CS 5420 Project 2: Naive Bayes Bag-of-Words Text Classification

Cooper Wooley

November 2025

1 Introduction

In this project, I will be implementing a Naive Bayes - Bag of Words classifier. The model will classify text documents from the 20 newsgroups text dataset. After evaluating the model, I chose to implement some more modern techniques to improve the accuracy of the model. Accuracy will be the primary evaluation metric for performance, and improvements will focus on underflow and zero-frequency issues in calculations.

2 NB BoW Classification

An NB BoW classifier is a simple, effective, and widely used method for text classification tasks. This approach assumes conditional independence between words given a class and treats each document as an unordered collection of words (bag of words).

2.1 Data Splitting and Cleaning

The 20 newsgroups dataset contains 20,000 documents. These come from 20 different newsgroups (1,000 documents from each group). For the submission of the assignment, we were asked to package our code with the dataset. Because the graders must install the dataset many times because of this, I thought of another solution.

I have packaged my code with the compressed dataset. My program will extract the dataset to a temporary storage location. Upon termination, the program cleans the temporary location.

Since we were tasked with splitting the data in half for training and testing, I took the approach of selecting 500 random documents from each newsgroup and assigning them to train, and the rest to test. A fixed random seed was used to ensure reproducibility. This method makes probability calculations fair

between the newsgroups.

At first, when splitting each document into individual words, I did not want to strip punctuation. I thought that this might improve accuracy by leaving stylistic choices of the individual newsgroups. This, however, failed to hold, and I eventually stripped all words of punctuation.

2.2 Training

When training NB, we must calculate the prior of each class, and the probabilities of each individual word occurring within that class. To calculate these, we use the following:

For class prior:

$$P(Y = c) = \frac{\text{\# docs in c}}{\text{total \# of training docs}}$$

For likelihood:

$$P(\text{word}|c) = \frac{\text{count(word, c)}}{\text{total words in c}}$$

2.3 Predicting

When classifying, or predicting documents we haven't seen before, we can use the class priors and likelihoods we calculated from the training algorithm. To classify, we use:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{\text{length of doc}} P(x_i|y)$$

2.4 Results

From the standard NB BoW classifier, I calculate both the overall accuracy of the model, and each individual class (newsgroups) accuracy.

Overall Accuracy: **0.0465**

Per-Class Accuracy:

Class	Accuracy
alt.atheism	0.0000
comp.graphics	0.0000
comp.os.ms-windows.mist	0.0000
comp.sys.ibm.pc.hardware	0.0000
comp.sys.mac.hardware	**0.0040**
comp.windows.x	**0.0020**
misc.forsale	**0.0040**
rec.autos	**0.0040**
rec.motorcycles	0.0000
rec.sport.baseball	**0.0020**
rec.sport.hockey	**0.9080**
sci.crypt	0.0000
sci.electronics	**0.0060**
sci.med	0.0000
sci.space	0.0000
soc.religion.christian	0.0000
talk.politics.guns	0.0000
talk.politics.mideast	0.0000
talk.politics.misc	0.0000
talk.religion.misc	0.0000

As you can see, the model is not accurate at all. In the next section, I will implement some modern techniques to improve the model.

3 NB BoW Improvements

In the previous section, we saw how poorly the base NB BoW model performs. Here, I will implement Laplace smoothing during likelihood calculations (in the training phase) and utilizing log probabilities (in the testing/classification phase).

3.1 Log Probabilities

Multiplying small probabilities can underflow in computation, but summing their logs keeps the numbers manageable.

Let's say you have a set of probabilities $p_1, p_2, \ldots, p_n$ and you consider their product:

$$P = p_1 \cdot p_2 \cdot \ldots \cdot p_n$$

Now, take the logarithm of P:

$$\log(P) = \log(p_1 \cdot p_2 \cdot \ldots \cdot p_n) = \log(p_1) + \log(p_2) + \ldots + \log(p_n)$$

Therefore, the log of a product of probabilities is equivalent to the sum of the log probabilities. Applying this to NB BoW's classification, we get:

$$\hat{y} = \arg\max_{y} \log P(y) + \sum_{i=1}^{\text{length of doc}} \log P(x_i|y)$$

3.2 Laplace Smoothing

Not all words present in the entire dataset are present in each individual document. Whenever we are predicting a class, if the word is never contained in the class, it's likelihood is 0, which results in taking the prediction probability closer to 0. To account for this, we can use a very common practice in NLP problems: Laplace smoothing.

Very simply, Laplace smoothing just ensures that every word has some probability for every class.

Likelihood with Laplace smoothing:

$$P(\text{word} \mid c) = \frac{\text{count(word, } c) + 1}{\text{total words in } c + \mid \textit{Vocab} \mid}$$

3.3 Results

Overall accuracy: **0.8647**
Difference between accuracies: **0.8182**
As you can see, my new implementation significantly outperforms the base model.