Sentiment Analysis Project Documentation

Overview

This project focuses on building a sentiment analysis model to classify text into one of four sentiment categories: **Positive**, **Negative**, **Neutral**, and **Irrelevant**. The goal is to analyze text data, such as tweets or comments, and predict the sentiment expressed in the text. The project uses the **Multinomial Naive Bayes** algorithm with two text vectorization techniques: **Bag of Words (BoW)** and **Term Frequency-Inverse Document Frequency (TF-IDF)**. The model is trained on a dataset containing 74,681 text samples, each labeled with its corresponding sentiment.

Dataset

The dataset used in this project is stored in a CSV file named twitter_training.csv. It contains two main columns:

- **Message**: The text samples (e.g., tweets or comments).
- **Output**: The sentiment label for each text sample (Positive, Negative, Neutral, or Irrelevant).

Dataset Statistics

- **Total Samples**: 74,681
- **Sentiment Distribution**:
    - Negative: 22,542
    - Positive: 20,831
    - Neutral: 18,318
    - Irrelevant: 12,990
- **Missing Values**: 686 (in the Message column, which were removed during preprocessing).

Project Workflow

1. Importing Libraries

The project uses the following libraries:

- **Pandas**: For data manipulation and analysis.
- **NumPy**: For numerical operations.
- **Scikit-learn**: For machine learning tasks, including model training, evaluation, and text vectorization.
- **NLTK**: For natural language processing tasks, such as stopword removal and lemmatization.

2. Loading the Dataset

The dataset is loaded into a Pandas DataFrame, and the first few rows are inspected to understand its structure.

3. Data Preprocessing

- **Column Renaming**: The dataset columns are renamed for clarity.
- **Text Cleaning**:
    - Convert text to lowercase.
    - Remove special characters and short words (1-2 letters).
    - Remove stopwords (e.g., "the", "and", "is") using NLTK.
    - Apply lemmatization to reduce words to their base form.
- **Handling Missing Values**: Rows with missing values in the Message column are dropped.

4. Splitting the Dataset

The dataset is split into training and testing sets using an 80-20 split:

- **Training Data**: 70% of the dataset.
- **Testing Data**: 30% of the dataset.

5. Text Vectorization

Two text vectorization techniques are used to convert text data into numerical format:

1. **Bag of Words (BoW)**: Converts text into a matrix of token counts.
2. **Term Frequency-Inverse Document Frequency (TF-IDF)**: Converts text into a matrix of TF-IDF features, with n-grams (1-3).

6. Model Building

The **Multinomial Naive Bayes** algorithm is used for classification. Two models are trained:

- **Model 1**: Using BoW vectorization.
- **Model 2**: Using TF-IDF vectorization.

7. Model Training

Both models are trained on the vectorized training data.

8. Model Evaluation

The models are evaluated on the testing data using **accuracy score** as the metric:

- **BoW Model Accuracy**: ~72.96%
- **TF-IDF Model Accuracy**: ~82.95%

9. Making Predictions

The trained models are used to predict the sentiment of new text inputs. For example:

- Input: "This product is absolutely terrible, complete waste of money!"

- Predicted Sentiment: Negative

Results

- The TF-IDF model outperformed the BoW model, achieving an accuracy of approximately 82.95% compared to 72.96% for the BoW model.
- The models are capable of accurately predicting the sentiment of text inputs across all four sentiment categories.

Conclusion

This project successfully demonstrates the use of machine learning for sentiment analysis. The Multinomial Naive Bayes algorithm, combined with text vectorization techniques like BoW and TF-IDF, proves to be effective for this task. The TF-IDF model, in particular, provides better accuracy, likely due to its ability to capture the importance of words in the context of the entire dataset.

Tools and Libraries Used

- **Pandas**: Data manipulation and analysis.
- **NumPy**: Numerical operations.
- **Scikit-learn**: Machine learning tasks (text vectorization, model training, evaluation).
- **NLTK**: Natural language processing tasks (stopword removal, lemmatization).
