Sentiment Analysis

By Sarah Sheikh

Introduction

Movie reviews are an important way to gauge the performance of a movie. While providing a numerical/stars rating to a movie tells us about the success or failure of a movie quantitatively, a collection of movie reviews is what gives us a deeper qualitative insight on different aspects of the movie. A textual movie review tells us about the strong and weak points of the movie and deeper analysis of a movie review can tell us if the movie in general meets the expectations of the reviewer.

Sentiment Analysis is a major subject in machine learning which aims to extract subjective information from the textual reviews. In this project we aim to use Sentiment Analysis on a set of movie reviews given by reviewers and try to understand what their overall reaction to the movie was, i.e. if they liked the movie or they hated it. We aim to utilize the relationships of the words in the review to predict the overall polarity of the review.

Dataset

The dataset used for this task was downloaded from the link below:
https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

IMDB dataset having 50K movie reviews for natural language processing or Text analytics.
This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing.

Steps involved

- Loading Dataset
- Preprocessing
- Feature Engineering
- Model selection
- Tuning and Validation
- Metrics

Loading Dataset

We are using the IMDB dataset which contains reviews on movies and corresponding sentiment either in positive or negative.

IMDB dataset having 50K movie reviews for natural language processing or Text analytics.
This is a dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. We provide a set of 25,000 highly polar movie reviews for training and 25,000 for testing.

Reviews : (str) Data
Sentiment : (str) Target

Preprocessing

Initially to process the data we load the data from respective folders which is encoded in utf8
Using the function **get_data()** which loads all data to the list and adds 1 for positive label and 0 for negative label data for every review record.
In Order to process the given sentence we need to preprocess certain special characters which add redundancy to the text using the function **clean_text()** .
We use regular expressions to filter out the special characters and numerical data to clean the text.

Feature Engineering

The data containing the text is then spitted word by word and for every unique word a unique ID which is the length of the dictionary is assigned for achieving the n-dimensions of the given corpus for building the feature matrix . This functionality is achieved for train and test data and the final dictionary of all unique words in the corpus are stored as key-value pairs using **get_matrix_of_x().** The dictionary is used for creating a feature matrix with the number of rows the same as the train and test data and the dimensions with the number of unique words in the corpus dictionary. The terms count are then updated for every word occurrence in that particular review. This technique in traditional machine learning is called Bag-of-Words.

Model Selection

For training the feature matrix with the matrix data as X with n-dimensions where n is the number of unique word-counts. For training we are using the Multinomial Naive Bayes algorithm from sklearn package. The Multinomial Naive Bayes algorithm is a Bayesian learning approach popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as a movie review in our case, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the greatest chance.

Because the Naive Bayes theorem is based on the Bayes theorem, it is necessary to first comprehend the Bayes theorem notion. The Bayes theorem, which was developed by Thomas Bayes, estimates the likelihood of occurrence based on prior knowledge of the event's conditions. When predictor B itself is available, we calculate the likelihood of class A. It's based on the formula below: $P(A|B) = P(A) * P(B|A)/P(B)$.

It is simple to implement because all you have to do is calculate probability. This approach works with both continuous and discrete data. It's straightforward and can be used to forecast real-time applications. It's very scalable and can handle enormous datasets with ease.

Tuning and Validation

Since the Multinomial NaiveBayes has a tunable parameter alpha which initially is set to 1. We select a range of five values from 0.01 to 10 with a multiplicative step size of 10.
For every parameter selection the model is initialized and is validated using k-fold cross validation with k=10. At the end of every validation, the mean accuracy percentage is calculated and stored. Finally, we select the model with the maximum mean average accuracy for final training of the data.

Metrics

Finally, the accuracy is calculated using the best configuration from the validated data. We calculate the accuracy by comparing the predicted values with the ground truth.

The train accuracy of the sentiment analysis model is 0.9082142857142858 %
The test accuracy of the sentiment analysis model is 0.8536111111111111 %