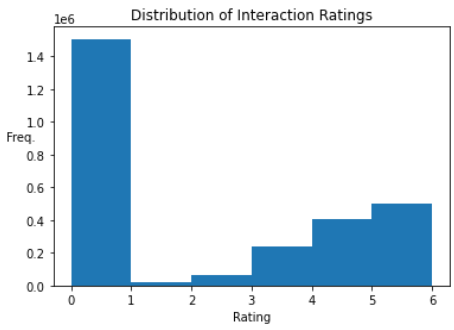
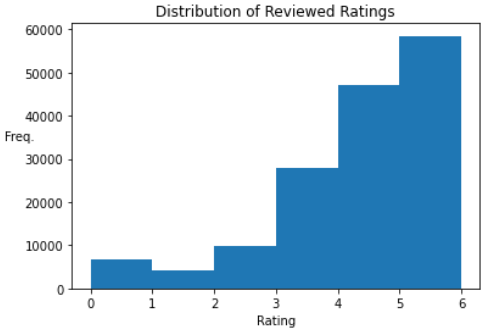
Assignment #2 - Goodreads Poetry Rating Prediction

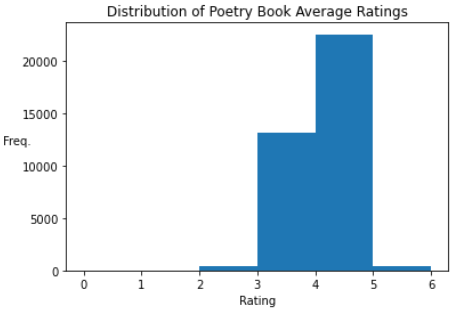
Sujeet Yeramareddy, Raymond Wang, Evan Kim

1. EDA

For this project, we chose to use the data about poetry books from Goodreads to make a predictive model about how a user would rate a given book. We used three separate datasets from the Professor's repository for recommender systems. The first one is the interactions dataset which contains approximately 2.7 million data points with features such as userID, bookID, rating given to book by user (0 representing no rating given), and datetime information. The second dataset has slightly more than 150,000 data points that contains the text of user reviews of poetry books, number of votes/comments, datetime information, etc. The third dataset is the data of ~36K books containing publisher information, book information, along with images and most popular shelves. Below we plotted the distribution of ratings in the three datasets to help us understand the distribution of poetry book ratings. Understanding these graphs can help us understand where bias and overfitting could arise from when we create our predictive model.

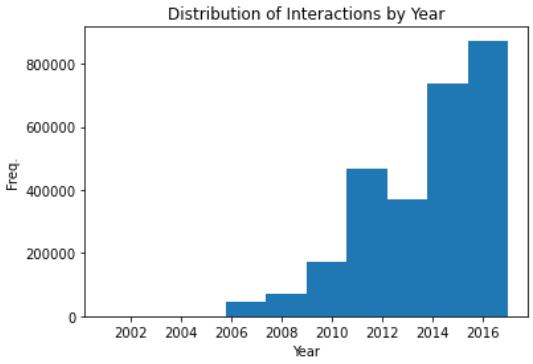


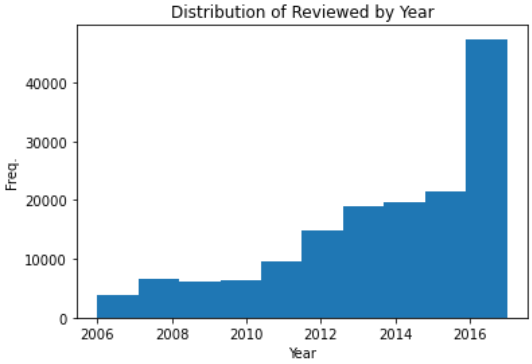




As we can see from the first histogram of all interactions most users don't rate books they interact with. This is expected behavior considering that more than half of our interaction data is an interaction between a user that didn't read the book where we would expect these data points to fall under the 0 rating. We can see that ratings of 5 occur the most which is expected because people who are passionate about a book they read are more likely to take the time and effort to rate it. Likewise, we can see in the second histogram of ratings of reviewed books, the frequency of each rating increases as the rating gets higher. This is also expected. Finally, the third histogram plots the distribution of average book ratings across all poetry books in this dataset. From these histograms we can clearly see a pattern among how common a specific rating occurs and because of this we decided to create a model that predicts the rating using two different feature matrices. In the histograms underneath, we can see how the dataset is distributed in terms of year of interaction and distribution of the year of all reviewed books, respectively. Most of our interactions occurred within the last decade, and similarly most of our reviewed dataset also occurs more recently. Both distributions seem to start around 2006, which might mean that was when the data was available to the public initially. Additionally, our poetry

book publications dates range from the 1300's to 2018, although we did have a few outlying reviews of poetry books being published in the future.





2-3. Predictive Tasks and Model Analysis

The Goodreads review dataset includes over fifteen million reviews, with each row containing information such as user id, book id, review text, and rating. This entire dataset is over five gigabytes in size and it is unrealistic for the team to run the entire dataset due to hardware limitations. Therefore we have decided to focus on the poetry genre in the Goodreads review dataset, which contains over 150,000 data points.

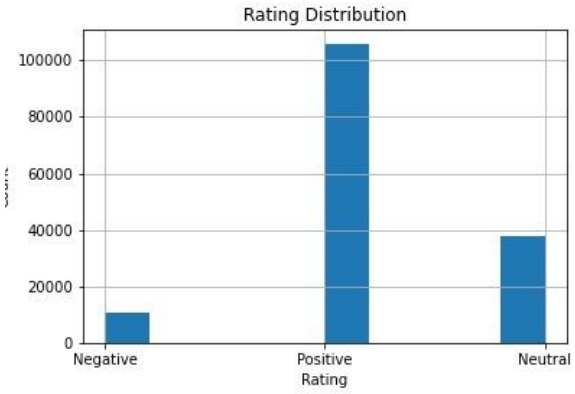
As our predictive task, we will predict the ratings a user will give a poetry book. We will first predict the ratings by using a similarity based method, then we will do a sentiment analysis on text review to predict the ratings, both models will be trained using the Goodreads review dataset for poetry. Because all the ratings in our dataset are integers from one to five with no decimal points, we have decided to round our predicted ratings so that they will be whole integers. We will evaluate our model performance by the accuracy of our predicted ratings.

For our baseline model for our similarity based method, we will simply predict the ratings as the average rating of the entire dataset. For our poetry data, the average rating is 3.8. This will round up to a rating of 4, so we will predict a rating of 4 for every user book combination. This results in an accuracy of about 0.30 for both the train and test set. Next, we will use the Jaccard similarity to predict the ratings a user will give a book. To predict the rating that the user will give a book, we will create a list of previous ratings that the user has given and a list of similarities of the different items using the Jaccard similarity. Then we will compute the weighted ratings by taking the product of the ratings and similarities and averaging the sums, then rounding to the nearest whole number. This results in an accuracy of 0.68 on the training set and an accuracy of 0.62 on the test set, much better results than our baseline model. When we swap the Jaccard similarity function for a cosine similarity function, we get a train set accuracy of 0.66 and a test set accuracy of 0.64. The training set accuracy is lower using the cosine similarity function than using the Jaccard similarity to predict the ratings. However, the cosine similarity model seems to be performing slightly better on the test set than the Jaccard similarity model. This indicates that the cosine similarity is doing better at generalization than the Jaccard similarity. However, the difference between the accuracies are less than four percent and this may just be due to variance in the data and we should not come up with a final model just based

on these numbers. So we will experiment with different models and further improve our prediction accuracy.

During further investigation, we noticed that the distribution of ratings are not even and ratings can be very subjective to individual users, so we have decided that a sentiment analysis on text review may be a better method to predict ratings.

We do this by grouping the rating to “positive”, “neutral”, and “negative”. First, we drop all rows where the user does not provide a rating. Then, we group the ratings in the following scheme: if the rating is 0 or 1, then we categorize it as “negative”; if the rating is 2 or 3, then we categorize it as “neutral”; if it is 4 or 5, then “positive”. Now, we will be using natural language processing and classification models to predict the sentiment of the review text.

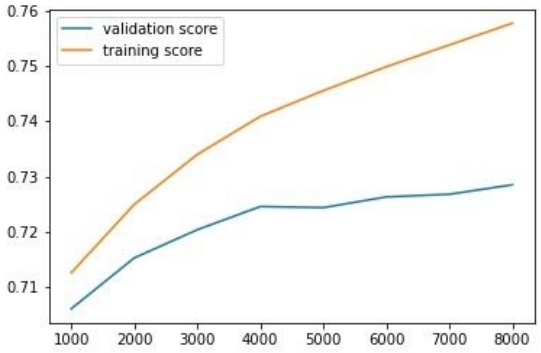


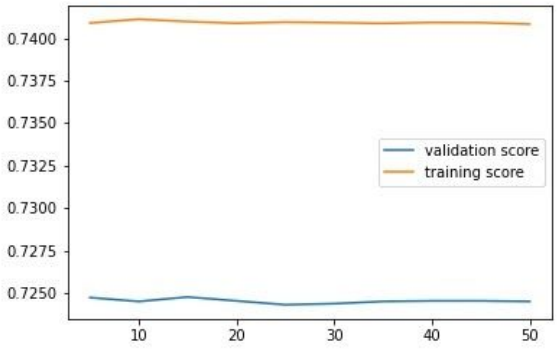
To do so we will first split the dataset into train, test and validation sets, with respective proportions of 0.6, 0.2, 0.2. We will fit the data on the training set, adjust parameters of models with the validation set, and test our predictive accuracy on the test set. The rating distribution, as shown in the histogram, is extremely negatively skewed. Therefore, when considering the accuracy for our models, we will put more weight into the accuracy for the higher ratings. We will preprocess the text data by removing the punctuation and stop words, and converting all text into lowercase to avoid redundancy.

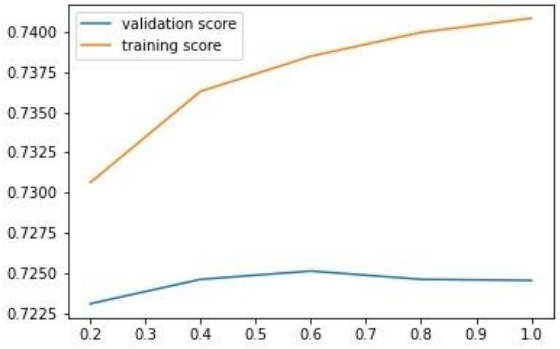
For the baseline, we will build a basic bag of words model by extracting 1000 most common words. Then we will fit it into a logistic regression model with the regularization constant set to 1. After that, we will use TF-IDF as a metric to help build our bag of words. This technique is helpful in our scenario since TF-IDF takes into account how often a word appears in the entire dataset, but also accounts how often the word appears in each document. Therefore, this technique will give us a more accurate predictive model. Furthermore, we will explore different models including Naive Bayes, Decision Tree, Random Forest, and Gradient Boosting. We will set up a pipeline for each model and determine the best model for this scenario with the validation set.

For the baseline model, we got an accuracy of 0.66. We will start from there. We first use TF-IDF as the metric to vectorize our reviews. To keep it simple for now, we choose ‘max_feature’ to be 1000, “min_df” to be 10, and the regularization constant “C” for the logistic regression model to be 1. We will later adjust those values to optimize accuracy. After vectorizing the data, we will fit it into each one of the models mentioned above. Overall, the logistic regression model outperformed the other models with an average accuracy of 0.71. Next up, we want to finetune some parameters. For the TF-IDF vectorizer, we want to try out a few different values for the “max_features” (maximum number of features used) and “min_df” (minimum number of documents that a word needs to appear in) parameters. For the logistic regression model, we want to try different values for the regularization constant, “C”. For each

parameter we provide a series of suitable values, run the model on each one of them, record the accuracy on both the training and testing dataset, and finally graph the result to check for overfitting. For the “max_features” parameter, we see the training accuracy increases as the number of maximum features increases. However, the validation accuracy plateaus at 4000. This is a sign of training data overfitting and therefore we choose 4000 for the parameter “max_features.” For the “min_df” parameter, we see that there is no significant change across different options of the parameter. We suspect that this is because we have already fine tuned the “max_features” parameter, which can be redundant with the “min_df” parameter. Since the accuracy doesn't vary much, we choose our default value, 10. Lastly, we see that the training accuracy increases as regularization constant increases .However, the validation accuracy does not change much. We set C equals 0.5. For the final result, we ran the pipeline with the selected parameters on the test dataset and achieved an accuracy of 0.723.







Compared to the baseline model, our model yields significantly better accuracy (more than 6 percent). It outperformed the baseline model because it uses the TF-IDF metric, which accounts for both a word’s frequency in the document and its frequency in the entire dataset. Moreover, a larger maximum feature (4000 compared to 1000) allows us to retain more information from the text review.

4. Literature

We are using the dataset from Prof. McAuley’s repository which was obtained by scraping the goodreads website for user-book interactions, poetry book list, and poetry review text list. The dataset was obtained from the public view and the ID’s for users and reviews are anonymized to maintain confidentiality. We used this dataset to extract information like overall interactions, and review text to predict what a given user would rate a given book.

In 2016 Professor Anna Rafferty conducted a study which was done with the help of a group of data scientists researching the field of book recommendations. This group used datasets from

BookCrossing, Amazon Book reviews which are both smaller than our Goodreads dataset. For their first approach, they decided to use the text of the reviews to conduct a sentiment analysis of a user's review and use that to predict how that user would rate the book. They started with cleaning up the text data by accounting for case-sensitivity, lemmatizing each word to its base form and removing stopwords, which is similar to our text cleanup as described above. Their first step was to use TF-IDF valued representations of each book and they later tried to use sentiment analysis based on stop phrases that could contain sentimental value. For example, they would have specific modifiers such as 'very' or 'not' or 'however' or 'but' that they used as features in their sentiment algorithm. They decided to do this because they found through their EDA that most of the reviews were very opinionated, therefore by using key stop words it can help improve the accuracy of their algorithm. Similar to their idea of using sentiment to help improve accuracy of their model, we decided to group ratings in terms of negative, neutral, or positive review. Similar to the results of this research, we found that using TF-IDF representation increases the accuracy of our model because it is a better representation of meaningful words. Additionally, our similar text cleaning strategies proved to improve our model as well.

In 2010, Avi Rana and K. Deeba published a research paper on how they used collaborative filtering to create a book recommendation system. The dataset they used was compiled by Cai-Nicolas Ziegler with ~200,000 books, ~1.1M user ratings, and ~278K users. To make their recommendation algorithm they chose to use the jaccard similarity metric which computes the ratio between the size of the intersection and the size of the union of the compared sets to determine how similar the compared sets are. Later in their paper, they suggest that due to the common expected distributions of ratings, most researchers use adjusted vector cosine similarity, which helps take into account that different "users have different rating schemes". This is similar to our approach because we used cosine similarity adjusted with the rating the user gives; this way our recommended book takes into account how the user rates other books as well as the actual other books they read. For our dataset, using cosine similarity didn't improve our model accuracy substantially which could be a result of biases in our dataset.

Overall, from all the research papers we've read, we can see that most common recommender system methods for book recommendation consists of collaborative filtering and some use a hybrid method which combines the content-based approach with collaborative filtering for a more diverse model. We can see that by using similar approaches, we have improved our models from the baseline, similar to the improvements of the results from these studies.

5. Conclusion

In predicting users' ratings on poetry books, we attempted mainly two different methods. Our first approach was to use a similarity based model. Using a Jaccard similarity, we got a train set accuracy of 0.68 and a test set accuracy of 0.62. Then we replaced the Jaccard similarity with a cosine similarity function, resulting in a train set accuracy of 0.66 and a test set accuracy of 0.64. Using the cosine similarity resulted in a lower train set accuracy than using the Jaccard similarity, but the cosine similarity produced a higher test set accuracy. This indicates that the Jaccard similarity is overfitting the training set and that the cosine similarity is doing a better job

at generalizing the data. Our next approach was to use sentiment analysis to predict users' rating on poetry books. Since each rating is heavily subjective to the individual user, we grouped the ratings into general sentiment towards the literature. Then we use natural language processing methods to predict the sentiment given a text review. Our baseline model was a simple bag of word model with dictionary size 1000 fitted into a logistic regression model. To improve from our baseline, we made use of the TF-IDF metric, which helps increase the accuracy substantially. We tried fitting the new features into several different models and decided that the logistic regression model still provides the best accuracy. Next we tuned our parameters, mostly the maximum size of the word features, the minimum document a word needs to be in to be considered, and the regularization constant for the logistic regression model. We calculated the accuracy of the model with different combinations of values of the parameter on both the training set and the test set and graphed them to check for overfitting. Overall, a bag of word model using TF-IDF metric, fitted into a logistic regression model with regularization constant C equals 0.5, with parameters "max_features" and "min_df" being 4000 and 10 respectively provided the best result on the test set, with an accuracy of 0.723. This result is satisfactory considering many users type in random letters in their text reviews so it is difficult to predict those users' sentiment given what they have provided. In the future, we can improve our model by taking those random reviews into account. A possible improvement could be to identify those reviews and predict them as a neutral sentiment.

Works Cited

Avi Rana & K. Deeba, “Online Book Recommendation System using Collaborative Filtering (With Jaccard Similarity)”, SRM Institute of Science and Technology, Chennai, India 2019

Mengting Wan, Julian McAuley, "Item Recommendation on Monotonic Behavior Chains", in RecSys'18. [bibtex]

Mengting Wan, Rishabh Misra, Ndapa Nakashole, Julian McAuley, "Fine-Grained Spoiler Detection from Large-Scale Review Corpora", in ACL'19. [bibtex]

Professor Anna Raffery et. al, “Book Recommendation System”, http://cs.carleton.edu/cs_comps/1617/book_rec/final-results/paper.pdf