Sentiment Analysis of IMDb Movie Reviews Using Naive Bayes

Noah Antisseril, Praneetha Bhogi, Madhav Parthasarathy, Khushi Patel

CS 410 Final Report, December 9, 2024

ABSTRACT

In this project, we will be applying sentiment analysis techniques to classify IMDb movie reviews as positive or negative. Using the Naive Bayes classifier, we aim to preprocess the dataset, use specific vectorization techniques, and implement n-gram representations and tuning parameters to overall improve the accuracy and precision of our model.

KEYWORDS

Sentiment analysis, IMDb, Naive Bayes, n-grams, TF-IDF

1 INTRODUCTION

Sentiment analysis is a powerful tool for interpreting opinions within large volumes of text, making it particularly useful in fields where large amounts of user-generated content is produced. As the prevalence of this large scale content creation increases, it is important to fine-tune sentiment analysis techniques in order to gain valuable insights from data. In this project, we aim to apply sentiment analysis to a dataset of IMDb movie reviews. The classifier will categorize IMDb movie reviews as either positive or negative based on the content of the text. We will use this project as a way to explore sentiment analysis, as well as apply the techniques learned in CS 410.

2 SUMMARY

In this project we implemented a Naive Bayes classifier to mark movie reviews as either positive or negative. First, the dataset will be preprocessed to clean the reviews by removing stop words (common words like "the" or "and"), punctuation, and other noise that could affect the model's accuracy. Next, to convert the processed text into numerical data, we will apply one of the vectorization techniques discussed in class so far: Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), or Word2Vec. These techniques essentially transform the text into a more processable structure so it can be read by the machine learning model. Leveraging these techniques, we were able to implement multiple different kinds of Naive Bayes Classifiers by using Unigram, Bigram, IDF, and Mixture techniques. With these classifiers, we predicted the sentiment of different IMDb movies reviews and compared the accuracies.

3 DESCRIPTION

Dataset:
The dataset is a Kaggle dataset containing roughly 50,000 movie reviews taken from the website IMDb. Please find the dataset linked here. The dataset has each review labeled as positive or negative with respect to their sentiment. We will use the Pandas library in Python to load these reviews into a dataframe. We will transform the dataset by converting the text to lowercase, removing stop words and punctuation marks, and applying a vectorization technique discussed in class. The vectorization technique will be either Bag of Words, TF-IDF, or Word2Vec. Then, based on the technique (IDF, n-gram, Mixture), we will transform the text into probabilities using Laplace smoothing, probabilities using inverse document frequency, or a combination of these two techniques.

4 METHODOLOGY

We will implement from scratch a Naive Bayes classifier, and this will be trained on the vectorized dataset. Initially, a unigram model will be used to gauge the model's baseline performance. Then we will implement a bigram model in order to capture more complex relationships between words. Additionally, we will also implement a mixture classifier that utilizes both bigram and IDF. Then, we can use n-grams to check their impact on the classifier's accuracy. The model's performance will be measured using precision and F1-Score.

5 SCHEDULE

- **Week 1-2:** Data exploration and cleaning
- **Week 3:** Apply Bag of Words vectorization
- **Week 4:** Naive Bayes classifier implementation using BoW
- **Week 5:** Baseline model performance evaluation
- **Week 6:** Implement n-gram models
- **Week 7:** Hyperparameter tuning
- **Week 8:** TF-IDF vectorization
- **Week 9:** Word2Vec vectorization
- **Week 10:** Model tuning and cross-validation
- **Week 11:** Performance reporting and comparison
- **Week 12:** Final report and presentation

6 EVALUATION

We have coded a Naive Bayes class that initializes with stop words as well as a lemmatizer (reduces words to root form) to aid with preprocessing the IMDB reviews. We then split the data (80/20) into a training and testing set. Next, we have a preprocessing function that converts reviews to lowercase, and the function also removes URLs, stopwords, and punctuation. Then, we train the Naive Bayes model by building word counts for the positive and negative classes, so that we can calculate probabilities. Then, we compute the log probability given a review to determine whether the review is positive or negative. We will repeat this process for the different Naive Bayes models we have created (unigram, bigram, IDF, bigram-IDF).

Data Preparation and Preprocessing
We leveraged the IMDB reviews dataset, which includes text data that is labeled as either positive or negative for each review. We also split the data into training and testing sets using an 80/20 ratio, allowing us to train the model on most of the given data while preserving aspects of it to test and evaluate the model's performance. Our preprocessing function addresses several parameters to ensure that our data is standardized throughout:

1) Lowercasing all of the text to eliminate case sensitivity
2) Removing URLS to preserve the most useful aspects of the data
3) Removing common stopwords to reduce noise
4) Removing punctuation and numbers to focus on the actual textual content

Model Training
After preprocessing our data, we implemented a training function for our Naive Bayes model. We iterate through the training subset, identifying the sentiment label associated with each word: if a word is labeled as 1, we update the positive word count; if it is labeled as 0, we update the negative word count. These counts are essential for computing the probability of the word appearing in each class to make informed predictions for future reviews.
The Naive Bayes function computes the log probabilities of positive and negative sentiments for a given review. For each word in the text, it retrieves the count of each word in both classes, applies Laplace smoothing to avoid zero probabilities for unseen words, calculates smoothed probabilities, and adds the log probabilities to the respective totals. Comparing the cumulative scores reveals the sentiment analysis of the review: if the positive probability is greater than the negative probability, the review is classified as positive and the function returns 1; otherwise, it is classified as negative and returns 0.

6.1 Final Evaluation

The methods above incorporate the word counts from the positive and negative categories from the training process to accurately classify new reviews. To evaluate the model, we test it by calculating the accuracy, precision, recall, and confusion matrix associated with the model's performance.

6.2 Confusion Matrices and Metrics

The confusion matrices show the performance of our models under different configurations:

```
Accuracy: 0.8596
Precision: 0.8723622208563819
Recall: 0.8450089303433221
Confusion Matrix:
 [[4338  623]
 [ 781 4258]]
```

Figure 1: Confusion matrix and metrics for the Unigram model.

```
Accuracy: 0.8598
Precision: 0.8715015321756895
Recall: 0.8465965469339155
Confusion Matrix:
 [[4332  629]
 [ 773 4266]]
```

Figure 2: Confusion matrix and metrics for the TF-IDF-based model.

```
Accuracy: 0.8822
Precision: 0.8976313079299691
Recall: 0.8648541377257393
Confusion Matrix:
 [[4464  497]
 [ 681 4358]]
```

Figure 3: Confusion matrix and metrics for the Bigram model.

```
Accuracy: 0.8826
Precision: 0.8944682588283324
Recall: 0.8696169874975194
Confusion Matrix:
 [[4444  517]
 [ 657 4382]]
```

Figure 4: Confusion matrix and metrics for the Bigram + TF-IDF mixed model.

6.3 Comparison of Model Accuracy

To compare the performance of the models across different metrics, we visualized the results using bar charts:

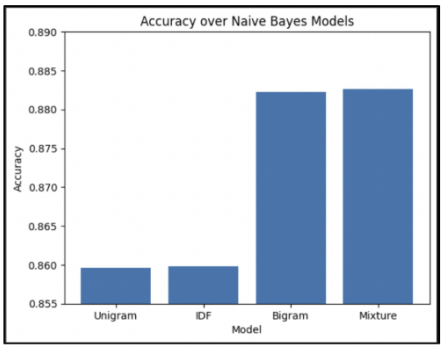

Figure 5: Accuracy comparison of Naive Bayes models (Unigram, TF-IDF, Bigram, Mixed).

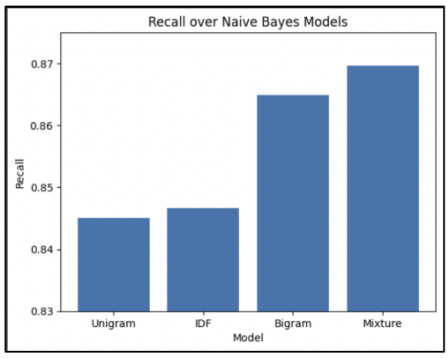

Figure 6: Recall comparison of Naive Bayes models.

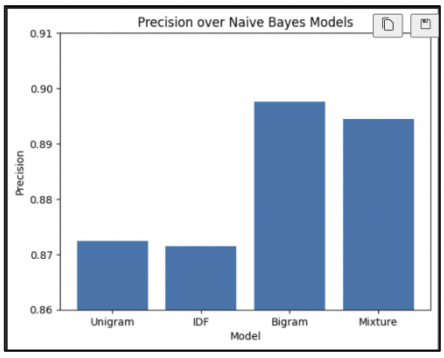

Figure 7: Precision comparison of Naive Bayes models.

7 DISCUSSION

For our statistical analysis, we decided to output the accuracy, precision, recall, as well as the confusion matrix to determine how our models are performing with Naive Bayesian Classification. In the first images, we output the accuracy, precision, recall, and confusion matrix values for each classifier. For the original Naive Bayes classifier, we saw an accuracy of 0.8596, which is a relatively high value for classification. However, by leveraging the other techniques, we hoped to increase the accuracy even more. This can be seen with the Bigram Naive Bayes and Bigram IDF Naive Bayes classifiers. Both of these accuracies were above 0.88, but on the other hand, one of the other methods (IDF Naive Bayes), kept the accuracy at the same value as the original. The reason why these accuracies might be the same is due to the removal of stopwords, which would lower the amount of common words seen, meaning that there will be a smaller effect on IDF. Additionally, IDF may be performing a redundant form of data normalization that Naive Bayes already takes care of. Therefore, in the future, it would be useful to explore other techniques such as feature selection with chi-square tests for example. In the graphs, there is a clear trend in that Bigram and the Mixture models perform better than the Unigram and IDF models. One comparison to make is the difference between the Bigram and Mixture model precision/recall values. Precision is the proportion of correctly identified positive predictions over the amount of positive predictions made by the model. Recall is the proportion of positives identified over the total amount of existing positives. Thus, in relation to the graphs, the Bigram model was better with retrieving true positives, while the Mixture model was better with retrieving positives in general with a higher false positive rate. The reason for this could be because in a pure Bigram model, there is a large emphasis on exact matches with bigrams, while with the Mixture model, there is an inclusivity of rare bigrams, leading to more false positives. Through our analysis, we determine that the purely Bigram model would be best for retrieving the most accurate sentiment analysis of IMDb reviews, as the accuracy and precision are highest.