Sentiment Analysis Project Report

Introduction

Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) task that involves determining the sentiment or emotional tone expressed in a piece of text. It is widely used in various applications, including social media monitoring, customer reviews, and market research. In this project, I have built a sentiment analysis model using Python and NLTK (Natural Language Toolkit).

Dataset

We start by collecting a labelled dataset containing text samples and their corresponding sentiment labels. The dataset can be obtained from various sources, such as customer reviews, social media posts, or news articles. For this project, I have taken the dataset from Kaggle platform. Its file is available in the repository. The training and validation files were available separately. The main two columns were the text data and the sentiment associated with it. The dataset had 4 relevant sentiments for classification: Positive, Negative, Neutral and Irrelevant. In this project, I have considered irrelevant sentiment to be a part of the neutral class.

Preprocessing

Before training the model, we need to preprocess the text data. The preprocessing steps include:

1. **Lowercasing**: Convert all text to lowercase to ensure consistent representation.
2. **Tokenization**: Split the text into individual words (tokens).
3. **Stop Word Removal**: Remove common stop words (e.g., “the,” “and,” “is”) that do not carry significant meaning.
4. **Stemming/Lemmatization**: Reduce words to their root form (e.g., “running” becomes “run”).

Feature Extraction

To train the sentiment analysis model, I needed to convert the text data into a numerical representation. We use the bag-of-words model, where each document (text sample) is represented as a vector of word frequencies. The features are the unique words (or tokens) present in the entire dataset.

Model Selection

For this project, I have used the Naive Bayes classifier as our sentiment analysis model. Naive Bayes is a probabilistic classifier that works well for text classification tasks. Other models like LSTM (Long Short-Term Memory) networks or deep learning architectures can also be used, but Naive Bayes is simple and effective.

Training

1. **Splitting the Dataset**: Divide the dataset into training and testing sets (e.g., 80% for training, 20% for testing). This was done for the first classifier. For using the 2nd classifier, no separate test set was created. The data of the csv file was used for training. The validation dataset was then used for prediction.
2. **Feature Extraction**: Create feature vectors for the training data using the bag-of-words representation.
3. **Training the Model**: Train the Naive Bayes classifier using the training data.

Evaluation

1. **Testing**: Use the trained model to predict sentiment labels for the testing data.
2. **Accuracy**: Calculate the accuracy of the model by comparing predicted labels with actual labels.

Validation Dataset

On using the validation dataset through the classifiers, I got the accuracy of the trained model. For the 1st classifier, we have an accuracy of 0.79 or 79%.

For the 2nd classifier, wherein the training has been done on the entire training dataset, we have an accuracy of 0.82 or 82%.

I feel these are fairly good results which we have obtained on this dataset.

Suppose we have a new text sample: “The movie was excellent!” We preprocess it, extract features, and predict its sentiment label using the trained Naive Bayes classifier.

Conclusion

Sentiment analysis is a powerful tool for understanding public opinion and customer feedback. By building and evaluating a sentiment analysis model, we can gain insights into how people feel about specific topics or products.