**Book rating prediction using Collaborative Filtering**

**Team members:**

Anqi Chen (axc157130)

Xinhe Chen (xxc170630)

Yisu Tian (yxt172830)

**Introduction:**

Nowadays, an increasing number of web based merchant or rental services use recommender systems. Some e-commerce web like Amazon and Netflix have successfully applied recommender systems to deliver personalized recommendation to their customers. The task of recommender system is to recommend items that fit a user’s taste, in order to help the user, make the decision when selecting or purchasing items from a large set of choices. The ability of recommender systems to provide personalized suggestions greatly increase the likelihood of a customer making a purchase compared to un-personalized ones. [3]

Collaborative filtering (CF) is one of the most effective methods to build a recommender system. CF simply means that people collaborate to help one another perform filtering by recording their reactions to documents they read. [4] CF approaches can be applied to recommender systems independently of the domain. The CF algorithms identify relationships between users and items and make associations using this information to predict user preferences.

The goal of this project is to create a method to recommend future reads to the user based on their ratings for books they have read. Item based and user based collaborative filtering will be used to create an item-user matrix. Unrated items for each user will be predicted using the algorithm created. The performance of these two approaches will be compared and discussed.

**Problem Definition and Algorithm:**

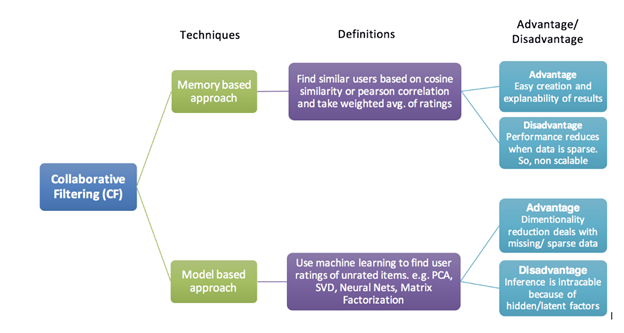
**Task Definition**

A recommender system recommends users items based on items they have rated. Ratings for ten thousand books have been collected and a recommender system shall be built using memory based collaborative filtering to predict the rating a user will give a book. Item based and user based collaborative filtering will be compared using different methods of calculating similarity including cosine similarity, Pearson correlation, and Jaccard Index.

**Algorithm Definition**

There is generally two approaches to collaborative filtering. Memory based and model based as shown in figure 1. Memory based approach uses cosine similarity to calculate the ‘distance’ between users or items and take a weighted average of ratings. It is easier to create than the model based approach but suffers when there is not enough data. Model based approach uses methods such as neural networks and does not need as much data, however it is harder to implement. This project will use the memory based approach as there is sufficient data present.

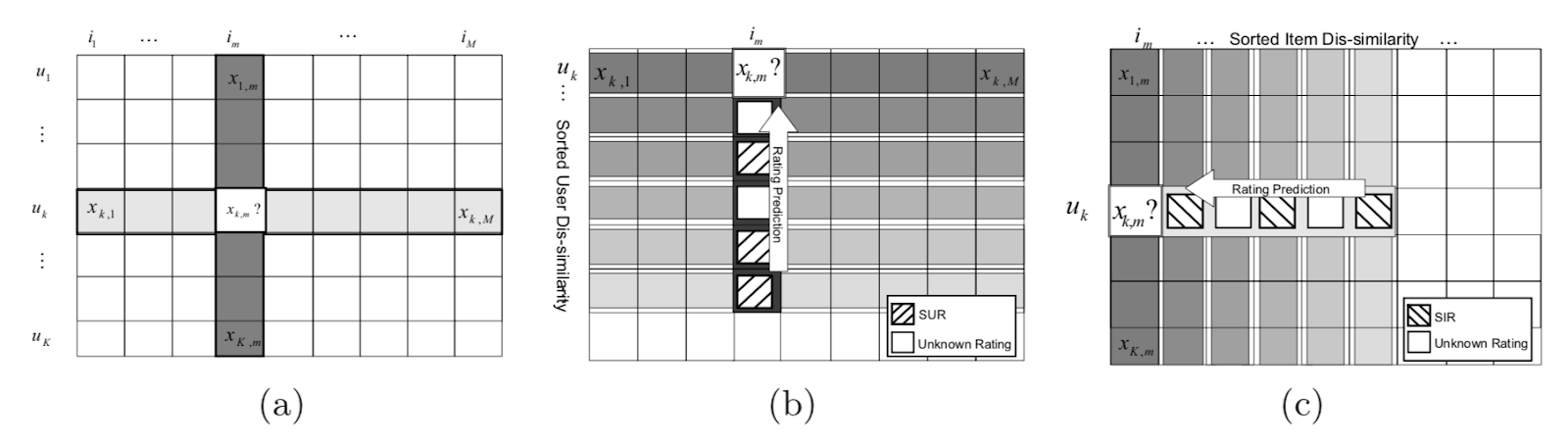
Memory based approach can be further separated into two categories. User-based collaborative filtering and Item-based collaborative filtering.



**Figure 1.** Types of collaborative filtering approaches [2]

User-based filtering takes a user and find similar users and recommend the items these similar users like. Each user profile (row vector) is sorted by its dis-similarity towards the test user’s profile. Rating by more similar users contribute more to predicting the test item rating.

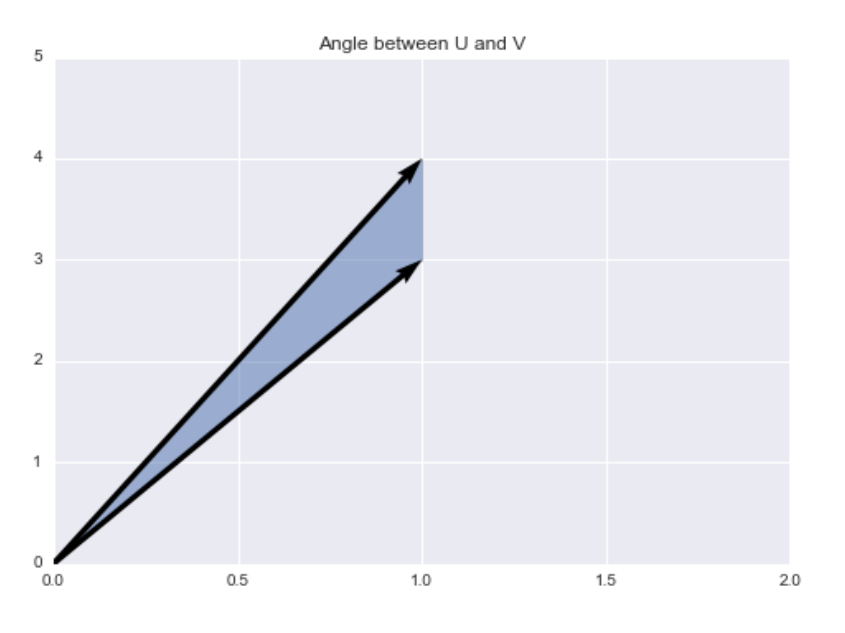
Item-based filtering takes an item a user likes and finds users who also like this item and recommend the items these other users like. The unknown rating of a test item by a test user can be predicted by averaging the ratings of other similar items rated by this test user. Again, each item (column vector) is sorted and re-indexed according to its dis-similarity towards the test item in the user-item matrix, and ratings from more similar items are weighted stronger. [6]



**Figure 2.** (a) The user-item matrix (b) Rating prediction based on user similarity (c) Rating prediction based on item similarity [6]

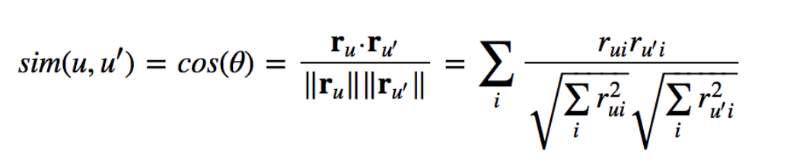
Both approaches use the same method of similarity to evaluate ‘distance’ between item or user. There are many ways to calculate this similarity including cosine similarity, Pearson correlation coefficients and Jaccard Index.

To measure distance between user or item, the rating a user gives an item can be considered a vector. To find the most similar item, the vector that is closest to the item should be found. To measure the distance between these vectors the angle between these two vectors are calculated as shown in figure 3. If the angle is 0 degrees, they are the same, if 180 degrees they are the exact opposite.

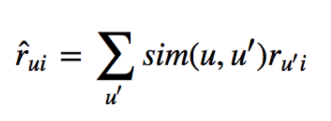


**Figure 3.** Cosine similarity between two vectors [2]

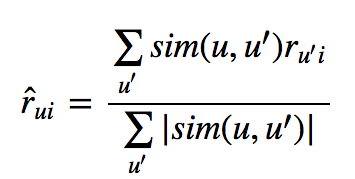
The cosine similarity between u and u’  can be calculated through the following equation



The weighted sum of all the other users or items can then be calculated to predict the rating of an item for user u using the equation:



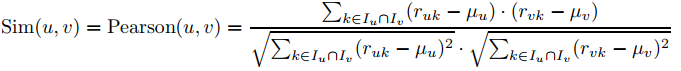
To normalize the ratings by the total number of other users’ ratings we use:

****

Another way to measure the similarity between the rating vectors of two user’s u and v is the Pearson correlation coefficient. The mean rating is first computed for each user u:

https://lh5.googleusercontent.com/G9Ck1m3XLy8_jNuhjsM5m0i89sc4ww0UstMX11i6A1nK09zQsJG56uYLrCRnvzW3EkSyLT7kzV4PFqeSn1nTEfQ77YLWthqZTHiYo0jNtTasnSX0sBpkK1VpL4UF_RAHq1XpSeAe

The Pearson correlation coefficient between the user’s u and v is the calculated using:



To prevent the case that different users rate items on different scales, the raw ratings are mean-centered in row-wise fashion before calculating the average rating of the group

https://lh4.googleusercontent.com/M_mdUTf9GP9_NMBIB2bBF-DhC9KxQuKk3hxUiOe7c-VEXxIUG-171WEgCfhBpqDCVzJksDz6cvwX8Vzn7GRmXzxcxAL_k4NX89xFVgv7U2W18M4p1XDmm9hCZrFCSsJxqAUEk0mY

The prediction rating of a user u on an item j can be calculated using:

https://lh5.googleusercontent.com/M3u96f9KqUIS16ucYzxm3IJ1Ji0WKsIWNgfZyvfJgjanT4gnnX0jTchSJPqjMUN1OE-yICc5A4ERP3HOjCZO0qXkcZDfRGlrmLotVZ_x6163I1vr5Kdqg0sbXyqxJ8nPpA5ISixR

Jaccard index measures the number of items commonly rated by two user’s u and v out of the total number of items rated by them using:

https://lh5.googleusercontent.com/metwlIzVdBaqua2uoRBx6kovmIhxX2nqh8cxvvzahi7ZX8hkyS-rXmzIXkYO6s8h9QdxCzfilIYrIIJaTMyYl7ns0TID6FPVWJpqZlxfD8iEq4MXP29I5e_yrhIUYUKq1kBqW2zY

The issue with Jaccard is it assumes users who have given ratings to the same items are similar, but does not take into consideration the actual rating each user gives.

**Experimental Evaluation**

This project uses Python 3.6 with libraries including numpy, pandas, sklearn.

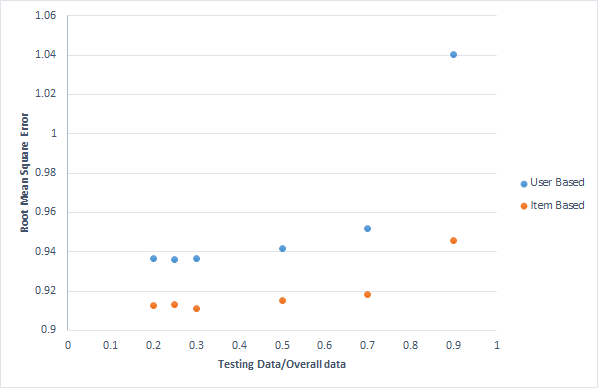
The analysis will use memory based collaborative filtering to predict the ratings of books for users based on their ratings for other books. The data can be found here:

<https://github.com/zygmuntz/goodbooks-10k>

The data used contains ratings for ten thousand books. With more than fifty thousand users, each user has at least made 2 ratings. The data will be separated into training data and testing data to evaluate the algorithm. Different ratios of training data to testing data will be tested to see the effect of large training data to low training data. Due to the large data present, regular computers can run into memory issues when processing the data. The matrix may need to be shortened/cut to make the program runnable. Different amounts of users will be fed into the algorithm to see the effect of the number of users to the accuracy of the algorithm. Finally, the three methods Cosine, Pearson and Jaccard will also be compared.

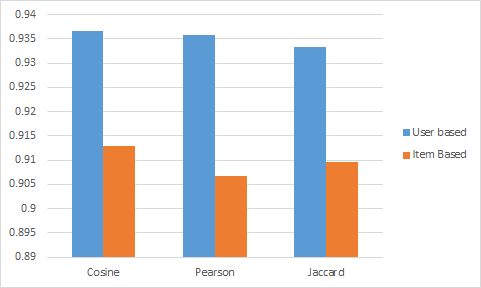
**Results**

The overall data was split into training data and testing data.  The ratio of training data to testing data was tested to see the effect it had on the error produced by the program. The ratings for 10,000 users on 10,000 books were used and cosine similarity was used for the similarity calculations. As shown in Figure 4. The more testing data and less training data, the higher the root mean square error. This is expected since the algorithm needs large amounts of training data to produce an accurate estimate. Once the Testing data was lower than 30% of the total data, the error stayed about the same showing the algorithm had enough data fed to produce an estimate. In other tests, a testing data of 20% will be used to ensure the algorithm has been thoroughly trained.

****

**Figure 4.** Effect of testing data/overall data to error produced.

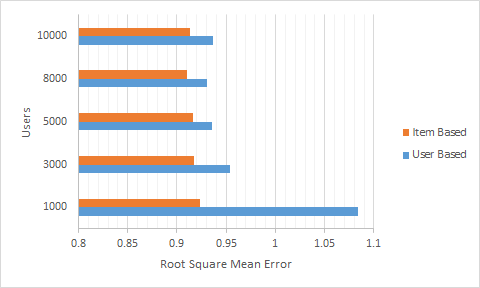
Three different similarity calculation methods were compared and results are shown in figure 5. Cosine similarity seems to perform the worst with Jaccard performing the best in the user based section, and  item based Pearson performing the best overall. Jaccard performs best in the user based because Jaccard calculates the similarity between users based on the books they have both read. Pearson performs the best because if there are item ratings not shared by both users, Pearson drops them, but cosine treats them as 0.

****

**Figure 5.** Comparison between Cosine Similarity, Pearson correlation coefficient, Jaccard Index.

The next comparison made was effect of the amount of user base to the error produced. As shown in figure 6. The lower the users present, the less accurate the algorithm becomes as expected. The algorithm cannot compute an accurate estimate with low training data and the user based collaborative filter suffers significantly with low amounts of users.

The last comparison was made throughout the tests. In all cases, Item based collaborative filtering performs much better than user based. This can be because item based looks at the similarity between items while user based looks at the similarity between users. If a user likes one item, it is highly likely they will like similar items. However, a person may have many different likings, and two user’s liking in one item may not correlate to their liking in other items.

****

**Figure 6.** Effect of user base in collaborative filtering.

**Future Work**

Due to the large quantity of data, regular computers cannot calculate the all the similarities for all the data. To decrease the matrix calculations, methods like k means can be used to first separate the users into k groups. Once all the users are separated into their own groups, they are more likely to have similarities to users in their own group. Collaborative filtering can then be used on each group to predict their ratings on unrated books.

**Work Distribution**

All members contributed to searching for the data. After deciding on using the data found here:<https://github.com/zygmuntz/goodbooks-10k>, and deciding on using collaborative filtering, the group discussed how to write the program, and Yisu Tian oversaw writing the program. Anqi Chen and Xinhe Chen were in charge of learning more about different approaches to the algorithm and writing the report.

**References**

[1] Nandi, M. (2018). *Recommender Systems through Collaborative Filtering*. [online] Data Science Blog by Domino. Available at: https://blog.dominodatalab.com/recommender-systems-collaborative-filtering/ [Accessed 27 Apr. 2018].

[2]Towards Data Science. (2018). *Various Implementations of Collaborative Filtering – Towards Data Science*. [online] Available at:https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0 [Accessed 27 Apr. 2018].

[3] Takács, G., Pilászy, I., Németh, B., Tikk, D.: Scalable collaborative filtering approaches for large recommender systems. J. Mach. Learn. Res. 10, 623–656 (2009)

[4]D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[5]C. C. Aggarwal, *Recommender systems: the textbook*. New York: Springer, 2016.

[6] J. Wang, A. P. de Vries, M. J. T. Reinders, Unifying user- based and item-based collaborative filtering approaches by similarity fusion, in: Proceedings of the ACM Conference on Research and Development in Information Retrieval, 2006, pp. 501–508.