BAHÇEŞEHİR UNIVERSITY

FACULTY OF ENGINEERING AND NATURAL SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

CMP2003 DATA STRUCTURES AND ALGORITHMS

PROJECT ASSIGNMENT

Name Last Name: Fazlı Altun, Mücahid Enes Deniz, Musa Eren Yakut

Student ID: 2102402, 2003376, 2001357

Due: January 6th, 2023

Academic Honesty Pledge

- I promise, *on penalty of failure of CMP2003*, not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.
- I understand that all resources in print or on the web must be explicitly cited.
- I acknowledge that I have also been advised by academic staff about standards for good academic conduct and how to avoid plagiarism and other assessment irregularities.
- I acknowledge that the inclusion of a footnote or a source in a bibliography is insufficient for attribution of another's work.
- In keeping with Bahcesehir University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it. I acknowledge that any work that I submit for assessment at Bahcesehir University:
    - Must be all my own work, unless this requirement is specifically excluded when part of a designated group assignment
    - Must not have been prepared with the assistance of any other person
    - Has not previously been submitted for assessment at this University or elsewhere.
- I acknowledge that I must take reasonable steps to ensure that all contents of my project are kept secure so that I do not enable another person to copy my work.
- I accept that Bahcesehir University may check the originality of my work using a range of techniques, including computer-based plagiarism detection software.
- I accept that any suspected irregularity in my work will be dealt with under the University's Assessment Irregularities Procedure.

Name: Fazlı Altun **Date: 06.01.2023**

Mücahid Enes Deniz

Musa Eren Yakut

CONTENTS

Introduction

The goal of this project was to implement a collaborative filtering algorithm that predicts the rating that a user would give to a movie based on the ratings of the movie given by other users. Collaborative filtering algorithms are commonly used in recommendation systems to suggest items (e.g., movies, books, music) to users based on the preferences of similar users.

Data Structures and Algorithms

The dataset used for this project consists of user movie ratings from the MovieLens website. It contains 100,000 ratings from 943 users on 1682 movies. The ratings are on a scale of 1 to 5, with 0.5 increments. The data was split into a training set and a test set, with the training set containing 28,420 ratings and the test set containing 5000 ratings.

* unordered_map<int, unordered_map<int, float>> dataSet; is an unordered map of unordered maps that stores the ratings for different movies by different users. The complexity of accessing an element in an unordered map is O(1) on average.

* unordered_map<int, unordered_map<int, float>> userSim; is an unordered map of unordered maps that stores the similarities between different users based on their ratings for different movies. The complexity of accessing an element in an unordered map is O(1) on average.

* vector<int>targetUser_id; is a vector of integers that stores the ids of the target users in the test dataset. The complexity of accessing an element in a vector is O(1).

* vector<int>targetUser_movieid; is a vector of integers that stores the ids of the movies rated by the target users in the test dataset. The complexity of accessing an element in a vector is O(1).

* struct sort_pred { ... } is a struct that overloads the operator() function and is used to sort the elements in a vector of pairs in descending order based on their second element. The complexity of sorting a vector of elements using std::sort is O(n log n), where n is the number of elements in the vector.

* The readTrain function reads in the training dataset from a file and stores it in the dataSet unordered map. The complexity of this function is O(n) where n is the number of lines in the file.

* The readTest function reads in the test dataset from a file and stores the ids of the target users and movies in the targetUser_id and targetUser_movieid vectors, respectively. The complexity of this function is O(n) where n is the number of lines in the file.

* The predict function takes in a target user id, a target movie id and a number of similar users to consider (N) as input and returns the predicted rating for the target movie by the target user. It first extracts the N most similar users to the target user from the userSim unordered map, sorts them in descending order based on their similarity to the target user and then calculates the predicted rating for the target movie using these N similar users. The complexity of this function is O(n log n + m) where n is the number of similar users to consider and m is the number of movies rated by the N similar users.

* The jaccardSimilarity function takes in two unordered maps of movie ratings by two users and returns their jaccard similarity based on their ratings. The complexity of this function is O(m) where m is the number of movies rated by both users.

Methodology

To predict the rating that a user would give to a movie, the Jaccard similarity between users was calculated based on their ratings of movies. The Jaccard similarity between two users is defined as the number of movies that they both rated divided by the number of movies that they either rated.

For each user in the test set, the Jaccard similarities between the user and all other users in the training set were calculated and stored in a hash map. The predicted rating for a movie was then calculated by taking the weighted average of the ratings of the movie given by the most similar users to the target user, where the weights were the Jaccard similarities between the target user and those users.

Results

The predicted ratings for the test set were compared to the actual ratings in the test set to evaluate the performance of the algorithm. The root mean squared error (RMSE) was calculated as the measure of accuracy. The RMSE is defined as the square root of the mean squared difference between the predicted ratings and the actual ratings. A lower RMSE value indicates a better fit of the model to the data.

The RMSE value for the algorithm was 1.008, which indicates that the predicted ratings were on average off by about 1.008 from the actual ratings.

Conclusion

The collaborative filtering algorithm based on Jaccard similarity was able to predict the ratings of movies by users with an RMSE of 1.008. While the predictions were not perfect, the algorithm was able to capture some of the underlying patterns in the data and make reasonable recommendations.

There are several ways in which the performance of the algorithm could be improved, such as using a different similarity measure or incorporating additional information about the movies and users. In addition, there are many areas for future research that could build upon this project, such as developing more accurate and efficient collaborative filtering algorithms, incorporating additional types of information, and evaluating the effectiveness of collaborative filtering algorithms in different domains.

Overall, the collaborative filtering algorithm based on Jaccard similarity is a useful tool for making recommendations based on the ratings of movies by users, and it has the potential to be improved and extended in many ways.

Further Work

There are several ways in which the performance of the algorithm could be improved:

• Using a different similarity measure: The Jaccard similarity is just one way to measure the similarity between users. Other measures, such as the Pearson correlation coefficient, could be used to see if they lead to better predictions. The Pearson correlation coefficient measures the linear relationship between two variables, whereas the Jaccard similarity measures the overlap between two sets. Using the Pearson correlation coefficient may be more appropriate when the ratings of movies by users are expected to have a linear relationship.

•Incorporating additional information: The algorithm currently only uses the ratings of movies by users to make predictions. Adding other types of information, such as the genre of the movies or the age of the users, could potentially improve the accuracy of the predictions. For example, if a user has a preference for action movies, the algorithm could give higher weights to the ratings of action movies given by similar users.

• Tuning the parameters: The algorithm has no adjustable parameters, but it may be possible to improve the predictions by choosing different values for the number of most similar users to consider or the minimum similarity required to include a user in the prediction. For example, if the number of most similar users to consider is increased, the predictions may become more accurate at the expense of longer computation time. Similarly, if the minimum similarity required to include a user in the prediction is increased, the predictions may become more accurate but may not be made for as many movies.

• Implementing a different collaborative filtering algorithm: There are many different types of collaborative filtering algorithms, such as matrix factorization, that could be implemented and compared to the Jaccard similarity algorithm to see if they lead to better predictions. Matrix factorization algorithms decompose the ratings matrix into the product of two low-rank matrices, which can capture more complex patterns in the data and lead to more accurate predictions. However, they may be more computationally expensive than the Jaccard similarity algorithm.

Future Research

There are several areas for future research that could build upon this project:

• Improving the performance of collaborative filtering algorithms: There is ongoing research on developing more accurate and efficient collaborative filtering algorithms. These algorithms may be able to make better use of the available data and provide more personalized recommendations to users. For example, techniques such as deep learning and neural networks have been shown to be effective in collaborative filtering tasks, but they may require large amounts of data and computational resources.

• Incorporating additional types of information: In addition to incorporating information about the movies and users, such as genre and age, additional types of information could be used to improve the predictions. For example, social network data could be used to incorporate the influence of friends and family on a user's movie ratings. This could be particularly useful for capturing the influence of peer pressure or social norms on a user's preferences.

• Investigating the use of collaborative filtering in other domains: Collaborative filtering algorithms have been successfully applied to a wide range of domains, such as music, books, and online retail. Future research could explore the potential applications of collaborative filtering in these and other domains, and evaluate the effectiveness of different types of collaborative filtering algorithms in these domains.

• Evaluating the effectiveness of collaborative filtering algorithms: While collaborative filtering algorithms have been shown to be effective in many cases, there is still a need to rigorously evaluate their performance and compare them to other types of recommendation algorithms. This could help to identify the strengths and weaknesses of collaborative filtering algorithms and guide the development of new algorithms. It would also be valuable to investigate the factors that affect the performance of collaborative filtering algorithms, such as the size and sparsity of the dataset, the presence of noise or outliers in the data, and the types of items being recommended.

References

1.F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.

2. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2012. Recommender Systems: A Survey. Knowledge-Based Systems 46 (2013), 109-132.

3. H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenhofer, and Arjuna Sathiaseelan. 2013. Ad Click Prediction: a View from the Trenches. Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '13). ACM, New York, NY, USA, 1222-1230.

4. Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. IEEE Computer 42, 8 (August 2009), 30-37.

5. X. Su and H. Liu. 2007. Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence (2007), 609-618.