Fake News Detection using Naive Bayes

Divam Manchanda 2022ME21336
Yash Bansal 2022CS51133

October 15, 2024

1 Objective

In this assignment, we aim to implement the Naive Bayes algorithm to detect fake news using the LIAR dataset. LIAR is a multi-class classification dataset with labels indicating the truthfulness of statements, categorized into six fine-grained classes: *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true*.

2 Model

The Naive Bayes classifier assumes that features are independent of each other given the class label, which makes it a simple but effective model for text classification. We experimented with various natural language processing techniques and feature sets to improve the classifier's performance on the dataset.

First, we incorporated the data given in the training file such as **subjects**, **speaker**, and **affiliation** with different weights assigned to each feature. After testing with different weights, the following set of weights produced the best results:

- **subjects**: 8
- **speaker**: 10
- **job**: 1
- **state**: 1
- **party**: 1

We also experimented with trigrams, but this caused the model to overfit the training data, resulting in poor performance on the testing data. As a result, we limited our word sets to unigrams and bigrams.

For smoothing, we used Laplace smoothing, and after tuning the hyperparameter, we found that a smoothing factor of $\alpha = 25$ gave the best results.

3 Tools and Libraries

The following tools and Python libraries were used for implementing the Naive Bayes classifier:

- **Python 3.10**
- **NLTK**: For text preprocessing tasks like tokenization and stemming.
- **pandas**: For data manipulation and handling.
- **NumPy**: For efficient array operations.

4 Data Preparation

The dataset contained various columns such as subjects, speakers, and other metadata, which were incorporated into the model with different weights assigned to each feature. Text preprocessing was a critical part of data preparation, and the steps we followed included:

- **Removing punctuations**: We removed all punctuations and other non-alphanumeric characters from the news texts and split the text on whitespaces.
- **Stemming and Lemmatization**: We tried different natural language processing techniques such as stemming and lemmatization. After testing various methods, we found that using the *PorterStemmer* yielded the best results, so it was used for the final implementation.
- **Speaker Data**: The speaker counts available in columns 9-13 of the dataset were utilized. Based on this, we eliminated some classification possibilities for specific speakers to refine our model.

5 Training Method

The Naive Bayes model was trained using the prepared dataset. The primary steps involved in training were:

- The data was tokenized and transformed into feature vectors using both unigrams and bigrams.
- Weights were assigned to different features like subjects, speaker, affiliation, etc., with the best performing set of weights detailed earlier.
- We applied Laplace smoothing, and after tuning, we found that $\alpha = 25$ gave the best results.

During training, we also experimented with other hyperparameters and smoothing factors. However, excessive use of larger n-grams such as trigrams led to overfitting, so they were excluded from the final model.

6 Conclusion

In this assignment, we successfully implemented a Naive Bayes classifier for fake news detection using the LIAR dataset. By experimenting with different feature sets, text preprocessing techniques, and model parameters, we optimized the performance of the classifier. The final model used unigrams and bigrams, Porter stemming, and a Laplace smoothing factor of $\alpha = 25$. The results demonstrate the effectiveness of Naive Bayes for text classification tasks, particularly when combined with proper feature selection and preprocessing techniques.