


## Project 5 - Book Review Predictions

### Introduction

For this project, we applied a Multinomial Naive Bayes algorithm to predict book ratings as either great, or not great. We obtained a dataset from Kaggle that included the raw review text and ratings for books on Goodreads book review website. Various text preprocessing methods were used, such as stop word removal and lemmatization. We also compared the difference in using unigrams and bi-grams for our bag of word representations. Overall, our best final model had an F1 score of 0.76, and we found that using unigrams slightly outperformed bi-grams in this instance. Our analysis will help inform book stores, authors, and other similar stores about the implications of customers reviews. It also has implications for sentiment analysis in book reviews.

Link to presentation slides:  CS6830-Group15-Project5-Presentation

Link to GitHub repository: <https://github.com/jakepper/CS6830-Group15-Project5.git>

### Dataset

The dataset used for analysis contained 900,000 customer reviews and ratings from Goodreads.com. The dataset is a free resource accessed from [Kaggle](#). The data was originally part of a Kaggle machine learning competition, so there were separate test and training files. However, we just used the training file, as the ratings were hidden from the test file. The original dataset contained ten variables: User ID, Book ID, Review ID, Rating, Review Text, Date Added, Date Updated, Read At, Started At, Number of Votes (e.g., that the review received), and Number of Comments (e.g., that the review received). Due to the irrelevance of many of the variables (such as the date variables), and the sparsity of the comment and vote variables (e.g., the 75th percentile for votes had 2, while comments had 0), we opted to just use the review text to predict the variables. The ratings ranged from 0-5. Additionally, we only used 100,000 total samples to save some runtime.

### Analysis Technique

The first step in our analysis was to recode the ratings into a binary variable, since we were using word counts as our predictor. We noticed a skew in the data towards ratings of 4 or 5 (see Figure 1), so we decided to recode ratings of 4 or 5 as 1 (e.g., great) and the rest of the ratings as 0 (e.g., not great). A significant part of the analysis for this project was text preprocessing. To clean the review text, we made everything lower-case and removed punctuation, URLs, the syntax for a new line ‘\n’ (a lot of reviews had this), and extra spaces. We also removed stopwords (e.g., commonly occurring words, such as “the”) and lemmatized the text (i.e., changing all variants of a word to the base form). We then created a vocabulary using the top 5000 occurring words using sklearn’s CountVectorizer, and transformed the reviews into “bag of words” representations. After receiving the results from the bag of words method, we used both unigrams (single words) and bi-grams (word pairs) to train two separate Multinomial Naive Bayes machine learning models. Each model was then optimized and then tested accordingly. We used Multinomial Naive Bayes because we were using words from the review text as predictors, which are categorical. Therefore, a Gaussian based model would not have been appropriate.

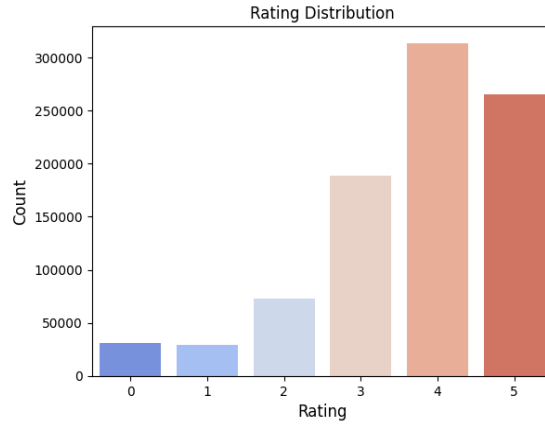


Figure 1

## Results

### Single words (unigrams)

The top 5 words in the vocabulary for unigrams, as well as their counts, are given in Table 1. We considered adding “book” to the stopwords list, but opted to keep it in, since we were also trying bi-grams. Figure 1 shows the results of the model from the unigram data. The precision score was 0.7476, with a recall of 0.7745, and an F1 score of 0.7608. The top 5 most influential words for both the “not great” class are given in Table 2, while the most influential words for the “great” class are in Table 3. The words that have the highest influence for either class seem intuitive. Notably, it appears that books that cause readers to feel emotions (e.g., “cried”) are likely to have a higher rating.

Word	Count
book	167011
like	68256
story	63336
read	62017
character	61232

Table 1

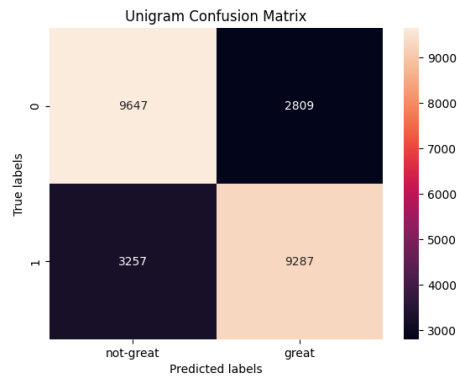


Figure 2

Word	“Not Great” Probability
dnf	0.9008
25	0.8744
meh	0.8123
disappointing	0.7859
flat	0.7803

Table 2

Word	“Great” Probability
45	0.8533
cried	0.7567
amazing	0.7546
wait	0.7399
beautifully	0.7267

Table 3

### Word pairs (bi-grams)

The top five frequently occurring bi-grams are given in Table 4. These are notably different than the unigrams, showing how bi-grams include additional context. Figure 3 shows the results of the final model for the bi-gram data. It has a precision score of 0.7367, a recall of 0.7676, and an F1 score of 0.7518. These scores were slightly worse than the unigrams, which we did not expect. It could be possible that we should have used more than the top 10000 bi-grams in our vocabulary. Or, unigrams are just better in this context. The top five most influential bi-grams for the “not-great” class are in Table 6, and the top five for the “great” class are in Table 7. We realized from these that the “25” and “45” were probably originally  $\frac{2}{5}$  or  $\frac{4}{5}$ , which makes sense as to why they have such a high probability for their respective classes. Furthermore, we found it interesting that saying “omg” or “wow” once did not show up in the top unigrams, but saying either word twice has a very high positive connotation. Overall, these results help inform authors or bookstores what customers will say if they think a book is great.

Bi-Gram	Count
hide spoiler	6852
main character	6089
feel like	5986
read book	5840
felt like	5499

Table 4

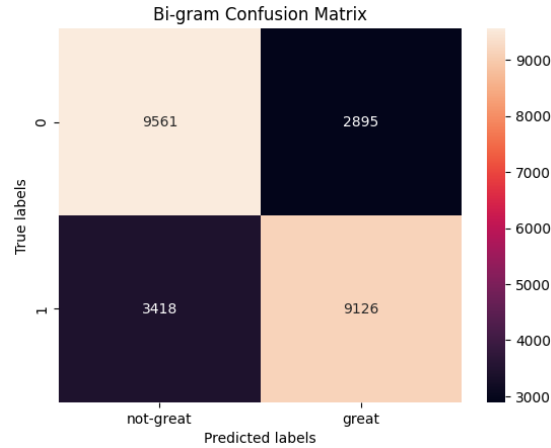


Figure 3

Bi-Gram	“Not Great” Probability
25 star	0.9717
rating 25	0.9345
two star	0.9336
wanted like	0.9311
15 star	0.9280

Table 2

Bi-Gram	“Great” Probability
45 star	0.9260
omg omg	0.9158
wow wow	0.9040
rating 45	0.8982
loved every	0.8864

Table 3

## Technical

Most of the technical details for the text preparation is given in the analysis technique section. However, we chose a “bag-of-words” representation, because it provides word counts, which is what a Multinomial Naive Bayes model uses to calculate probabilities. We made use of the Multinomial Naive Bayes algorithm to build our model as it provides a great classification platform that is simpler to implement than others when working with textual data. Since we were just using the raw text from ratings, transformed into word counts, the Multinomial version was appropriate. Attempts were made to optimize the model by using a grid search as well as cross validation to optimize the Multinomial Naive Bayes hyper-parameter alpha. We observed minor improvements to the models predictive capabilities after tuning the hyper-parameter alpha. We ended up with an alpha of 1000 for the model with unigrams and an alpha of 1 for bi-grams. We were then able to easily obtain results/scores for our models by making predictions on a test data set that stemmed from our previously cleaned dataset. Additionally, we performed the previous steps with a larger amount of training data and did observe a minor improvement in our results. We started out with 10,000 total samples, and the F1 scores of both models increased by about .06 when the training data was upped to 100,000.