UNIVERSITÀ DI PAVIA

Machine Learning Course

Movie Reviews: sentiment analysis

Andrea Alberti

Department of Computer Engineering - Data Science

University of Pavia, Italy
Email: andrea.alberti01@universitadipavia.it

February 25, 2024

Abstract

This project aims to develop a Multinomial Naive Bayesian classifier that can accurately predict the sentiment of movie reviews based on their textual content. The available data comprises a collection of labeled movie reviews classified as positive or negative. The study explores various versions, of the Multinomial Naive Bayesian model including different vocabulary sizes, to evaluate the classifier's performance. Additionally, the project investigates the use of logistic regression as an alternative approach for predicting the reviews' sentiment. The analysis of the results demonstrates that despite its simplicity the Multinomial Naive Bayesian classifier is a robust model for sentiment analysis tasks. The logistic regression model also achieves promising results with a small training effort.

Contents

1 Introduction

Text classification is used in many areas, such as spam filtering and document sorting. Sentiment analysis is a fascinating application that involves predicting the emotions of writers, such as anger, happiness, and sadness. It is useful for analyzing people's opinions on products, books, TV shows, and political parties.

1.1 Available Data

The dataset presented in the paper "Learning Word Vectors for Sentiment Analysis" by Andrew L. Maas et al. will be used. It contains 50,000 reviews that are equally distributed across two classes. The dataset divided into a training set of 25,000 reviews, a validation set of 12,500 reviews, and a test set of 12,500 reviews.

1.2 Goal

The goal of this project is to fit a Multinomial Naive Bayes classifier to the training set and use it to predict whether a new unseen review is positive or negative.

1.3 Naive Bayesian Classifier

The Multinomial Naive Bayes classifier is a simple probabilistic generative classifier based on applying Bayes' theorem with strong independence assumptions between the features. In particular the features are integers values and in this case are represented by the Bag of World representation (discussed later) of each review. Since the model is a generative one it works by estimating the probability of each feature given the class and then using Bayes' theorem to compute the probability of the class given the feature vector.

Formal description

Here is described the model in a more formal way:

1. The goal is to find $P(Y|X)$ which is the probability of a class (Y = y) given the feature vector (X = x). This is compute using Bayes' theorem.

$$\max_y P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

2. Introducing the Naive independence assumption assuming $P(X|Y)$ follows a multinomial distribution where $\pi_{y,j}$ is the probability of the j-th feature given the class y $P(X_j|Y = y)$ and x_j is the number of times the j-th feature appears in the document.

$$P(X|Y = y) = \frac{(x_1+x_2+...+x_n)!}{x_1!x_2!...x_n!} \prod_{j=0}^{n-1} \pi_{y,j}^{x_j}$$

3. Inserting 2 inside 1 and applying logarithm we obtain:

$$P(Y = y|X) = \sum_{j=0}^{n-1} x_j \log \pi_{y,j} + \log P(Y)$$

and

$$\hat{y} = arg\max_y \sum_{j=0}^{n-1} x_j \log \pi_{y,j} + \log P(Y)$$

Basically we classify to the class (Y = y) maximizing the function in point 3. The parameters of the model are the $\pi_{y,j}$ and the $P(Y)$ which are estimated simply computing the relative frequencies of the features and the classes in the training set. For example:

$$\pi_{y,j} = \frac{N_{X_j,Y=y}}{N_{Y=y}}$$

2 Model Building

To use the Multinomial Bayesian Classifier for text classification it is necessary to perform 3 steps. First of all a vocabulary containing the words that will be used to represent the reviews must be built. Then the features must be extracted from the reviews, creating the BoW representation of each review. Finally the classifier must be trained.

2.1 Build a vocabulary

The vocabulary is built taking the most frequent words in the training set. Since the number of words taken can influence the performance of the model, the vocabulary size is varied in the "Changing Vocabulary Size" section and the results are compared. The vocabulary is created in the following way:

1. The training set is read and the words are tokenized removing punctuation, capital letters and words shorter than 3 characters.

2. The words are counted and the most frequent ones are taken and stored in a txt file.

2.2 Extract Features

The features are extracted from the reviews creating the Bag of Words representation. The features are the words in the vocabulary and the value of each feature is the number of times the word appears in the review.

2.3 Train Classifier

The training phase is just the application of what said in the "Formal description" section. Using the relative frequencies the parameters to learn are:

1. The probability distribution of each word inside each class $\pi_{y,j}$.

2. The probability distribution for the class $P(Y = y)$.

Model performance
A first result of the train and test accuracies of the model, together with some specifications are reported in table 1:

Table 1: Base Model performance

Version	**Voc_size**	**Train_acc**	**Test_acc**
Base	1000	0.820	0.816

The model has already a good performance since the accuracy is above 80%. In particular the model is neither overfitting nor underfitting since the train and test accuracy are quite similar. However the model can be improved and in the next section some variants are proposed.

2.4 Variants

Excluding common words
To add this modification to the base version we need first of all to have a list of the words we want to exclude. The list is in the file *'stopwords.txt'*. A new vocabulary excluding the words in this latter is created and its size can be freely chose. At this point the BoW must be adapted to the new vocabulary and all is set. The model can now be trained and tested.

Introducing stemming
The functions have been designed to support the optional *stemming* argument. If set to *True* the functions use the Porter Stemmer algorithm to stem the words read from the files. Once again a new vocabulary is built and the BoW is adapted to the new vocabulary.

Comparison
In the table 2 is reported a comparison between the base model and the two variants. A more detailed comparison is done at the end of the report, in the "Changing Vocabulary size" section where also the training set and the vocabulary size are varied.

Table 2: Model performance comparison

Version	**Voc_size**	**Train_acc**	**Test_acc**
Base	1000	0.820	0.816
Stopwords	1000	0.829	0.826
Stemming	1000	0.823	0.817
Hybrid	1000	0.826	0.820

The results show that the model with the stopwords excluded performs better than both, the base model and the one with stemming with the specifications reported in the table 2. In the last row there are the results of the model with both stemming and stopwords excluded. The results are not better than the one with the stopwords excluded.

3 Model Analysis

In this section is done a comparison among the four models above proposed. The best model is chosen and analyzed in order to understand the most impactful words on its prediction and the worst errors it makes.

3.1 Changing Vocabulary Size

The vocabulary size is varied in order to see how the performance of the various models is affected. The results are reported in figure 1 and figure 2.

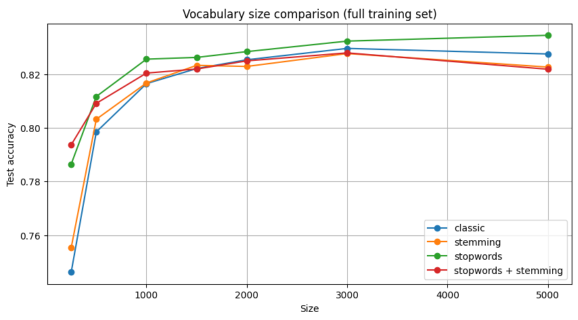

Figure 1: Test accuracies comparison

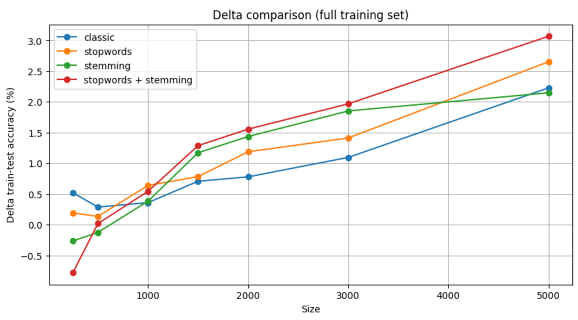

Figure 2: Delta error comparison

As expected the accuracy of the model increases with the vocabulary size. However the increase is not linear and after a certain size the accuracy stays almost constant. In the figure 2 is reported the delta between the training accuracy and the test accuracy as a function of the vocabulary size. For all the model the delta increases with the growing of the vocabulary, suggesting that the model is overfitting. This is reasonable because the number of parameters to learn increases with the vocabulary size. Some model, like the red one are more prone to overfitting. The model that doesn't consider stopwords is the best, probably because because it excludes words (e.g. 'the') that don't provide any information about the sentiment of the review. From now on, the model considered is the one that excludes the stopwords with a vocabulary size of 1000 words. This is a good compromise between computational cost and model performance, since the test accuracy is high and the delta is almost the lowest among the models.

3.2 Analysis

Most impactful words
The most impactful words for a class are those that most influence the classification score in favour of that class. They are taken looking at the probability gap of each word between the two classes. Results are reported in table 3 and table 4.

Table 3: Positive Class

Voc size	Word	Score
1000	superb	1.737
1000	stewart	1.643
1000	wonderful	1.586
1000	fantastic	1.530
1000	excellent	1.485
1000	amazing	1.415
1000	powerful	1.326
1000	favorite	1.289
1000	perfect	1.268
1000	brilliant	1.251

Table 4: Negative Class

Voc size	Word	Score
1000	waste	-2.601
1000	pointless	-2.435
1000	worst	-2.268
1000	awful	-2.208
1000	poorly	-2.201
1000	lame	-1.962
1000	horrible	-1.891
1000	pathetic	-1.880
1000	wasted	-1.818
1000	crap	-1.816

Worst errors for the chosen model.
The worst errors produced by the model are the data misclassified with the higher confidence. In other words, are the reviews for which the gap between the classification scores of the two classes is the largest. They are shown in figure 3 and figure 4.

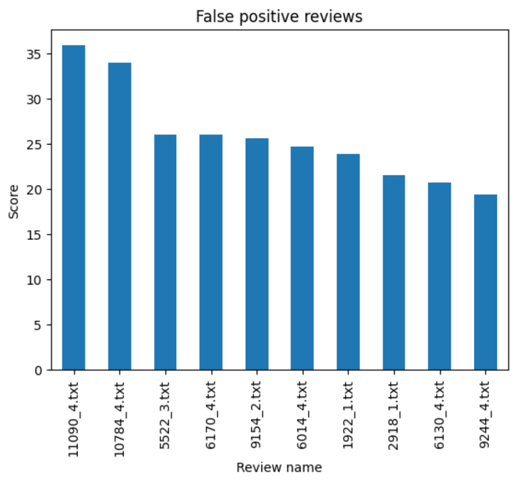

Figure 3: False positives

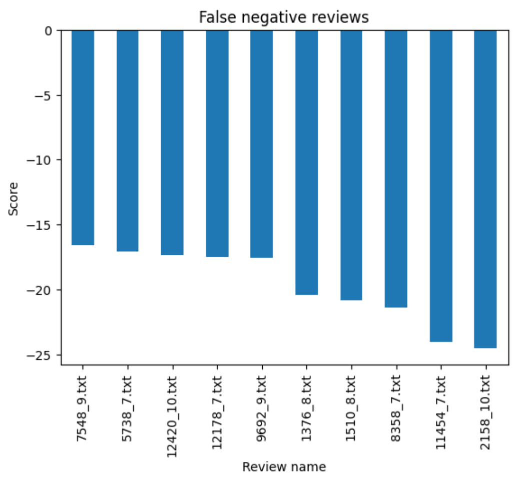

Figure 4: False negatives

4 Logistic Regression

For completeness the movie review problem is also faced using the Logistic Regression model.

4.1 Results

The model is a Logistic Regression without regularization, fitted on the stopwords version of the problem. The hyper parameters used together with the results are written in table 5. In general the results are quite good similarly to the ones obtained with the Naive Bayes model. In figure 5 and figure 6 are shown the worst errors by logistic regression.

Table 5: Hyper parameters

Tol.	**lr**	**Train_acc**	**Test_acc**
0.0001	0.0023	0.867	0.854

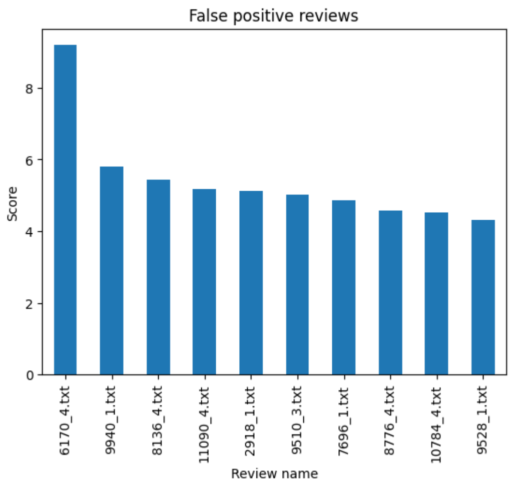

Figure 5: False positives

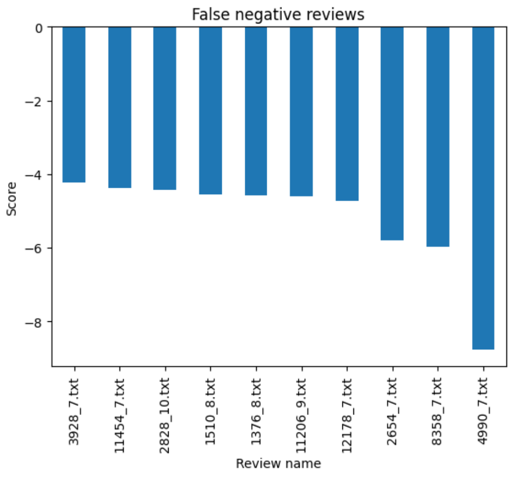

Figure 6: False negatives