Book Recommendation System Using Ensemble Approach

Umesh Maharshi Chinalachi[1], Anurag Yedla[2], Giang Nguyen[3], and Sri Harsha Nadella[4]

[1,2,3,4]Department of Computer Science, Iowa State University

***Abstract*— The aim of this work is to develop and evaluate a book recommendation system using ensemble model approach. We merged K means clustering, Cosine similarity and Heuristic function for building the Recommendation System. The model recommends books based on the authors of previous books read by the user, books read by the user and books read by similar users. We then evaluated the model based on the books recommended with rating prediction and RMSE.**

***Keywords:* Recommendation, K Means Clustering, Heuristic, Cosine Similarity, Ensemble**

I. INTRODUCTION

A recommendation system is an artificial agent with uses Artificial Intelligence and Machine learning techniques to recommend products or items to the users. They help in narrowing down the choices and reducing the overhead associated with selecting the best product. Recommendation systems are highly popular among E-commerce and media-services providers. A recommendation system consists of two important entities, one is the user to which the system recommends and the second is the product itself. Generally, the system takes the database of users, their behaviour, database of the products and their attributes. For our model, the input to the model is list of books and their attributes like ratings, genres the book associated with, authors of the book along with list of books the users have read(user history/behaviour). The database is downloaded from Kaggle [1].

II. ENSEMBLE BOOK RECOMMENDATION SYSTEM

In our ensemble model, we combined results from three different sub models and pick the recommendations based on their confidence values. This way make avoid picking products based on only one model and also to reduce getting into loop of recommending same products every time. Wwe also randomly recommend two most popular and highly rated books.

The data downloaded contains list of books with their ids, author ids, ratings and genres tags associated with them. There are about 10k books, around 6500 authors and about 55k users. Now, we divided the model into 3 parts namely **Author Clustering**, **Books Similarity** and **User Based Collaborative filter(UBCF)**. We will describe in detail about each model in the coming sections.

A. Author Clustering

In this model, we cluster author into ten different sets based on how similar they are to each other. The most important task of this model is to convert ordinal data into a clusters, i.e group similar author names or ids together. To solve this problem, we came with a metric function to represent the attributes of an author. We then cluster authors based on the defined metric function.

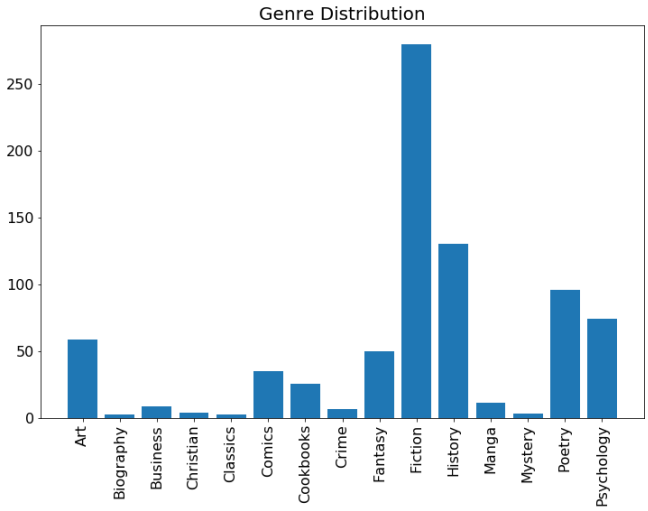

Fig. 1. Genre Distribution

Author Representation equations:

$$G = [g_1, g_2, g_3, ..g_g], \tag{1a}$$

$$A = [a_1, a_2, a_3, ..a_n] \tag{1b}$$

$$a_i = [g_{1i}, g_{2i}, g_{3i}, ..g_{gi}] \tag{1c}$$

Here G represents the list of available genres and g_i represents its name. A is the list of all available authors and each a_i is represent by a $1 \times g$ dimensional matrix. The j^{th} element in the a_i represents the number of books written by the author a_i that tagged as genre g_j. Also, n represents the total number of authors available and g is the size of relevant genres. Moreover, to strengthen the model, we removed all the tags which have a count of less than 50

After building the matrix A, the next step is to normalize the attributes of each a_i i.e, make .

$$\Sigma_{j=1}^{g} g_{ji} = 1, \forall i \tag{2}$$

Once, we achieve normalization, we then used KMeans Clustering [3] algorithm to cluster them into ten different groups. Our next step is to recommend books given an author name. For this, we randomly pick ten authors from the author's cluster and after that we the pick few books

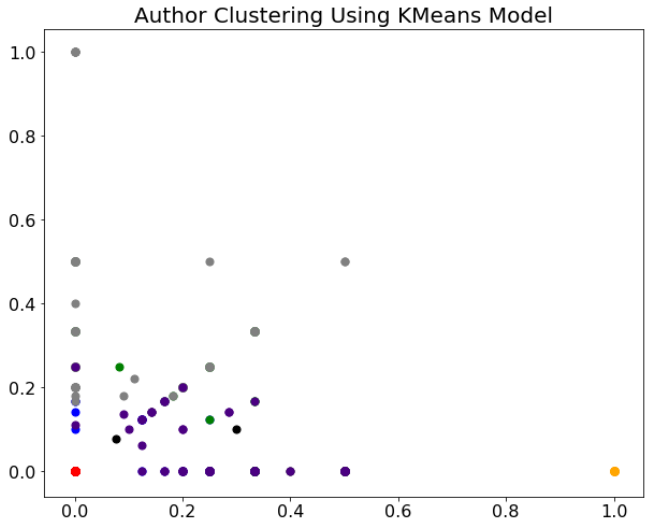

Fig. 2. Author Cluster

written by each author based on the confidence value in this case book rating and return only the top 10 out of all the books. By doing this we are not only avoiding the restriction of a user's domain of recommendation but also giving user more choices.

Fig 1 is a bar chart representing the genre distribution and Fig 2 is a scatter plot representing author clustering. From the figures, we can infer that a user who reads fiction has lot of choices compared to a user who reads Biography or Crime. We can also notice that the scatter plot is not grouped properly, this is because of the skewed count of number of books tagged under fiction.

B. Book Similarity

1) Preprocessing: . We use the k-Nearest Neighbors algorithm to find out the top common book. However, it is necessary to preprocess the dataset to increase the performance of the algorithm. To do that, we explore the rating distribution to filter the low number of rating star. Moreover, we extract the personal name of location name from the book titles to make a comparison with other books. Lastly, we find the common author between two books.

Ratings data. The rating data set provides a list of ratings that users have given to books. It includes 981,758 records of 10000 books. We explored the rating distribution of rating dataset and found out that there are more number of 3, 4 and 5-star ratings than 1 or 2. Therefore, our whole dataset is currently 3, 4, and 5-star biased.

Common Book Tittle We extract the personal name or location name from the title of a selected book and then compare with other titles. For example, The New York Trilogy is a book in our database and our dataset also has an another the book which has a title Live from New York: An Uncensored History of Saturday Night Live. These books' title mention New York; therefore, they may have some similarity in their content. These personal names or

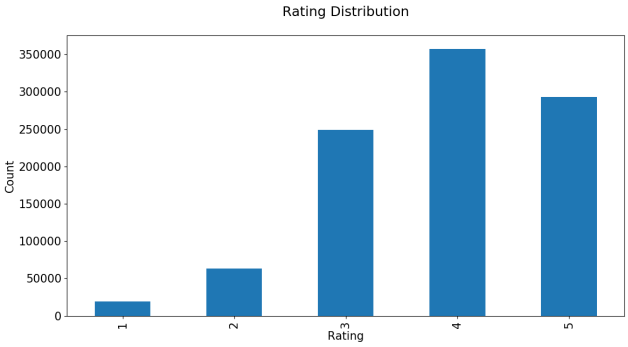

Fig. 3. Rating Distribution.

location names are being found in the title using the corpus module of the Natural Language toolkit library that contains automatically generated reader instances to read the text and find the special names.

Common Book Authors Each author has his/her writing style. For instance, J.K. Rowling is a fiction author, and most of her books tell fiction stories like Harry Potter or Fantastic Beasts and Where to Find Them. Therefore, if a reader loves reading Harry Potter, he/she would like to read other books of J.K. Rowling like Fantastic Beasts and Where to Find Them.

2) Collaborative Filtering Using k-Nearest Neighbors (kNN): KNN is a machine learning algorithm to find clusters of similar users based on common book ratings, and make predictions using the average rating of top-k nearest neighbors. For example, we first present ratings in a matrix with the matrix having one row for each item (book) and one column for each user. We use unsupervised algorithms with sklearn.neighbors. The algorithm we use to compute the nearest neighbors is brute, and we specify metric=cosine so that the algorithm will calculate the cosine similarity between rating vectors. Finally, we fit the model. Then, the kNN algorithm will measures distance to determine the closeness of instances. It then classifies an instance by finding its nearest neighbors, and picks the most popular class among the neighbors.

3) Combine heuristic function with KNN: We apply KNN to find a similar book based on common book ratings. Then, we will have the list of output is the distance d. We use common book titles and common book authors as a heuristic function for distance to increase the value of d. Then, based on the distance d, we can rank the similarity books.

If the two books have one or more common author: d = d + 0.03

If the two books have one or more common special name: d = d + 0.05

C. User Based Collaborative filtering

Collaborative filtering is a method where recommendations are made to a user based on the interests and behaviour of similar users. We assume that if two users rate the common books read by them in a similar fashion, their likes

and dislikes for other books might be same. Based on this assumption we first find similar users to the current user and then based on the behaviour of similar users we recommend the books that current user might like. We break down this method into 3 tasks.

1) Identify users who have rated 3 books in common with the current user: We remove all the users who have rated less that two books. Then we form a list of all the users who have rated at least three books among the books that the current user has rated.

2) Computing the cosine similarity between users ratings: We compute the cosine similarity of the ratings given by two users for common books. The cosine similarity is computed as follows; U_1 = [$R_{1,1}, R_{1,2}$, ...] be the ratings of user one and U_2 = [$R_{2,1}, R_{2,2}$, ...] be the ratings of user two, where $R_{1,1}$ is the rating of user 1 for book 1. We compute the cosine similarity as follows;

$$1 - cos(\boldsymbol{U}_1, \boldsymbol{U}_2) = 1 - \frac{\boldsymbol{U}_1 \cdot \boldsymbol{U}_2}{||\boldsymbol{U}_1|| \cdot ||\boldsymbol{U}_2||} \tag{3}$$

3) Select top 3 highest rated books read by similar users: We select the top 3 highest rated books read by similar users which are not red by the current user.

4) Advanced Ideas:: We can improve the algorithm using the following ideas:

Instead of simply averaging the predictions of the similar users you can weight the ratings by similarity. This means that the more similar a user is to the current user the more weight his/her ratings receive in the calculation of the predictions.

The similarity calculation can also be weighted, according to how many books users co-rated. The more books users co-rated the more reliable is their similarity score.

III. Experiments

We used the following three techniques mentioned in section II to recommend book to a user.

A. Recommending books based on Author Similarity using clustering

B. Recommending books based on Book Similarity using K Nearest Neighbours

C. Recommending books based on User Based Collaborative Filtering

D. Using Singular Value Decomposition to predict the rating given to a book by a user

For Singular Value Decomposition we used grid search to tune the hyper parameter to get optimum results. The hyper parameters are as follows : Number of epochs : 10, Learning rate for all parameters: 0.005, Regularization for all parameters : 0.4. Figure 4 shows the work flow of the recommendation system.

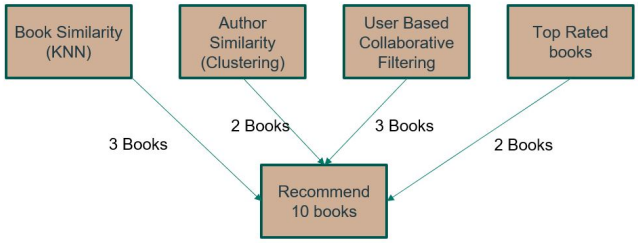

Fig. 4. Work flow of the Recommendation System.

IV. Results

Evaluating the recommendation systems is not straight forward. One way to gauge the performance of the recommendation systems is to look at how many recommendations were successfully taken and looking at the ratings. Root mean squared error is also a helpful metric to compare the performance. We compared the original rating given by the user and the rating predicted by the model.

Book recommendations to the user		
Book Suggested	Original Rating	Predicted Rating
The Fault in Our Stars	3	3.86
The Maze Runner (Maze Runner, 1)	5	4.78
Looking for Alaska	5	4.54
All the Light We Cannot See	4	4.21
The Complete Phantom of the Opera	3	2.89

The predicted rating is very close to the original rating. The Root Mean Squared Error (RSME) value for the predictor using singular value decomposition is 0.985. Grid search was used to tune the model with hyper parameters which yield the best result.

V. Conclusion

To summarize, we developed a ensemble model which takes into consideration author similarity, book similarity and also collaborative filter. We then fetch two books from author cluster model, three books from Book similarity model, three from collaborative filter and also we randomly pick two of most popular and highly rated books from the entire list. We combine all these to create the output of ensemble recommendation system. The model was evaluated based on the rating prediction and RMSE. We got an RMSE value around 0.98 which is pretty close to the values used by major e-commerce and media streaming companies. Although, the predictions were accurate and diverse further improvements needed to be done in improving the run time of the model.

References

[1] goodbooks-10k
https://www.kaggle.com/zygmunt/goodbooks-10k

[2] Book Recommendation System, IJIRST International Journal for Innovative Research in Science and Technology | Volume 1 | Issue 11 | April 2015
[3] KMeans Cluster
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html