**: Text is lowercased for uniformity, and contractions are expanded (e.g., "can't" to "cannot") to avoid inconsistencies.
3. **Removal of URLs, Usernames, and Special Characters**: These are filtered out to reduce noise in the data.
4. **Stopword Removal**: Common English words that do not add significant meaning (e.g., "the", "is") are removed to reduce dataset complexity.
5. **Lemmatization**: Words are reduced to their base forms (e.g., "running" becomes "run"), which helps the model generalize across different word forms.

This preprocessing step aims to transform raw text into a clean, structured format that retains only the most meaningful information.

1.4 Data Visualization

To understand the distribution of sentiment types, a count plot is created. This visualization is essential for detecting any class imbalances, as highly imbalanced datasets can affect model performance.

2.Feature Engineering

2.1 TF-IDF Vectorizer

The TfidfVectorizer is a tool that converts a collection of text documents into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features. This is useful in natural language processing (NLP) tasks to transform text into numerical features suitable for machine learning models.

By default, it removes stop words, tokenizes text, normalizes words to lowercase, and computes TF-IDF weights for words in the text corpus.

Fit: It learns the vocabulary (unique words) and computes IDF (Inverse Document Frequency) values from the text data.

Transform: It converts the text into a sparse matrix where each row represents a document, and each column represents a term. The values in the matrix are the TF-IDF weights for the terms.

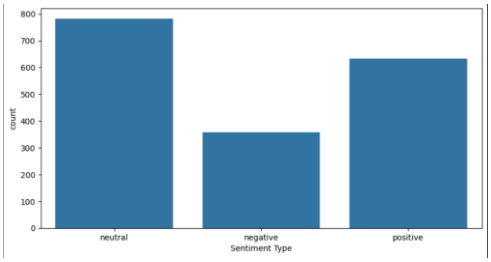
.

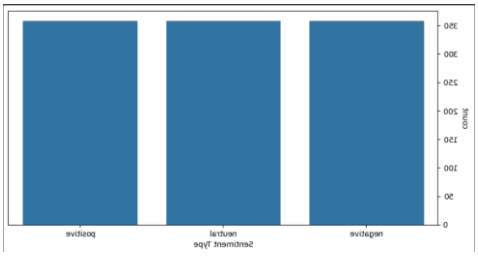
---

3. Model Implementation and Evaluation

Dataset Balancing

Before training, the dataset was observed to have uneven sentiment distribution, with one or more sentiment classes underrepresented. To prevent biased learning and ensure fair performance across all sentiment categories, the dataset was balanced by sampling so that negative, neutral, and positive classes contributed equally. This balancing step improves model generalization and avoids overfitting to the majority class**.**





Several machine learning models are implemented and tested to identify the best approach for sentiment classification. Here's a brief description of each model:

3.1 Logistic Regression

Logistic Regression is a linear model often used for binary classification tasks. In this case, it's applied to classify sentiment types. Logistic Regression estimates the probability of each class and assigns the label with the highest probability. It serves as a baseline due to its simplicity and interpretability.

3.2 K-Nearest Neighbors (KNN)

The K-Nearest Neighbors algorithm classifies a data point based on the labels of its nearest neighbors. The value of k determines how many neighbors influence the classification decision. KNN is non-parametric and simple, but it can be computationally expensive, especially for large datasets.

3.3 Decision Tree Classifier

A Decision Tree is a hierarchical model that splits the data into branches based on feature values, leading to a final decision at each leaf node. This model is intuitive and interpretable, as it mirrors human decision-making. However, it can overfit the data if not carefully tuned.

3.4 Support Vector Machine (SVM)

Support Vector Machine finds a hyperplane that best separates the classes in high-dimensional space. SVM is effective for datasets with many features and is known for its robustness against overfitting, especially when using the kernel trick for non-linearly separable data. Here, it's applied to assess how well it can classify sentiment types based on text features.

3.5 Perceptron

The Perceptron is a basic neural network model used for binary classification. It adjusts weights iteratively based on errors made during training, using a straightforward update rule. While simple, it may struggle with complex patterns without added layers or non-linear transformations.

5. Experimentation and Results

The models are trained on the processed training data and evaluated based on accuracy.

- **Logistic Regression** provides a baseline with straightforward interpretability.
- **KNN** is intuitive but sensitive to feature scaling and computationally intensive.
- **Decision Tree** offers flexibility and interpretability but may overfit without regularization.
- **SVM** shows robustness with high-dimensional data and provides good generalization.

- **Perceptron** demonstrates how neural networks can be applied in a simple form but may be limited without layers.

MODEL TRAINED	ACCURACY
Logistic Regression	0.8515801354401806
KNN	0.4554176072234763
Decision Trees	0.9018058690744921
SVM	0.8623024830699775
Perceptron	0.8572234762979684