TOWARDS TIME-AWARE COLLABORATIVE FILTERING RECOMMENDATION SYSTEM

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Dawei Wang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

TABLE OF CONTENTS

Page

LIST OF TABLES vi

LIST OF FIGURES xii

ABSTRACT xvi

1 INTRODUCTION 1
- 1.1 Motivation 1
- 1.2 Organization 4

2 LITERATURE REVIEW 6
- 2.1 An overview of recommendation system 6
  - 2.1.1 Content-based filtering 7
  - 2.1.2 Collaborative filtering (CF) 8
  - 2.1.3 Hybrid 10
- 2.2 Recommendation with Collaborative Filtering 10
  - 2.2.1 Notation 10
  - 2.2.2 Memory-based methods 13
  - 2.2.3 Model-based methods 23
- 2.3 Time-aware recommendation systems 25

3 IMPROVING NEIGHBOR-BASED COLLABORATIVE FILTERING BY USING A HYBRID SIMILARITY MEASUREMENT 28
- 3.1 Abstract 28
- 3.2 Introduction 29
  - 3.2.1 Recommendation systems 30
  - 3.2.2 Characteristics and Challenges of Collaborative filtering 34
  - 3.2.3 Summary of Main Contributions 36
- 3.3 Background 37
  - 3.3.1 Notation 38
  - 3.3.2 Item-Based Similarity Computation 41
  - 3.3.3 Preference Prediction 44
- 3.4 Proposed Approach 45
  - 3.4.1 Choice of structural and rating-based similarity measurements . 46
  - 3.4.2 Improvements over using rating or structural similarity alone . . 48
  - 3.4.3 Amplification parameter α 49
- 3.5 Experimental Setup 50
  - 3.5.1 Data Sets 50

Page

3.5.2 Accuracy Evaluation 52
3.5.3 Hubness Evaluation 53
3.6 Experimental Results 54
3.6.1 Prediction Accuracy 55
3.6.2 Computational Resources 63
3.6.3 Performance without cold-start users 64
3.7 Discussion 65
3.7.1 Prediction Accuracy 65
3.7.2 Computational Resources 67
3.7.3 Performance without cold-start users 68
3.8 Conclusion 68

4 IMPROVING MEMORY-BASED COLLABORATIVE FILTERING BY CONSIDERING CONTRAST EFFECTS, ANCHORING AND ADJUSTMENT BEHAVIORS 69
4.1 Abstract 69
4.2 Introduction 69
4.2.1 Time-aware Recommendation Systems 69
4.2.2 Summary of Main Contributions 70
4.3 Background 71
4.3.1 Related works 71
4.3.2 Memory-based collaborative filtering 73
4.4 Proposed Approach 73
4.4.1 Reducing contrast effects in rating-based similarity measurements 74
4.4.2 Ignoring relationship between items rated long time apart in structural similarity measurements 76
4.4.3 Hubness 78
4.4.4 Anchoring and Adjustment 84
4.5 Experiment Design 86
4.5.1 Data Sets 86
4.6 Experimental Results 86
4.6.1 Time affecting prediction accuracy 86
4.6.2 Reducing contrast effects in rating-based similarity measurements 90
4.6.3 Ignoring relationship between items rated long time apart in structural similarity measurements 94
4.6.4 Coverage and Hubness 99
4.6.5 Anchoring and Adjustment 103
4.7 Discussion 106
4.8 Conclusion 108

5 QUANTIFICATION AND VISUALIZATION OF USER MOVIE GENRE SHIFTS AND DYNAMIC RECOMMENDATION USING COLLABORATIVE FILTERING 109

Page
5.1 Introduction 109
5.2 Method 110
5.2.1 Sampling 110
5.2.2 Visualization 112
5.2.3 Quantification 112
5.3 Experiment 114
5.4 Dynamic Recommendation List - Future Work 119
5.4.1 Future Expected Setup and Expected Results 120
Bibliography 122
A Amplification Parameter 138
B MAE v.s. RMSE 142

LIST OF TABLES

2.1 Notation 11

3.1 Notation 38

3.2 Lists of some structural similarity measurements approximated locally. Those measurements do not require global information such as the total number of users. All information required for the calculation can be gathered from those 2 associated vectors $\boldsymbol{i}_c$ and $\boldsymbol{i}_i$. 47

3.3 Data sets. Datasets vary in different sizes (number of users, number of items, number of ratings, and density). MovieLens have been preprocessed-users who had less than 20 ratings or did not have complete demographic information were removed [115]. MovieLens 100K dataset contains data collected through MovieLens web site (movielens.umn.edu) from 1997/09/19 through 1998/04/22. MovieLens 1M dataset contains data collected during from 2000/04/25 through 2003/02/28. Netflix dataset contains data collected from 1998/10 through 2005/12 and reflects all ratings received during this period. Density is the ratio of the number of actual ratings to the possible maximum number of ratings 50

3.4 Comparison of prediction accuracy among state-of-the-art CFs on ML-100K dataset. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119]. 58

3.5 Comparison of prediction accuracy among state-of-the-art CFs on ML-1M dataset. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119]. 58

3.6 Comparison of prediction accuracy among state-of-the-art CFs on Netflix probe testing dataset. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119]. 60

3.7 Comparison of prediction accuracy among state-of-the-art CFs on Netflix dataset with training dataset containing no ratings from the testing dataset. The table is ranked by MAE. The proposed method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119]. 61

3.8 Comparison of hubness and rating conditioned hubness between rating and structural based similarity measurements on MovieLens and Netflix dataset. Both the hubness and rating conditioned hubness decrease as the size of the neighbor list increases, with K=20, the rating conditioned hubness H_R^{20} of combined similarity measurement is approximately 2 times lower than adjusted cosine similarity measurement across all 3 datasets. . . 61

3.9 Comparison of prediction accuracy among state-of-the-art CFs on Netflix probe testing dataset for users with more than 50 ratings. The table is ranked by MAE. The proposed method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119]. 62

3.10 Comparison of prediction accuracy among state-of-the-art CFs on Netflix dataset with training dataset containing no ratings from the testing dataset for users with more than 50 ratings. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119]. 63

3.11 Comparison of time consumption between memory-based and model-based algorithms on ml-1M dataset. We choose one method that performs well in accuracy prediction from previous section since other methods are derived from those. We can see that memory-based CFs are much faster than model-based CFs. Model-based CF Library provided by [119]. Note that: all methods in this section are single-threaded coded. No parallelization is done for any method. 63

3.12 Prediction accuracy and coverage with bottom-K cutoff. Bot-K is the number of minimum neighbors used to make a prediction. ItemKNNCombined is using K=25, α=2.6. 64

4.1 Lists of some time-aware structural similarity measurements approximated locally. $|C_{ic}^{t}|$ denotes the pairs of items that are rated by the same user within a certain natural time distance or item distance. Note that those measurements do not require global information such as the total number of users. All information required for the calculation can be obtained from those 2 associated vectors $\boldsymbol{i}_c$ and $\boldsymbol{i}_i$. 77

4.2 Data sets. Datasets vary in different sizes (number of users, number of items, number of ratings, and density). MovieLens datasets have been pre-processed with users who had less than 20 ratings or did not have complete demographic information were removed [115]. ML100K dataset contains data collected through MovieLens web site (movielens.umn.edu) from 1997/09/19 through 1998/04/22. MLNew contains data collected between March 29, 1996 and September 24, 2018. Netflix datasets are selected sub-dataset from Netflix Prize dataset which contains data collected from 1998/10 through 2005/12 and reflects all ratings received during this period. Netflix contains users who have rated at least 20 items and been active for more than 1 day. Netflix257 contains less ratings with less users compared to Netflix2761. Density is the ratio of the number of actual ratings to the possible maximum number of ratings. 87

4.3 Prediction accuracy of memory-based and model-based CF performed on 4 datasets: $S_{U>1}$: users active for more than 1 day, $S_{U\leq1}$: users active for less than 1 day, S_{U+}: users with more ratings, S_{U-}: users with less ratings. By comparing the MAE of S_{U+} and S_{U-}, more number of ratings can potentially contribute to better prediction accuracy of CFs(but not necessarily, by comparing the MAE of S_{U-} and $S_{U\leq1}$ of SVD++). By comparing the MAE of $S_{U>1}$ and S_{U+} users who are active for longer time have better prediction accuracy than users who have more ratings. 89

4.4 Results for enlarging contrast effects in rating-based similarity measurements. Traditional memory-based CF with adjusted cosine can now perform better than sophisticated model-based CF SVD++. Memory-based CF with our combined similarity measurement improves 1% in terms of lower MAE compared to S_{U+} Combined in Table 4.3. 91

4.5 MAE of combined+ on Netflix257 dataset with α=2.6, topN=30: best prediction accuracy is achieved by setting β close to 0 to neglect similarities of pairs of items rated close to each other(within a time distance, or within an item distance) in final similarity aggregation. Using item distance as selecting criteria performances better than using time distance. 92

4.6 MAE of Ochiai with structural similarity cutoff on Netflix257 dataset with α=2.6, β=0.01, topN=30. 94

4.7 MAE of combined+ with structural similarity cutoff on Netflix257 dataset with α=2.6, topN=30. There is only an improvement of 0.3% in MAE by using dynamic time distance. 96

4.8 MAE of combined+ with structural and rating similarity cutoff on Netflix257 dataset with α=2.6, β=0.01(itemDist=4), topN=30. 98

4.9 Comparison of hubness between rating-based, structural-based and combined similarity measurement with and without considering time effects. Combined+ with both techniques achieves the lowest hubness. 101

4.10 Comparison of prediction accuracy (MAE) of CFs using PCC v.s. Adjusted Cosine over the ML100K and Netflix257 datasets. CF with PCC performs better over Netflix257, Netflix2761 and MLNew, which users know item averages when rating. CF with adjusted cosine performs better over ML100K which users don't know item averages when rating. . . 105

5.1 Average step size and predictability of each user: MDS successfully preserves the distance between points as the largest and least shifts are the same as calculated in original space and reduced space. User 4 shifts the most with the largest average shifting step size. But the shifting is the most predictable as the predictability of user 4 is ranked the highest. . . 117

A.3 Choice of amplification parameter α. CASE 1: If α is too large, prediction is biased towards the opinions of the first few neighbor items. CASE 2: If α is too small, prediction can be biased towards very few users since the ranking can be determined by rating-based similarity measurement which could be calculated based on very few users. CASE 3: Ranking is determined mainly by structural similarity while allowing rating-based similarity to adjust on a smaller scale. 140

A.4 MAE v.s. RMSE with increasing error variance. MAE stays the same while RMSE increases as the frequency distribution of magnitudes of error variance increases. Large errors will result a higher RMSE. 141

B.1 MAE v.s. RMSE with different variance of errors. RMSE stays the same while MAE differs with error variance of case 4 is greater than that of case 5.142

LIST OF FIGURES

2.1 Illustration of item-user network and Item-based network: From item-user network as in Figure 2.1a, we draw the item-based network as in Figure 2.1b13

3.1 Illustration of item-user network and Item-based network: From item-user network as in Figure 3.1a, we draw the item-based network as in Figure 3.1b40

3.2 Density Distribution of Ratings of MovieLens, Netflix, where x-axis is the rating from 1 to 5, y-axis is the density of number of users who rated given rating. 51

3.3 Degree Distribution of Items of MovieLens, Netflix, where x-axis is the degree of the number of users, y-axis is the density of number of items gets rated by given number of users. Rating-based similarity measurement will not be able to capture and utilize those structural information. 52

3.4 MAE performance for different size of nearest neighbor list and case amplification parameter, where x-axis is the power (case amplification parameter) raised on structural similarity measurement, y-axis is the size of the nearest neighbor list, z-axis is the MAE(Mean Absolute Error). We can see MAE follows a concave shape respect to K and α. We choose the parameters with MAE at the lowest point. 56

3.5 Accuracy performance for different nearest neighbor list size and amplification parameter. Those two figures are transections of Figure 3.5. 57

3.6 Prediction accuracy with different nearest neighbor size K 59

3.7 Accuracy Performance For Different Bot-K Cutoff 65

4.1 The scores of ratings affected by the sequence of ratings. Ratings given may be anchored and adjusted based on user memories. In Fig 4.1a, User X rated item 3, and later found that item 4 and item 5 were better, but the highest rating this user can give is 5. So the user gave item 4 and 5 with ratings of 5 as well. While in Fig. 5.6a, User X rated the best item 5, and then found item 4 and 3 were not as good as item 5, so he gave lower ratings to those items. The score of the ratings are affected by the sequence of ratings. 74

4.2 Plot of $y = z_i \left((|\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}| + 1)^{\beta} - 1 \right)$. Contrast effects can be enlarged or reduced by choosing different β. when $\beta > 1$, the contrast effects are enlarged, when $\beta < 1$, the contrast effects are reduced. 76

4.3 Static average v.s. dynamic average. Using constant user average as the reference point, item 1 and item 2 will yield a negative relationship, since item 1 is above the average while item 2 is below. Using dynamic user average as the reference point, item 1 and item 2 will yield a positive relationship. 85

4.4 Distribution of time distance and item distance of all pairs of items rated by the same user over ML100K dataset 93

4.5 Distribution of time distance and item distance of all pairs of items rated by the same user over Netflix257 dataset 93

4.6 MAE of Ochiai with structural similarity cutoff on Netflix257 dataset with β=0.01, topN=30. Using a dynamic item distance with 20 items yield the lowest MAE of 0.704031, which is about 1.4% lower than static Ochiai without considering time. 95

4.7 MAE of combined+ with structural similarity cutoff on Netflix257 dataset with α=2.6, topN=30. 97

4.8 MAE of combined+ with structural and rating similarity exclusion on Netflix257 dataset with α=2.6, β=0.01(itemDist=4), topN=30. The MAE is improved by 1.3% over the combined method with neglecting items rated close to each other within item distance of 4 in rating-based similarity measurement and 2.7% over the static combined method. 99

4.9 Comparison over Hubness (H_R^K) of KNN list with similarities randomly and strategically exclusion. Combined+ (both) achieves the lowest hubness. Hubness of Ochiai+ is less than Ochiai with random exclusion, while there is no significant difference in hubness between strategically and random exclusion over rating-based AdjCos (random exclude 14%) and AdjCos+. 102

4.10 Comparison of prediction accuracy and hubness (H_R^5) of KNN list with Ochiai+ and Ochiai with different density of pairs of items excluded. To achieve the same hubness of Ochiai with 14% pairs excluded, Ochiai needs to remove at least 80% of pairs. However, Prediction accuracy (MAE) of Ochiai is dramatically decreased after excluding 50% of pairs. 103

4.11 MAE of Scene building techniques. #items means the number of items excluded as warm up period in accumulative average calculation in PCC, AdjCos methods, and number of items included in AdjCosRecent methods. There isn't much difference in MAE of CFs using similarity measurements with dynamic averages compared to static averages. 105

5.1 Time sampling: the ratings are sampled during sampling windows once every skipping windows. The warm up period is also skipped as users may rate lots of movies they have seen before at once at the beginning. But the time he watched the movie may be a long time ago. 111

5.2 User switching movie genres over time. X-axis is the time in week, week 0 is the date when the user started to rate in the database. Y-axis is the density of movie genres. Those are two users selected randomly from the MovieLens100K [115] dataset to show two different shifting patterns. In Figure 5.2a, at different time period, the user watched movies belonging to different genres, which indicates a user preference shift over time. . . 115

5.3 MDS plot for user genre preference shifts: Each dot is colored, labeled by user from 1 to 9, connected sequentially with arrows pointing in chronological order. Users don't shift too much such as user 2, while some users shift a lot such as user 4 based on distances between dots. 116

5.4 Shepard plot for 2D MDS: most points are spread around the fitted function, which indicates a good fit of the 2D MDS 118

5.5 Stress with different dimensions: 2D MDS gives us a good fit with stress under 15% 118

5.6 User preference shifts comparison: the changes of the proportions of each genre for user 2 is not as dramatic as user 4. But user 4 is shifting at some pattern such as the proportion of action movies are watched more at a steady increasing rate. 119

A.1 MAE of adjusted cosine combined with different structural similarity measurements with $\alpha = 1$ over ML-100K dataset. The performance of the combination of adjusted cosine and structural similarity measurement works better than using adjusted cosine or structural similarity measurement alone. 138

ABSTRACT

Wang, Dawei PhD, Purdue University, August 2020. Towards Time-aware Collaborative Filtering Recommendation System. Major Professors: Yuehwern Yih, Mario Ventresca.

As technological capacity to store and exchange information progress, the amount of available data grows explosively, which can lead to information overload. The difficulty of making decisions effectively increases when one has too much information about that issue. Recommendation systems are a subclass of information filtering systems that aim to predict a user's opinion or preference of topic or item, thereby providing personalized recommendations to users by exploiting historic data. They are widely used in e-commerce such as Amazon.com, online movie streaming companies such as Netflix, and social media networks such as Facebook. Memory-based collaborative filtering (CF) is one of the recommendation system methods used to predict a user's rating or preference by exploring historic ratings, but without incorporating any content information about users or items. Many studies have been conducted on memory-based CFs to improve prediction accuracy, but none of them have achieved better prediction accuracy than state-of-the-art model-based CFs. Furthermore, A product or service is not judged only by its own characteristics but also by the characteristics of other products or services offered concurrently. It can also be judged by anchoring based on users' memories. Rating or satisfaction is viewed as a function of the discrepancy or contrast between expected and obtained outcomes documented as contrast effects. Thus, a rating given to an item by a user is a comparative opinion based on the user's past experiences. Therefore, the score of ratings can be affected by the sequence and time of ratings. However, in traditional CFs, pairwise similarities measured between items do not consider time factors such as the sequence of rating, which could introduce biases caused by contrast effects. In this research, we proposed a new approach that combines both structural and rating-based similarity measurement used in memory-based CFs. We found that memory-based CF using combined similarity measurement can achieve better prediction accuracy than model-based CFs in terms of lower MAE and reduce memory and time by using less neighbors than traditional memory-based CFs on MovieLens and Netflix datasets. We also proposed techniques to reduce the biases caused by those user comparing, anchoring and adjustment behaviors by introducing the time-aware similarity measurements used in memory-based CFs. At last, we introduced novel techniques to identify, quantify, and visualize user preference dynamics and how it could be used in generating dynamic recommendation lists that fits each user's current preferences.

1. INTRODUCTION

1.1 Motivation

As technological capacity to store and exchange information progress, the amount of available data grows explosively, which can lead to information overload. The difficulty of making decisions effectively increases when one has too much information about that issue [1]. The world's technological capacity to store information grew from 2.6 exabytes (optimally compressed) in 1986 to 295 exabytes (optimally compressed). This is equivalent to less than one 730-MB CD-ROMs per person in 1986, and almost 61 CD-ROMs per person in 2007 [2]. The world's technological capacity to receive information through one-way broadcast networks was 432 exabytes (optimally compressed) of information in 1986, and 1,900 (optimally compressed) in 2007 [2]. The world's technological capcacity to exchange information through two-way telecommunication networks was 0.281 exabytes (optimally compressed) of information in 1986, and 65 (optimally compressed) in 2007 [2].

An information filtering system is a system that removes redundant or useless information from information steam using automated methods to let users manage information overload. Recommendation systems are a subclass of information filtering systems that aim to predict a user's opinion or preference of topic or item, thereby providing personalized recommendations to users by exploiting historic data. They are widely used in e-commerces such as Amazon.com [3], online movie streaming companies such as Netflix [4], and social media networks such as Facebook [5]. With a large amount and diversity of products, a recommendation system could also help streaming service providers or online vendors provide users with recommendations that are specific to their preferences. This could improve user experience in searching for items or services and potentially lead them to make more purchases, watch more movies, or subscribe to more services. For examples, data gathered for three weeks in the summer of 2001 showed that between 20% to 40% of sales on Amazon are due to recommended products that do not belong to the shop's 100,000 most sold products [6], and 60% of movies rented by Netflix are selected based on personalized recommendations[1]. Furthermore, a recommendation system could generate not only more direct revenue, but also additional revenue by introducing shoppers to new categories [7]. Hence, a recommendation system can significantly impact a company's revenue [8]. Note that 1% improvement in prediction accuracy in terms of MAE(Mean Absolute Error) and RMSE(Root Mean Squared Error) may be a small number, but could result in a significant difference in the ranking of the "top-10" most recommended movies for an individual user [9].

Recommendation systems typically generate a list of recommendations to users in one of three ways [10] - (1) collaborative filtering, (2) content-based filtering, or (3) a hybrid of those two approaches. Collaborative filtering (CF) analyzes historical data about user behavior to predict what they might like by learning interactions between users and items. Content-based filtering predicts by learning descriptions of items or profiles of users. Collaborative filtering can be memory-based or model-based. Memory-based approaches rely on pairwise similarities between vectors of item-user rating matrix while model-based approaches rely on factorizing the entire rating matrix. Both memory-based and model-based CFs can be implemented from an item-based or user-based perspective.

A product or service is not judged only by its own characteristics but also by the characteristics of other products or services offered concurrently [11] or by anchoring based on users' memories [12–14]. Rating or satisfaction is viewed as a function of the discrepancy or contrast between obtained and expected outcomes [15,16]. This is documented as contrast effects [17]. Thus, the score of rating can be affected by the sequence of rating. However, in traditional collaborative filtering, pairwise similarities

[1]As presented by Jon Sanders (Recommendation Systems Engineering, Netflix) during the talk "Research Challenges in Recommenders" at the 3rd ACM Conference on Recommender Systems (2009).

measured between items do not consider the sequence of rating, which could introduce biases caused by contrast effects.

Moreover, It is well known that consumers' preferences can shift over time [18–25]. This can be due to (1) *exploration* of new items, instead of repeatedly interacting with the same items [22]. New items may continuously enter the market as well. (2) *experience or tendency* to interact with items they have previously had positive interactions with, and to stop interacting with items they have previously negative interactions with [22]. (3) *popularity* of the items, irrespective of personal interaction history [22]. (4) *social influence* may cause preference changes as a result of observing the preference changes of friends [22, 25].

Unfortunately, traditional recommendation systems learn users' preferences from a static dataset, which implies that the system is unaware of any new or developing user preferences. One approach to overcome this problem is to update the dataset at predefined intervals (e.g., every week or month), but this strategy still leads to missed opportunities between updates. Moreover, learning from an updated dataset still requires sufficient user ratings to cause sufficient changes in the system model before changes in preference (and thus recommendations) are detected.

Although classic recommendation methods learn over a static dataset, collaborative filtering (one of the most widely used methods in recommendation systems) has been previously studied with temporal dynamic-based modifications. One approach is to model the users and items in a network and use only the most recent ratings of user or item neighbors to predict ratings or suggest recommendations [26]. Another idea uses different K values for K-Nearest Neighbor algorithms over time, using whichever K that minimizes the Root-Mean-Square-Error (RMSE) or other error measure [27]. Some other methods use a decay weight on old rated items or the relation between users and items to decrease the importance of older data versus newer data [28–30]. In [31], a more sophisticated model-based approach was proposed, but it required a high computation time cost because their algorithm added a time dimension to each model-based parameter, which is already computationally and data-intensive to train.

Without introducing a high computational burden, our goal is to modify memory-based collaborative filtering by devising a new measure of user preference changes such that recommendations are more reflective of users' recent preferences. We will test our approach using the most referred dataset MovieLens , as well as other standard benchmarks such as that from Netflix [4]. In these contexts, the changes in question could be by genre in a movie or music context, or a brand loyalty change. The project is broken into two steps (1) modifying memory-based collaborative filtering to include this dynamic information, and (2) modeling and estimating preference changes. Upon completion, the system can be used for a number of applications. For instance, we can test predictability of recommendations made before users are even aware of their own desires. Also, we may be able to discover patterns in group preference changes earlier than other approaches. Therefore, we may be able to make accurate recommendations even before a user develops new preferences or are about to switch back to old preferences.

1.2 Organization

The thesis is organized as follows:

In Chapter 2, an overview of background knowledge of recommendation systems, especially memory-based Collaborative filtering is provided.

In Chapter 3, we propose a new approach that combines both structural and rating-based similarity measurement to be used for memory-based collaborative filtering. We found that memory-based CF using combined similarity measurement can achieve better prediction accuracy than model-based CFs in terms of lower MAE and reduce memory and time by using less neighbors than traditional memory-based CFs on MovieLens and Netflix datasets.

In Chapter 4, we propose a time-aware similarity measurement that approximate relationship between items and users by using information that are only available at the time to the users when this relationship is formed to be more accurately representative of the relations between items from the perspective of a particular user.

In Chapter 5, we propose an approach to identify, quantify, and visualize user preference shifts and ideas based on collaborative filtering that considers user preference changes such as category preference shifts to generate recommendations that fit users' most recent preferences.

2. LITERATURE REVIEW

2.1 An overview of recommendation system

Recommendation systems are a subclass of information filtering systems that aim to predict a user's opinion or preference of a topic or item, thereby providing personalized recommendations to users by exploiting historic data. They are widely used in e-commerces such as Amazon.com [3], online movie streaming companies such as Netflix [4], and social media networks such as Facebook [5]. With a large amount and diversity of products, a recommendation system could also help streaming service providers or online vendors provide users with recommendations that are specific their preference. This could improve user experience in searching for items or services and potentially lead them to make more purchases, watch more movies, or subscribe to more services. For examples, data gathered for three weeks in the summer of 2001 showed that between 20% to 40% of sales on Amazon are due to recommended products that do not belong to the shop's 100,000 most sold products [6], and 60% of movies rented by Netflix are selected based on personalized recommendations[1]. Furthermore, a recommendation system could generate not only more direct revenue, but also additional revenue by introducing shoppers to new categories [7]. Hence, a recommendation system can significantly impact a company's revenue [8].

Recommendation systems were first mentioned in 1990 by Jussi Karlgren at Columbia University [32], and implemented later from 1994 onwards by Jussi Karlgren at SICS Research Report [33] and research groups led by Pattie Maes at MIT [34], Will Hill at Bellcore [35], and Paul Resnick [36, 37], also at MIT whose work with GroupLens was awarded the 2010 ACM Software Systems Award. GroupLens lab at University

[1]As presented by Jon Sanders (Recommendation Systems Engineering, Netflix) during the talk "Research Challenges in Recommenders" at the 3rd ACM Conference on Recommender Systems (2009).

of Minisota, Twin Cities was one of the first to study automated recommendation systems. GroupLens' popular "MovieLens" movie recommendation site and dataset is one of the most referred datasets in recommendation systems.

Recommendation systems typically generate a list of recommendations to users in one of three ways [10] - (1) collaborative filtering, (2) content-based filtering, or (3) a hybrid of those two approaches. Collaborative filtering (CF) analyzes historical data about user behavior to predict what they might like by learning interactions between users and items. Content-based filtering predicts by learning descriptions of items or profiles of users. Collaborative filtering can be memory-based or model-based. Memory-based approaches rely on pairwise similarities between vectors of item-user rating matrix, while model-based approaches rely on factorizing the entire rating matrix. Both memory-based and model-based CFs can be implemented from an item-based or user-based perspective.

2.1.1 Content-based filtering

These approaches use keywords or phrases to describe the contents of items, build user profiles to indicate the types of contents each user prefers using those keywords(phrases), and then recommend a list of items that fits each user's preference. Several techniques have been studied such as pLSA(Probabilistic latent semantic analysis) [38] [39], LDA(Latent Dirichlet Allocation) [40], etc. While content-based methods incorporate descriptive information from items by characterizing using keywords, they do not necessarily incorporate interactions between other users or items. Recommendations are made based solely on the content information of objects that the target user has rated in the past [8]. Content-based filterings are widely used in a variety of domains ranging from recommending webpages, news articles, restaurants, etc [41]. For examples: Pandora Radio[2] recommends users with songs that share similar characteristics [42], and Rotten Tomatoes[3] recommends users with movies that

[2]https://www.pandora.com
[3]https://www.rottentomatoes.com

share similar casts and storylines. However, Content-based filtering is unable to make a good recommendation that matches a user's preference if the profiles of users or descriptions of items do not contain sufficient information to tell if the user likes or dislikes the item [41]. Generally, in some domain such as movies, restaurants, items are not amenable to any useful feature extraction methods with current technology. Only a very shallow analysis of certain kinds of content can be supplied [43].

2.1.2 Collaborative filtering (CF)

These approaches analyze historical data on user activities, and use it to predict what they might like based on their similarities to other users, or to items that are similar to the ones the user is known to like [44]. A key advantage of this approach is that CFs study only the interactions between individuals without incorporating feature or attribute information of items and users. Thus, they don't require knowledge about the actual context of the data in order to make recommendations. That is, they can make a prediction without "understanding" a movie, a friend, or a music, etc. Thus, this approach can be applied broadly regardless of the contents of the data. However, when users haven't rated a sufficient number of items, these methods may not perform very well (known as the cold start problem [45] [46]). Collaborative Filtering could also suffer from scalability or sparsity issues [47], (details are listed in Section 3.2.2).

Collaborative filtering can be implemented in two ways: user-based or item-based. To make a prediction of how a user u would rate an item i, User-based CF aggregates the opinions about item i from users that are similar to user u. It assumes that if two persons share similar opinions on some items, they are likely to hold similar opinions on other items as well. On the other hand, item-based CF aggregates the opinions about the items that are similar to item i, and have been rated by user u. It assumes that users are likely to hold the same opinions on similar items. For both item-based and user-based CFs, two approaches have been studied:

1. Memory-based. This approach calculates similarities between items or users and uses them as weights on ratings to represent how much the opinions on items a user has rated can represent a user's opinion on the unrated item. In general, it is effective and easy to implement. A typical example of this approach is K-Nearest-Neighbor (KNN) [48] [47]. First, item-based memory-based CFs find similar items by calculating pairwise similarities between the predicting item and all other items the user has rated. These pairwise similarities are used to rank how representative of the predicting item each other item is. Therefore, the prediction accuracy of collaborative filtering algorithms is highly dependent on how accurate the similarity measurement is. Second, based on pair-wise similarity measurement, K most similar items to this predicting item are selected from the items that this user has already rated on and then it combines the user's opinions of those K items by weighted average or weighted sum to predict a rating for the unrated item from the user. Such approaches are widely used due to their simplicity, explainability and effectiveness [49], and predictions can be made in real time as new rating data is added. However, its prediction accuracy decreases when few items have been rated. Its scalability is also limited for large datasets [8].

2. Model-based. This approach uses data mining and machine learning algorithms to develop and train predictive models. There are many different algorithms such as Singular-Value Decomposition (SVD) [50], and Principal component analysis (PCA) [51], which use matrix factorization techniques. Dimensionality reduction is usually used through model-based approach to improve the scalability and accuracy. It addresses the sparsity and scalability problems and performs better in prediction accuracy in comparison to memory-based CFs [10]. But, the models usually use iterative methods to estimate the parameters for the models, which take more time to build and train compared to memory-based approaches. Moreover, they could lose useful information as a result of dimensionality reduction [10].

2.1.3 Hybrid

This strategy is to combine two or more techniques [52] [53] [54] [55], or with other techniques such as deep learning [56] or clustering [57] to overcome the limitations of their individuals and improve the performance such as prediction accuracy, scalability. However, it increases the computational complexity such as time and resources(memory) required [10].

A complete introduction to all available recommendation systems is beyond the scope of this thesis, in the next section, we will elaborate more on techniques of collaborative filtering since CF is the focus of this preliminary proposal.

2.2 Recommendation with Collaborative Filtering

2.2.1 Notation

Before introducing the underlying algorithms of recommendation systems, we define some terms to better explain CF algorithms through this thesis.

Table 2.1.: Notation

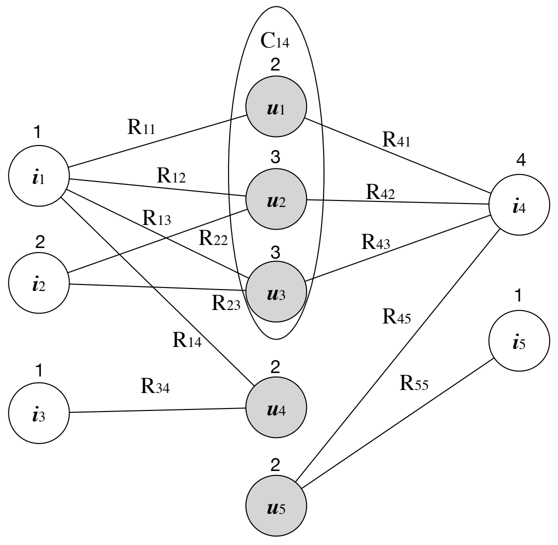
Notation	Meaning
m	total number of users
n	total number of items
$\mathbf{R}$	n-by-m item-user matrix containing ratings from users to items
$\mathbf{R}_{ij}$	rating of item i from user j , $\mathbf{R}_{ij} \in \mathbb{R}_{>0}$
$\hat{\mathbf{R}}_{ij}$	predicted rating from targeted user j to predicting item i, $\hat{\mathbf{R}}_{ij} \in \mathbb{R}_{>0}$
$\boldsymbol{u}_j$	vector of ratings for user j, $\boldsymbol{u}_j = \mathbf{R}_{*j}$, $\forall * \in [1..n]$
$\|\boldsymbol{u}_j\|$	total number of items that user j has rated, $\|\boldsymbol{u}_j\| \in [0..n]$
$T(\boldsymbol{u}_j)$	time difference of the first and last item user j has rated
$\boldsymbol{i}_i$	vector of ratings for item i, $\boldsymbol{i}_i = \mathbf{R}_{i*}$, $\forall * \in [1..m]$
$\|\boldsymbol{i}_i\|$	total number of users who has rated item i, $\|\boldsymbol{i}_i\| \in [0..m]$
$\bar{\boldsymbol{u}}_j$	$\bar{\boldsymbol{u}}_j = \frac{\sum_{x=1}^{n} \mathbf{R}_{xj}}{\|\boldsymbol{u}_j\|}$, average rating of user j
$\bar{\boldsymbol{i}}_i$	$\bar{\boldsymbol{i}}_i = \frac{\sum_{x=1}^{m} \mathbf{R}_{ix}}{\|\boldsymbol{i}_i\|}$, average rating of item i
$C_{ij} = \boldsymbol{i}_i \cap \boldsymbol{i}_j$	set of co-rated users who rated both *item* i and *item* j
$\|C_{ij}\| = \|\boldsymbol{i}_i \cap \boldsymbol{i}_j\|$	number of co-rated users who rated both *item* i and *item* j, $\boldsymbol{i}_i \cap \boldsymbol{i}_j \in [0..m]$
$S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)$	rating-based similarity between item i and item c, measured based on ratings from users C_{ic} on item i and item c, $S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c) = I_i \odot I_c \mapsto \mathbb{R}$, the operation $\odot$ can be any rating-based similarity measurement such as Cosine Similarity or Pearson Correlation Coefficient.
$S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$	structural similarity between item i and item c, measured based on "who rates what", $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c) = I_i \odot I_c \mapsto \mathbb{R}_{\geq 0}$, the operation $\odot$ can be any structural similarity measurement such as Jaccard, common neighbor, Sorensen or Ochiai.

$$\mathbf{R} = \begin{array}{c c} \begin{array}{ccccc} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 & \dots & \boldsymbol{u}_m \end{array} & \\ \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \mathbf{R}_{13} & \dots & \mathbf{R}_{1m} \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \mathbf{R}_{23} & \dots & \mathbf{R}_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{R}_{n1} & \mathbf{R}_{n2} & \mathbf{R}_{n3} & \dots & \mathbf{R}_{nm} \end{bmatrix} & \begin{array}{c} \boldsymbol{i}_1 \\ \boldsymbol{i}_2 \\ \vdots \\ \boldsymbol{i}_n \end{array} \end{array} \tag{2.1}$$

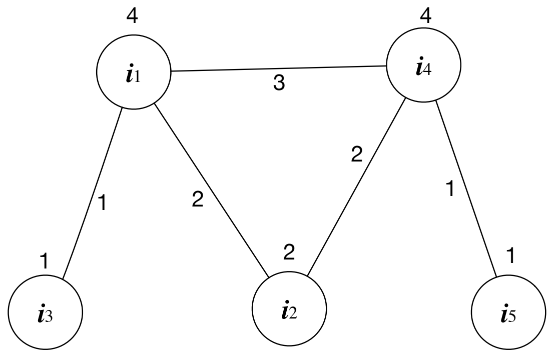
Given a dataset containing ratings from m users on n items, we can get an item-user matrix $\mathbf{R}$ as shown in (3.1). For example, a dataset is given as:

$$\mathbf{R} = \begin{array}{c} \begin{array}{ccccc} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 & \boldsymbol{u}_4 & \boldsymbol{u}_5 \end{array} \\ \begin{bmatrix} 4 & 2 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 \\ 5 & 3 & 5 & 0 & 4 \\ 0 & 0 & 0 & 0 & 5 \end{bmatrix} \begin{array}{c} \boldsymbol{i}_1 \\ \boldsymbol{i}_2 \\ \boldsymbol{i}_3 \\ \boldsymbol{i}_4 \\ \boldsymbol{i}_5 \end{array} \end{array} \qquad (2.2)$$

Then we can draw the item-user network as in Figure 3.1a and draw item-based network as in Figure 3.1b. Then we can make a prediction of the rating from a user on an item by aggregating the opinions from the user on the neighbor items which have been co-rated by other users.



(a) Item-user network: Items are nodes $\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3, ..., \boldsymbol{i}_m$, users are nodes $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, ..., \boldsymbol{u}_n$, edges between an item node $\boldsymbol{i}_i$ and an user node $\boldsymbol{u}_j$ means user j has rated item i with rating of $\mathbf{R}_{ij}$, where $\mathbf{R}_{ij} \in \mathbb{R}_{>0}$, C_{ij} is the group of users who have rated both item i and item j.



(b) Item-based network: Nodes are items, connected by edges if there are users who rated both items. The number on top of each node $|\boldsymbol{i}_i|$ is the number of users who rated item i, the number on each edge is the number of co-rated users $|C_{ij}|$ who rated both item i and item j. Neighbor items are the items that are co-rated by users, for example: items $\boldsymbol{i}_1$ and $\boldsymbol{i}_1$ are neighbors, items $\boldsymbol{i}_1$ and $\boldsymbol{i}_5$ are not.

Figure 2.1.: Illustration of item-user network and Item-based network: From item-user network as in Figure 2.1a, we draw the item-based network as in Figure 2.1b

2.2.2 Memory-based methods

Here we introduce similarity-based algorithms which are often known as memory-based collaborative filtering. In the following section, we describe the basic algorithms as well as similarity measurement computation which plays a critical rule in memory-based CFs.

Similarity Measurements

As mentioned in the introduction, memory-based methods can be implemented in both user-based or item-based approach. So similarities can be either between items or users. Without loss of generality, we illustrate the process using item-based approach since the bottleneck of user-based approach is to search for neighbors among a large user population. With the number of total users m, the total number of items n, and the number of items $|\boldsymbol{u}_j|$ one particular user has rated, (1) to calculate pairwise similarities offline, the computational complexity for user-based will be $O(m^2n)$ as worst case where the item-user rating matrix $\mathbf{R}$ is full. The computational complexity for item-based will be $O(n^2m)$ as worst case. Furthermore, in practice, the item-based approach is closer to $O(nm)$, as most customers have very few purchases [3]. (2) Without pairwise similarities calculated offline, item-based approaches only search for nearest neighbors of the targeted item among what this targeted user has rated previously [58]. So, the search space of similar items of target item for this particular user is limited by how many items this targeted user has rated. The worst case runtime will be $O(|\boldsymbol{u}_j|)$. User-based approach searches for the nearest neighbors of the targeted users among all other users who have rated any of the items this targeted user has rated previously. The worst case runtime of searching for similar users of the targeted user j will be $O(\sum_{i\in \boldsymbol{u}_j} |\boldsymbol{i}_i|)$. So the search space for user-based approach is much larger than item-based approach since most users don't rate only 1 item and most items are not rated only by this 1 user. Thus, item-based approach can be less computational expensive. Another advantage of item-based approach is that similarity between items tends to be more static than similarity between users, allowing its values and neighbors to be pre-calculated, which could shorten the time needed to make a prediction [8].

Similarity measurements are chosen based on what type and quality of information is available. When rating matrix $\mathbf{R}_{ij}$ is available, similarities are defined based on ratings, and two users are considered similar when they give similar ratings to items

[8]. When rating $\mathbf{R}_{ij}$ is not available, similarities can be inferred from the structural information $B(I_i, I_c)$. For example, two users can be considered similar when they rated/reviewed/visited/purchased many items in common if we use the number of items rated by both users as the structural similarity measurement. Furthermore, external information such as user profile(age, occupation, etc.) or content information about items(category, price, etc.) can be incorporated into the similarity calculation [8].

To calculate similarity between users or items, we generally project the user-item bipartite network (Fig. 2.1a) which contains the complete information about the system into a monopartite user-user or item-item network (Fig. 2.1b) illustrated in Fig. 2.1 .

1. **Rating-Based Similarity Measurement**. In many e-commerce services, users are able to evaluate items by ratings. For example, on Amazon.com, products are rated from "1 star" to "5 star", on Pandora Radio, musics are rated from "dislike" or "like". With these explicit rating information, items are considered to be similar when users like both items or dislike both items. Similarity between two items can be measured by Cosine index [8, 59], which is defined as:

$$S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{\sum_{x\in C_{ic}} \mathbf{R}_{ix} \cdot \mathbf{R}_{cx}}{\sqrt{\sum_{x\in Cic} \mathbf{R}_{ix}^2}\sqrt{\sum_{x\in C_{ic}} \mathbf{R}_{cx}^2}}. \tag{2.3}$$

   The similarity can also be measured by Pearson Correlation Coefficient [36], which is defined as:

$$S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{\sum_{x\in C_{ic}} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{i}_i}) \cdot (\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{i}_c})}{\sqrt{\sum_{x\in Cic} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{i}_i})^2}\sqrt{\sum_{x\in C_{ic}} (\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{i}_c})^2}}. \tag{2.4}$$

The similarity can also be measured by Adjusted Cosine [47], which is defined as:

$$S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{\sum_{x \in C_{ic}} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}) \cdot (\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{u}_x})}{\sqrt{\sum_{x \in Cic} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x})^2} \sqrt{\sum_{x \in C_{ic}} (\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{u}_x})^2}}. \tag{2.5}$$

Experiments have shown that among those 3 rating based similarity measurement, adjusted cosine tends to give us the best prediction accuracy [47, 60].

Rating-based similarity measurements between two items are calculated based on co-rated users who have rated both items. Measurements can be based on only a few users. Thus, the similarity calculated may not be reliable to represent the similarity between 2 items. A shrunk correlation coefficient $simShrunk(\boldsymbol{i}_i, \boldsymbol{i}_j)$ was introduced to penalize similarity measured by few users [50]:

$$simShrunk(\boldsymbol{i}_i, \boldsymbol{i}_j) = \frac{|C_{ij}|}{|C_{ij}| + \lambda} S(\boldsymbol{i}_i, \boldsymbol{i}_j). \tag{2.6}$$

Where a typical λ is suggested to be 100 in [50].

2. **Structural-based Similarity Measurement**. Similarity can be quantified solely based on the network structure of the data when explicit rating information is not available. Some research shows that structural-based similarity can produce better recommendations than Pearson correlation coefficient, especially when the input data is sparse [61]. structural-based similarity measurement can be categorized into local vs. global, node-dependent vs. path-dependent, parameter-free vs. parameter-dependent, etc [8]. Here we review some of them.

   (a) **Local structure based similarity**

      i. Common Neighbor. The simplest measurement of this kind is common neighbors, where the similarity between 2 items is directly given by the number of users who rated/clicked/purchased both items. Common Neighbor is computed as:

$$S_{struct}^{CN}(\boldsymbol{i}_i, \boldsymbol{i}_c) = |C_{ic}| \tag{2.7}$$

ii. Salton/Ochiai [62] [63]. Salton is also called Salton Cosine index. It is used to calculate the similarity based on cosine angle between vectors of adjacency matrix [64]. Salton Index is computed as:

$$S_{struct}^{Salton}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{|C_{ic}|}{\sqrt{|\boldsymbol{i}_i| \cdot |\boldsymbol{i}_c|}} \tag{2.8}$$

iii. Jaccard [65]. Jaccard is proposed in 1901 as a statistic to compare similarity and diversity of sample sets. It is the ratio of common neighbors to all neighbors of two nodes. As a result, Jaccard prevents higher degree nodes to have high similarity with other nodes [66]. Jaccard is computed as:

$$S_{struct}^{Jaccard}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{|C_{ic}|}{|\boldsymbol{i}_i \cup \boldsymbol{i}_c|} \tag{2.9}$$

iv. Sørensen [67]. A similar measure to Jaccard proposed by Sørensen in 1948 to measure similarities among species. It is calculated as ratio of twice the common neighbors to all neighbors of two nodes. It is computed as:

$$S_{struct}^{Sørensen}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{2|C_{ic}|}{(|\boldsymbol{i}_i| + |\boldsymbol{i}_c|)} \tag{2.10}$$

v. Hub Promoted Index (HPI) [68]. It is defined as the ratio of common neighbors to the minimum of degrees of two nodes. It is computed as:

$$S_{struct}^{HPI}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{|C_{ic}|}{min(|\boldsymbol{i}_i|, |\boldsymbol{i}_c|)} \tag{2.11}$$

vi. Hub Depressed Index (HDI) [69]. It is defined as the ratio of common neighbors to the maxinum of degrees of two nodes. It gives lower score compared to HPI as HPI is taking the minimum degree. HDI is computed as:

$$S_{struct}^{HDI}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{|C_{ic}|}{max(|\boldsymbol{i}_i|, |\boldsymbol{i}_c|)} \tag{2.12}$$

vii. Leicht-Holme-Newman Index-1 (LHN1) [70]. It is the ratio of common neighbors to the product of degrees of two nodes. It is computed as:

$$S_{struct}^{LHN1}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{|C_{ic}|}{(|\boldsymbol{i}_i| \cdot |\boldsymbol{i}_c|)} \tag{2.13}$$

Compared to Salton in Equation 2.8, Salton always assigns a higher score as it takes the square root of the product as the denominator.

viii. Preferential Attachment Index (PA) [71]. This is calculated independent of the neighborhood of each node. Social networks expand when new nodes joins in with existing nodes with higher degree [71]. PA is computed as:

$$S_{struct}^{PA}(\boldsymbol{i}_i, \boldsymbol{i}_c) = |\boldsymbol{i}_i| \cdot |\boldsymbol{i}_c| \tag{2.14}$$

ix. Adamic-Adar Index [72]. This is calculated by applying an inverse of log scale to the degree of common neighbors of the 2 nodes. It is computed as:

$$S_{struct}^{AA}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \sum_{x \in C_{ic}} \frac{1}{\log k_x} \tag{2.15}$$

where x is a common neighbor of item i and c, k_x is the degree of node x.

x. Resource Allocation Index [69]. This is defined as the amount of resource one node receives from the other node through indirect links, and each intermediate link contributes an unit of resource. It is computed as:

$$S^{RA}_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \sum_{x \in C_{ic}} \frac{1}{k_x} \tag{2.16}$$

The only difference between RA and AA is that AA takes the log of the denominator. As a result, AA always gives a higher score compared to RA.

(b) **Global structure based similarity**

i. Katz Index. The basic assumption is that two items or users are similar if they are connected by many paths. Since the number of distinct paths between pairs of nodes is equal to the elements of an n-th power of the adjacency matrix, $\mathbf{R}^n$, Katz similarity includes paths of all lengths, which is defined as:

$$S^{Katz}_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \sum_{l=1}^{\infty} \beta^l |paths^l_{\boldsymbol{i}_i,\boldsymbol{i}_c}| = \beta \mathbf{R}_{(\boldsymbol{i}_i,\boldsymbol{i}_c)} + \beta^2 (\mathbf{R}^2)_{(\boldsymbol{i}_i,\boldsymbol{i}_c)} + \beta^3 (\mathbf{R}^3)_{(\boldsymbol{i}_i,\boldsymbol{i}_c)} + ... \tag{2.17}$$

where β is a damping factor controlling the path weights, $|paths^l_{\boldsymbol{i}_i,\boldsymbol{i}_c}|$ is the set of all paths with length 1 connecting nodes $\boldsymbol{i}_i$ and $\boldsymbol{i}_c$. When β is small, Katz index will be similar to Common Neighbor method. This can also be written as $S^{Katz}_{struct} = (I - \beta \mathbf{R})^{-1} - I$.

ii. Leicht-Holme-Newman Index (LHN2) [70]. A variant of the Katz index, where the term $paths^l_{\boldsymbol{i}_i,\boldsymbol{i}_c}$ is replaced with $paths^l_{\boldsymbol{i}_i,\boldsymbol{i}_c}/E[paths^l_{\boldsymbol{i}_i,\boldsymbol{i}_c}]$.

iii. Average commute time (ACT) [73]. The average commute time between node $\boldsymbol{i}_i$ and node $\boldsymbol{i}_c$ is defined as the average of number of steps required for a random walker to reach node $\boldsymbol{i}_c$ from node $\boldsymbol{i}_i$ plus that from node $\boldsymbol{i}_c$ to $\boldsymbol{i}_i$. It can be obtained in terms of the pseudoinverse of the network's Laplacian matrix, L^+, as:

$$S^{ACT}_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{1}{L^+_{\boldsymbol{i}_i,\boldsymbol{i}_i} + L^+_{\boldsymbol{i}_c,\boldsymbol{i}_c} + L^+_{\boldsymbol{i}_i,\boldsymbol{i}_c}} \tag{2.18}$$

iv. Random walk with restart (RWR) [74]. This is a direct application of PageRank algorithm [75]. Consider a random walker starting from node x recursively moves to a random neighbor with probability c and returns to node x with probability 1-c. Denoting by q_{xy} the resulting stationary probability that the walker is located at node y, we can write:

$$q_x = c\mathbf{P}^T q_x + (1-c)e_x \tag{2.19}$$

where $\mathbf{P}$ is the transition matrix with elements $P_{xy} = 1/k_x$ if x and y are connected and $P_{xy} = 0$ otherwise. The solution to the equation is:

$$q_x = (1-c)(I - c\mathbf{P}^T)^{-1}\vec{e_x} \tag{2.20}$$

Finally, the similarity is defined as:

$$S^{RWR}_{struct} = q_{xy} + q_{yx} \tag{2.21}$$

v. SimRank [76]. This index is based on the assumption that two nodes are similar if they are connected to similar nodes. This allow us to compute SimRank in a recursive way:

$$S^{SimRank}_{xy} = C\frac{\sum_{z\in\Gamma_x}\sum_{z'\in\Gamma_y} S^{SimRank}_{zz'}}{k_x k_y} \tag{2.22}$$

where $S_{xx} = 1$ indicates each node is similar to itself and $C \in [0,1]$ is the decay factor. SimRank can also be interpreted as how fast two random walkers, who respectively start at node x and node y, are expected to meet at a certain node.

vi. Escape probability [77]. It is computed as:

$$S^{EP}_{xy} = \frac{Q_{xy}}{Q_{xx}Q_{yy} - Q_{xy}Q_{yx}} \tag{2.23}$$

where $Q = \frac{RPR}{1-\beta_{RPR}}$, RPR is the rooted page rank, which is similar to random walk with restart.

(c) **Quasi-local structure based similarity**

i. Local path index (LPI) [69]. This index is similar to local structure based methods but it considers paths of length ¿2. It is computed as:

$$S_{xy}^{LPI} = \mathbf{R}^2 + \epsilon\mathbf{R}^3 + \epsilon^2\mathbf{R}^4 + ... + \epsilon^{n-2}\mathbf{R}^n \tag{2.24}$$

where ϵ is a damping parameter. If $\epsilon = 0$, this index is equal to common neighbors since Eq. 2.24 reduces to $S_{xy}^{LPI} = \mathbf{R}^2$ if $\epsilon = 0$.

ii. Extended Jaccard index [78]. This index considers paths of length 2. It is computed as:

$$S_{xy}^{EJI} = \frac{|\Gamma_d(x) \cap \Gamma_d(y)|}{|\Gamma_d(x) \cup \Gamma_d(y)|} \tag{2.25}$$

where $\Gamma_d(x)$ and $\Gamma_d(y)$ denote the set of extended neighbors of node x and y respectively at hops 1...d for each of the node.

iii. Local random walk [79]. To measure similarity between node x and y, a random walker is introduced in node x and thus the initial occupancy vector is $\pi_x(0) = e_x$. This vector evolves as $\pi_x(t+1) = P^T\pi_x(t)$ for $t \geq 0$. The LRW index at time step t is defined as

$$S_{xy}^{LRW} = q_x\pi_{xy}(t) + q_y\pi_{xy}(t) \tag{2.26}$$

where q is the initial configuration function and t denotes the time step. It was suggested to use a simple approach where q is determined by node degree [79].

iv. Superposed random walk [79]. This is similar to local random walk, but random walker in this approach is continuously released at the starting point. As a result, this index gives us higher similarity compared to local random walk. It is computed as:

$$S_{xy}^{SRW}(t) = \sum_{\tau=1}^{t} S_{xy}^{LRW}(\tau) = \sum_{\tau=1}^{t} [q_x \pi_{xy}(t) + q_y \pi_{xy}(t)] \tag{2.27}$$

where S_{LRW} is defined in Eq. 2.26.

v. FriendLink [80]. This measurement uses paths of length ≥ 2. It is computed as:

$$S_{xy}^{FL} = \sum_{i=2}^{l} \frac{1}{i-1} \cdot \frac{|paths_{xy}^{i}|}{\prod_{j=2}^{i}(n-j)} \tag{2.28}$$

where n is the number of vertices in graph, l is the path length considered $l \geq 2$, $\frac{1}{i-1}$ is the attenuation factor that weights path according to length l. $\prod_{j=2}^{i}(n-j)$ is the number of possible length l-paths from x to y.

K-Nearest Neighbors

K-nearest neighbor (KNN) method is a type of instance-based learning, where the function is only approximated locally. There are 2 steps: similarity calculation and preference prediction. Without loss of generality, taking item-based approach, we first calculate similarities between items using methods listed in the previous section. These pairwise similarities can be calculated offline or can be calculated in real time if similarity measurement chosen is approximated locally, which doesn't require much time. Second, the top K most similar items are chosen to form a nearest neighbor list for the targeted item $\boldsymbol{i}_i$. Item-based memory-based CFs calculate the weighted aggregate of similarities from the K nearest neighbors on the rating from user u on those neighbor items to make the prediction of user u on item i:

1. Simple Weighted Average

The simplest way to predict the rating from user u on item i using simple weighted average is [47]:

$$\hat{\mathbf{R}}_{iu} = \frac{\sum_{c=1}^{K}(\mathbf{R}_{cu}) \cdot S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}{\sum_{c=1}^{K} S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}. \tag{2.29}$$

2. Weighted Sum of Others' Ratings

We can also use the weighted sum of others' ratings:

$$\hat{\mathbf{R}}_{iu} = \bar{\boldsymbol{i}}_i + \frac{\sum_{c=1}^{K}(\mathbf{R}_{cu} - \bar{\boldsymbol{i}}_c) \cdot S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}{\sum_{c=1}^{K} S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}. \tag{2.30}$$

2.2.3 Model-based methods

In this approach, models are developed using different machine learning, data mining techniques to predict users' rating of unrated items. Through this approach, dimensionality reduction techniques are mostly used. High dimensional sparse matrix containing abundant number of missing values are reduced into a much smaller matrix in lower dimensional space. Information to describe the underlying causes of co-occurrence data is preserved by hidden variables, or so called latent variables while the computation complexity and memory requirement for making recommendations is dramatically decreased [8]. There are many model-based CFs such as singular value decomposition (SVD) [81, 82],Bayesian clustering [83], probabilistic latent semantic analysis (pLSA) [39] and latent Dirichlet allocation (LDA) [40], etc. Instead of using latent variables to describe user preferences, items and users can also be assigned to classes or groups that share similar features or preferences respectively [84–86]. In this approach, relationship between individual items and users are explained or represented by that between classes or groups. Through clustering or classification methods, the number of classes or clusters are usually significantly smaller than the number of users or items, which then also result in dimensionality reduction. In the following section, we discuss one of those techniques - singular value decomposition (SVD).

Singular value decomposition (SVD)

We start with item-user matrix $\mathbf{R}$ which contains ratings from users to items. $\mathbf{R}_{ij}$ denotes the rating of item i from user j. If the rating is not given, then $\mathbf{R}_{ij} = 0$. If numerical rating is not known, then $\mathbf{R}_{ij}$ becomes the adjacency matrix as $\mathbf{R}_{ij}$=1 for connected pair between item i and user j. Recommendation process aims to find which zero entries in $\mathbf{R}$ that are likely to be non-zero in the future.

Singular value decomposition (SVD), which belongs to a broader class of latent semantic analysis (LSA) techniques, aims to use latent variables to preserve information. Dimensional reduction is achieved by introducing K hidden variables to preserve the relationships between item attributes and user preferences. The original n-by-m sparse matrix $\mathbf{R}$ is factorized into the product of two matrices $\mathbf{U}$ and $\mathbf{V}$ with dimension n-by-k and k-by-m respectively:

$$\mathbf{R} = \mathbf{U}\mathbf{V} \tag{2.31}$$

where $\mathbf{U}$ and $\mathbf{V}$ contains item attributes and user preferences respectively, in terms of K hidden variables. Items are selected based on the overlap between user preferences and item attributes by the product of U and V.

To obtain $\mathbf{U}$ and $\mathbf{V}$, singular value decomposition (SVD) is a common algebraic tool in LSA to reduce dimensionality while preserving relevant information. $\mathbf{R}$ is factorized as:

$$\mathbf{R} = \mathbf{U}\Sigma\mathbf{V} \tag{2.32}$$

where Σ is a k-by-k diagonal matrix with $k = min(n, m)$. The matrix Σ contains the so-called singular values of $\mathbf{R}$, which is the square root of the eigenvalues of $\mathbf{R}\mathbf{R}^*$. To reduce the dimension, we choose $k < min(n, m)$, which include only K largest singular values in Σ and replace others by zero. This is the K-rank approximation in SVD. $\mathbf{R}$ is now approximated with $\tilde{\mathbf{R}}$:

$$\tilde{\mathbf{R}} \approx \mathbf{R} = \mathbf{U}\tilde{\Sigma}\mathbf{V} \tag{2.33}$$

where $\tilde{\Sigma}$ is the k-rank approximation of Σ. More details such as quantities include the lower and the upper bound of the prediction errors, smallest number of non-zero entries in R to achieve prediction have been studied and proved analytically in [87–90].An estimate of K is also given in [91].

Once $\tilde{\mathbf{R}}$ is approximated, the rating from a user j to an item i can be calculated as the dot product between the item's attribute vector $\mathbf{U}$ and user's preference vector $\mathbf{V}$:

$$\tilde{\mathbf{R}}_{ij} = \sum_{x=0}^{k} \mathbf{U}_{ix} \times \mathbf{V}_{xj} \tag{2.34}$$

2.3 Time-aware recommendation systems

Nowadays, huge amount of information and data are generated and collected every second. Due to Internet's convenience and timeliness, more and more people read news online instead of from traditional media like newspapers [8]. However, given this enormous amount of news and information, an urgent problem emerges: how to filter out irrelevant information and receive timely news and information?

Moreover, a product or service is not judged only by its own characteristics but also by the characteristics of other products or services offered concurrently [11]. It can also be judged by anchoring based on users memory [12–14]. Rating or satisfaction is viewed as a function of the discrepancy or contrast between obtained and expected outcomes [15, 16]. This is documented as contrast effects [17]. Thus, rating scales for users may shift over time, which can bias the score of ratings.

Last but not least, It is well known that consumers' preferences can shift over time [18–25]. This can be due to (1) *exploration* of new items, instead of repeatedly interacting with the same items [22]. New items may continuously enter the market as well. (2) *experience or tendency* to interact with items they have previously had positive interactions with, and to stop interacting with items they have previously negative interactions with [22]. (3) *popularity* of the items, irrespective of personal interaction history [22]. (4) *social influence* may cause preference changes as a result of observing the preference changes of friends [22, 25].

To overcome those problems, collaborative filtering, as the most widely adopted method in recommendation systems, is the first one to be considered to adapt consumer preference dynamics. Most related work focuses on assigning weights with respect to time to suppress old evaluations or items [26, 29, 30, 92–94]. Chen assigned user credits based on non-linear forgotten function during similarity measurement process to select neighbor users that are more active [92]. Vaz found that CF benefits from a rating scale with smaller granularity and applied exponential decay on ratings [93]. Ding studied different similarity measurements and assign different weights on ratings based on time [29]. Koren pointed that time-window and decay cannot work as those techniques lose too much signal. He tracked multiple concept shifts simultaneously by modifying his SVD++ model, which won the Netflix Grand Prize [31]. Koenigstein modeled user temporal dynamics by sessions and item biases by a function and found out item temporal dynamics are much smoother compared to user temporal dynamics [19]. Jiang borrowed the idea of fluid dynamics and accommodate time by changing iteration step sizes [95]. Rafailidis modeled user preference dynamics via tensor-matrix factorization [22]. Lerman studied how a user's rank changes in time as the user becomes more influential in community and aggregate those users' opinions to promote news on Digg.com [96]. Jamali modeled social temporal dynamics of social rating networks (epinions.com, flickr.com) [25]. Lathia used time-dependent iterative method and refined neighbors for CF [27]. He studied a broader problems related to time-aware recommendation systems such as recommendation accuracy and diversity in his Ph.D thesis [97]. Koychev applied gradual forgetting function with linear decay on content-based approach [98]. Baltrunas focused on time partitions based on time cycles [99]. Potter considered psychological decision making processes and modeled user biases like dates by conducting normalization of different rating system by different users [100].

Another problem of recommendation system related to time is temporal diversity [101]. Although an item with the highest predicted score is the most possible candidate for a target user, it may occupy the recommendation list over and over again. Recent developed preferences may take a long time to be reflected in the recommendation lists. Therefore, temporal diversity becomes crucial in designing time-based algorithms [8]. Xiang divided user preferences into longer-term and short term, adjusted similarities between items to make recommendations based on both long-term and short-term preferences.

3. IMPROVING NEIGHBOR-BASED COLLABORATIVE FILTERING BY USING A HYBRID SIMILARITY MEASUREMENT

Acknowledgements: This chapter was published in Expert Systems with Applications, Vol. 160, Dawei Wang, Yuehwern Yih, Mario Ventresca, Improving neighbor-based collaborative filtering by using a hybrid similarity measurement, 113651, Copyright Elsevier (2020).

3.1 Abstract

Memory-based collaborative filtering is one of the recommendation system methods used to predict a user's rating or preference by exploring historic ratings, but without incorporating any content information about users or items. It can be either item-based or user-based. Taking item-based Collaborative Filtering (CF) as an example, the way it makes predictions is accomplished in 2 steps: first, it selects based on pair-wise similarities a number of most similar items to the predicting item from those that the user has already rated on. Second, it aggregates the user's opinions on those most similar items to predict a rating on the predicting item. Thus, similarity measurement determines which items are similar, and plays an important role on how accurate the predictions are. Many studies have been conducted on memory-based CFs to improve prediction accuracy, but none of them have achieved better prediction accuracy than state-of-the-art model-based CFs. In this research, we proposed a new approach that combines both structural and rating-based similarity measurement. We found that memory-based CF using combined similarity measurement can achieve better prediction accuracy than model-based CFs in terms of lower MAE and

reduce memory and time by using less neighbors than traditional memory-based CFs on MovieLens and Netflix datasets.

3.2 Introduction

Recommendation systems are a subclass of information filtering systems that aim to predict a user's opinion or preference of a topic or item, thereby providing personalized recommendations to users by exploiting historic data. They are widely used in e-commerces such as Amazon.com [3], online movie streaming companies such as Netflix [4], and social media networks such as Facebook [5]. With a large amount and diversity of products, a recommendation system could also help streaming service providers or online vendors provide users with recommendations that are specific their preference. This could improve user experience in searching for items or services and potentially lead them to make more purchases, watch more movies, or subscribe to more services. For examples, data gathered for three weeks in the summer of 2001 showed that between 20% to 40% of sales on Amazon are due to recommended products that do not belong to the shop's 100,000 most sold products [6], and 60% of movies rented by Netflix are selected based on personalized recommendations[1]. Furthermore, a recommendation system could generate not only more direct revenue, but also additional revenue by introducing shoppers to new categories [7]. Hence, a recommendation system can significantly impact a company's revenue [8]. Note that 1% improvement in average on MAE(Mean Absolute Error) and RMSE(Root Mean Squared Error) may be a small number, but could result in a significant difference in the ranking of the "top-10" most recommended movies for an individual user [9].

Recommendation systems typically generate a list of recommendations to users in one of three ways [10] - (1) collaborative filtering, (2) content-based filtering, or (3) a hybrid of those two approaches. Collaborative filtering (CF) analyzes historical

[1]As presented by Jon Sanders (Recommendation Systems Engineering, Netflix) during the talk "Research Challenges in Recommenders" at the 3rd ACM Conference on Recommender Systems (2009).

data about user behavior to predict what they might like by learning interactions between users and items. Content-based filtering predicts by learning descriptions of items or profiles of users. Collaborative filtering can be memory-based or model-based. Memory-based approaches rely on pairwise similarities between vectors of ratings, while model-based approaches rely on factorizing the entire rating matrix. Both memory-based and model-based CFs can be implemented from an item-based or user-based perspective. A comparison is given in Section 3.2.1.

This research takes the approach of item-based and memory-based collaborative filtering due to its simplicity, efficiency, and ability to produce accurate recommendations [49]. The way it makes predictions is accomplished in two steps: first, it selects K of the most similar items based on a pair-wise similarity measurement to the predicting item, from the items that the particular user has already rated. Second, it combines this user's ratings on those K items to predict a rating on the predicting item (more details are given in Section 3.3). In this work, we introduce a framework to combine similarity measurements between items. We compare the proposed algorithm against state-of-the-art collaborative filtering techniques using MovieLens [58] and Netflix datasets [4]. Our results indicate that the prediction accuracy of the proposed algorithm performed better than state-of-the-art collaborative filtering techniques in terms of a lower MAE, while also requiring less wall time and computer memory.

3.2.1 Recommendation systems

As stated above, there are three typical approaches for recommendation systems: content-based filtering, collaborative filtering or a hybrid of those two approaches [59] [10] [8]:

1. Content-based filtering. These approaches use keywords or phrases to describe the contents of items, build user profiles to indicate the types of contents each user prefers using those keywords(phrases), and then recommend a list of items

that fits each user's preference. Several techniques have been studied such as pLSA(Probabilistic latent semantic analysis) [38] [39], LDA(Latent Dirichlet Allocation) [40], etc. While content-based methods incorporate descriptive information from items by characterizing using keywords, they do not necessarily incorporate interactions between other individuals. Recommendations are made based solely on the content information of objects that the target user has rated in the past [8]. Content-based filterings are widely used in a variety of domains ranging from recommending webpages, news articles, restaurants, etc [41]. For examples: Pandora Radio[2] recommends users with songs that share similar characteristics [42], and Rotten Tomatoes[3] recommends users with movies that share similar cast and storyline. However, Content-based filtering is unable to make a good recommendation that matches a user's preference if the profiles of users or descriptions of items do not contain sufficient information to tell if the user likes or dislikes the item [41].

2. Collaborative filtering (CF). These approaches analyze historical data on user activities, and use it to predict what they might like based on their similarity to other users, or to items that are similar to the ones the user is known to like [44]. A key advantage of this approach is that CFs study only the interactions between individuals without incorporating feature or attribute information of items and users. Thus, they don't require knowledge about the actual context of the data in order to make recommendations. That is, they can make a prediction without "understanding" a movie, a friend, or a music, etc. Thus, this approach can be applied broadly regardless of the contents of the data. However, when users haven't rated a sufficient number of items, these methods may not perform very well (known as the cold start problem [45] [46]). Collaborative Filtering could also suffer from scalability or sparsity issues [47], (details are listed in Section 3.2.2).

[2]https://www.pandora.com
[3]https://www.rottentomatoes.com

Collaborative filtering can be implemented in two ways: user-based or item-based. To make a prediction of how a user u would rate an item i, User-based CF aggregates the opinions about item i from users that are similar to user u. It assumes that if two persons share similar opinions on some items, they are likely to hold similar opinions on other items as well. On the other hand, item-based CF aggregates the opinions about the items that are similar to item i, and have been rated by user u. It assumes that users are likely to hold the same opinions on similar items. For both item-based and user-based CFs, two approaches have been studied:

(a) Memory-based. This approach calculates similarities between items or users and uses them as weights on ratings to represent how much the opinions on items a user has rated can represent a user's opinion on the unrated item. In general, it is effective and easy to implement. A typical example of this approach is K-Nearest-Neighbor (KNN) [48] [47]. First, item-based memory-based CFs find similar items by calculating pairwise similarities between the predicting item and all other items the user has rated. These pairwise similarities are used to rank how representative of the predicting item each other item is. Therefore, the prediction accuracy of collaborative filtering algorithms is highly dependent on how accurate the similarity measurement is. Second, based on pair-wise similarity measurement, K most similar items to this predicting item are selected from the items that this user has already rated on and then it combines the user's opinions of those K items by weighted average or weighted sum to predict a rating for the unrated item from the user. Such approaches are widely used due to their simplicity, explainability and effectiveness [49], and predictions can be made in real time as new rating data is added. However, its prediction accuracy decreases when few items have been rated. Its scalability is also limited for large datasets [8].

(b) Model-based. This approach uses data mining and machine learning algorithms to develop and train predictive models. There are many different algorithms such as Singular-Value Decomposition (SVD) [50], and Principal component analysis (PCA) [51], which use matrix factorization techniques. Dimensionality reduction is usually used through model-based approach to improve the scalability and accuracy. It addresses the sparsity and scalability problems and performs better in prediction accuracy in comparison to memory-based CFs [10]. But, the models usually use iterative methods to approximate the parameters for the models, which take more time to build and train compared to memory-based approaches. Moreover, they could lose useful information as a result of dimensionality reduction [10].

3. Hybrid. This strategy is to combine two or more techniques [52] [53] [54] [55], or with other techniques such as deep learning [56] or clustering [57] to overcome the limitations of their individuals and improve the performance such as prediction accuracy, scalability. However, it increases the computational complexity such as time and resources required [10].

Recent years, deep neural networks yield immense success on computer vision and natural language processing. There are also successful works on applying Graph Convolutional Networks (GCNs) to recommendation systems. The basic idea of GCNs is to iteratively train the model over multiple layers through two steps at each layer: 1) node embedding with convolutional neighborhood aggregation; 2) non-linear transformation of node embeddings parameterized by a neural network [102] A general framework of Neural network-based Collaborative Filtering (NCF) is proposed to learn user-item interaction via a multi-layer perceptron [103]. Neural Graph Collaborative Filtering (NGCF) is proposed embedding propagation layer to leverage high-order connectivities in user-item integration graph of model-based CF [104]. Multi-Component graph convolutional Collaborative Filtering (MCCF) approach is proposed to distinguish the latent purchasing motivations [105]. They have shown improvements over the state-of-the-art methods in prediction accuracy. There are also works on improving the scalability of Graph Convolutional Network (GCN) algorithms by reducing the complexity, such as PinSage, which combines random walks and graph convolutions to incorporate both graph structure and node feature information [106]; Simple Graph Convolution (SGC) [107] and Linear Residual Graph Convolutional Collaborative Filtering (LR-GCCF) [102] removes the non-linearities. Although most GCN based approaches including GCN based recommendation models achieve the best performance with two layers [106, 108, 109]. It still requires training the model with all user-item interaction information iteratively as model-based approaches.

3.2.2 Characteristics and Challenges of Collaborative filtering

We focus our approach on memory-based collaborative filtering since it can be generalized to be used on any relational data without knowing the content of the data. However, there are some key fundamental challenges of collaborative filtering to predict an accurate rating in real time:

Sparsity

In practice, recommendation systems are used with very large datasets such as those from Amazon [3], Facebook [5] or Netflix [4]. There are usually at least tens of thousands of items and millions of users, but most users only review few items with regard to the total number of items. One of the typical challenges that could be introduced by data sparsity is the Cold Start Problem [45] [46]. Since collaborative filtering methods make recommendations based on users' past preferences, new users need to rate a sufficient number of items in order to let the recommendation system learn their preferences and make reliable recommendations. Collaborative filtering methods are generally unable to make accurate recommendations if users have only rated very few items.

Scalability

As the number of users and items can grow extremely large, traditional collaborative filtering methods will suffer scalability problems. In order to react to new user ratings in real time to make an updated recommendation, it would be challenging for model-based approaches since they use the entire dataset to train. However, in practice, most users have only reviewed relatively few items relative to the total number of items [3], and memory-based methods can react to new ratings and make a prediction in real time even for extremely large datasets [47] [3]. While model-based approaches can mitigate scalability problems by using dimensionality reduction techniques such as SVD [110], they suffer from computationally expensive matrix factorization and may lose useful information in the process. Thus, there are tradeoffs between scalability and performance for model-based approaches [10].

Curse of Dimensionality

Collaborative filtering needs to calculate similarities between items or users in order to identify similar items or users. Those pairwise similarities are calculated in high-dimensions since there are many users and items. With a fixed size of training samples, the predictive power reduces as the dimensionality increases, which is known as the Hughes Phenomenon [111]. Two outcomes may result [112]:

1. **Concentration**. Similarities between all users or items become the same, then memory-based CF is unable to find the most similar items or users, thus will not be able to make a reliable prediction.

2. **Hubness**. Some items occur more frequently in other items' nearest neighbor lists. Those items are usually high rated popular items and are not contributing any personal preference information for recommendations since they may be liked by many users. They can behave like noise making memory-based CF not able to make accurate predictions.

Memory-based Collaborative filtering methods that use cosine-like similarity measurements to calculate pairwise similarities suffer hubness and concentration problems caused by high dimensionality of the data. Model-based methods with dimension reduction methods such as SVD cannot solve those problems either [112]. Hubness starts reducing only when intrinsic dimensionality is reached, where further reduction may incur loss of information [112]. The concentration and hubness are inherent properties of high dimensionality, not proprieties like sparsity or skewness of the distribution of ratings [112]. Both phenomena can negatively affect the accuracy of predictions since they impact the representativeness of nearest neighbor lists [112]. While reducing hubness by using mutual proximity as a similarity measurement can increase the performance of prediction, the accuracy cannot rival the state-of-the-art of model-based approaches [113] [114].

3.2.3 Summary of Main Contributions

In this research, we review traditional memory-based collaborative filtering methods and propose a new approach. We discuss problems that traditional similarity measurements try to solve, and problems each hasn't overcome (Section 4.4). We study how hubness appears in nearest neighbor list using rating-based and structural similarity measurement alone. The main contributions of this research are:

1. We propose a similarity measurement framework that combines rating-based similarity measurements and structural similarity measurements to overcome the limitations of using either of those two measurements alone, and the problems of hubness (Section 4.4).

2. We compare two benchmarks for prediction accuracy evaluation: MAE(Mean Absolute Error) and RMSE(Root Mean Squared Error) (Section 3.5.2) and provide experimental results on three most widely referenced datasets: MovieLens100K, MovieLens1M, and Netflix Challenge datasets (Section 4.5).

3. We show: (1) Our method outperforms state-of-the-art collaborative filterings in terms of lower MAE with 1/3 to 1/2 number of neighbors compared to traditional memory-based CFs on MovieLens 100K, 1M and Netflix datasets (Section 3.6.1); (2) Memory-based CF with the proposed similarity measurement uses 1/2 to 1/39 wall time compared to state-of-the-art model-based CFs on MovieLens 1M dataset (Section 3.6.2) and (3) Our method can achieve 3% lower MAE and RMSE compared to traditional memory-based CFs on non-cold start users on MovieLens 100K dataset (Section 3.6.3).

3.3 Background

As briefly mentioned in Section 3.2.1, memory-based collaborative filtering can be item-based or user-based. There are 2 steps for both approaches: similarity calculation and preference prediction. Without loss of generality, taking item-based approach, the detailed steps are in Sections 3.3.2 and 3.3.3. Before that, let's define some terminologies and notations in Section 3.3.1.

3.3.1 Notation

Table 3.1.: Notation

Notation	Meaning
m	total number of users
n	total number of items
$\mathbf{R}$	n-by-m item-user matrix containing ratings from users to items
$\mathbf{R}_{ij}$	rating of item i from user j , $\mathbf{R}_{ij} \in \mathbb{R}_{>0}$
$\hat{\mathbf{R}}_{ij}$	predicted rating from targeted user j to predicting item i, $\hat{\mathbf{R}}_{ij} \in \mathbb{R}_{>0}$
$\boldsymbol{u}_j$	vector of ratings for user j, $\boldsymbol{u}_j = \mathbf{R}_{*j}$, $\forall * \in [1..n]$
$\lvert\boldsymbol{u}_j\rvert$	total number of items that user j has rated, $\lvert\boldsymbol{u}_j\rvert \in [0..n]$
$\boldsymbol{i}_i$	vector of ratings for item i, $\boldsymbol{i}_i = \mathbf{R}_{i*}$, $\forall * \in [1..m]$
$\lvert\boldsymbol{i}_i\rvert$	total number of users who has rated item i, $\lvert\boldsymbol{i}_i\rvert \in [0..m]$
$\bar{\boldsymbol{u}}_j$	$\bar{\boldsymbol{u}}_j = \frac{\sum_{x=1}^{n} \mathbf{R}_{xj}}{\lvert\boldsymbol{u}_j\rvert}$, average rating of user j
$\bar{\boldsymbol{i}}_i$	$\bar{\boldsymbol{i}}_i = \frac{\sum_{x=1}^{m} \mathbf{R}_{ix}}{\lvert\boldsymbol{i}_i\rvert}$, average rating of item i
$C_{ij} = \boldsymbol{i}_i \cap \boldsymbol{i}_j$	set of co-rated users who rated both *item* i and *item* j
$\lvert C_{ij}\rvert = \lvert\boldsymbol{i}_i \cap \boldsymbol{i}_j\rvert$	number of co-rated users who rated both *item* i and *item* j, $\boldsymbol{i}_i \cap \boldsymbol{i}_j \in [0..m]$
$S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)$	rating-based similarity between item i and item c, measured based on ratings from users C_{ic} on item i and item c, $S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c) = I_i \odot I_c \mapsto \mathbb{R}$, the operation $\odot$ can be any rating-based similarity measurement such as Cosine Similarity or Pearson Correlation Coefficient.
$S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$	structural similarity between item i and item c, measured based on ”who rates what”, $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c) = I_i \odot I_c \mapsto \mathbb{R}_{\geq 0}$, the operation $\odot$ can be any structural similarity measurement such as Jaccard, common neighbor, Sorensen or Ochiai.

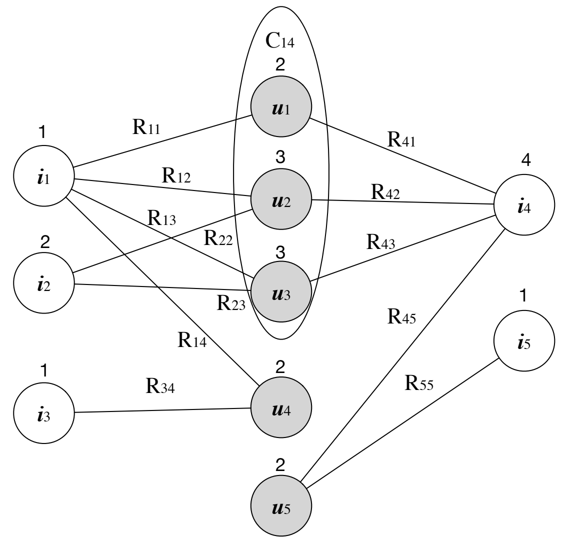
Given a dataset containing ratings from m users on n items, we can get an item-user matrix $\mathbf{R}$ as shown in (3.1).

$$\mathbf{R} = \begin{array}{c c} \begin{array}{ccccc} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 & \dots & \boldsymbol{u}_m \end{array} & \\ \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \mathbf{R}_{13} & \dots & \mathbf{R}_{1m} \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \mathbf{R}_{23} & \dots & \mathbf{R}_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{R}_{n1} & \mathbf{R}_{n2} & \mathbf{R}_{n3} & \dots & \mathbf{R}_{nm} \end{bmatrix} & \begin{array}{c} \boldsymbol{i}_1 \\ \boldsymbol{i}_2 \\ \vdots \\ \boldsymbol{i}_n \end{array} \end{array} \tag{3.1}$$

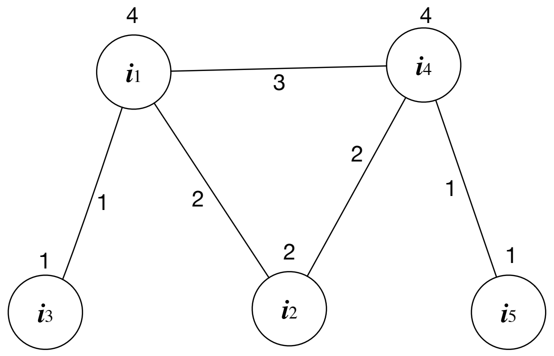
For example, a dataset is given as:

$$\mathbf{R} = \begin{array}{c} \\ \\ \\ \\ \\ \end{array}\begin{array}{ccccc|c} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \boldsymbol{u}_3 & \boldsymbol{u}_4 & \boldsymbol{u}_5 & \\ 4 & 2 & 4 & 1 & 0 & \boldsymbol{i}_1 \\ 0 & 4 & 2 & 0 & 0 & \boldsymbol{i}_2 \\ 0 & 0 & 0 & 3 & 0 & \boldsymbol{i}_3 \\ 5 & 3 & 5 & 0 & 4 & \boldsymbol{i}_4 \\ 0 & 0 & 0 & 0 & 5 & \boldsymbol{i}_5 \end{array} \tag{3.2}$$

Then we can draw the item-user network as in Figure 3.1a and draw item-based network as in Figure 3.1b. Then we can make a prediction of the rating from a user on an item by aggregating the opinions from the user on the neighbor items which have been co-rated by other users.



(a) Item-user network: Items are nodes $\boldsymbol{i}_1, \boldsymbol{i}_2, \boldsymbol{i}_3, ..., \boldsymbol{i}_m$, users are nodes $\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, ..., \boldsymbol{u}_n$, edges between an item node $\boldsymbol{i}_i$ and an user node $\boldsymbol{u}_j$ means user j has rated item i with rating of $\mathbf{R}_{ij}$, where $\mathbf{R}_{ij} \in \mathbb{R}_{>0}$, C_{ij} is the group of users who have rated both item i and item j.



(b) Item-based network: Nodes are items, connected by edges if there are users who rated both items. The number on top of each node $|\boldsymbol{i}_i|$ is the number of users who rated item i, the number on each edge is the number of co-rated users $|C_{ij}|$ who rated both item i and item j. Neighbor items are the items that are co-rated by users, for example: items $\boldsymbol{i}_1$ and $\boldsymbol{i}_1$ are neighbors, items $\boldsymbol{i}_1$ and $\boldsymbol{i}_5$ are not.

Figure 3.1.: Illustration of item-user network and Item-based network: From item-user network as in Figure 3.1a, we draw the item-based network as in Figure 3.1b

Without loss of generality, we illustrate the process using item-based approach since the bottleneck of user-based approach is to search for neighbors among a large user population. With the number of total users m, the total number of items n, and the number of items $|\boldsymbol{u}_j|$ one particular user has rated, (1) to calculate pairwise similarities offline, the computational complexity for user-based will be $O(m^2n)$ as worst case where the item-user rating matrix $\mathbf{R}$ is full. The computational complexity for item-based will be $O(n^2m)$ as worst case. Furthermore, in practice, the item-based approach is closer to $O(nm)$, as most customers have very few purchases [3]. (2) Without pairwise similarities calculated offline, item-based approaches only search for nearest neighbors of the targeted item among what this targeted user has rated previously [58]. So, the search space of similar items of target item for this particular user is limited by how many items this targeted user has rated. The worst case runtime will be $O(|\boldsymbol{u}_j|)$. User-based approach searches for the nearest neighbors of the targeted users among all other users who have rated any of the items this targeted user has rated previously. The worst case runtime of searching for similar users of the targeted user j will be $O(\sum_{i \in \boldsymbol{u}_j} |\boldsymbol{i}_i|)$. So the search space for user-based approach is much larger than item-based approach since most users don't rate only 1 item and most items are not rated only by this 1 user. Thus, item-based approach can be less computational expensive. Another advantage of item-based approach is that similarity between items tends to be more static than similarity between users, allowing its values and neighbors to be pre-calculated, which could shorten the time needed to make a prediction [8].

3.3.2 Item-Based Similarity Computation

Similarity measurements are chosen based on what type and quality of information is available. When rating matrix $\mathbf{R}_{ij}$ is available, similarities are defined based on ratings, and two users are considered similar when they give similar ratings to items [8]. When rating $\mathbf{R}_{ij}$ is not available, similarities can be inferred from the structural information $B(I_i, I_c)$. For example, two users can be considered similar when they rated/reviewed/visited/purchased many items in common if we use the number of items rated by both users as the structural similarity measurement. Furthermore, external information such as user profile(age, occupation, etc.) or content information about items(category, price, etc.) can be incorporated into the similarity calculation [8].

Rating-Based Similarity Measurement

Traditional methods only use ratings to compute similarity between items (e.g. Pearson correlation coefficient [36], cosine and adjusted cosine [47]) by only considering ratings from co-rated users C_{ij}. Taking adjusted cosine similarity measurement as an example, it is calculated as:

$$S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{\sum_{x \in C_{ic}} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x})(\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{u}_x})}{\sqrt{\sum_{x \in Cic} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x})^2} \sqrt{\sum_{x \in C_{ic}} (\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{u}_x})^2}}. \tag{3.3}$$

There are some common cases that using rating-based similarity measurements alone can misrepresent the similarity between items, and thus lead to the inaccuracy of prediction:

- **Opinions agreed upon by a different number of users are weighted the same.** Universally liked items are not as useful at capturing similarity of less common items since they do not contribute any user preference information as most users likes those items [60]. To compensate, the idea of inverse user frequency was introduced by multiplying $log(\frac{|\boldsymbol{i}_i|}{m})$ to the original item similarity measurement [60]. Note that if $|\boldsymbol{i}_i| = m$, then $log(\frac{|\boldsymbol{i}_i|}{m}) = 0$. In this case, less common items are assigned a higher weight. However, it doesn't consider the number of co-rated users. Therefore, opinions aggregated based on a few users will be weighted the same with regard to opinions aggregated based on more co-rated users. The similarity calculated can be biased towards the very few users.

- **Various popularities of items are not considered.** Similarity can be calculated based on opinions from few users who have rated both items as the number of users who rated an item varies. Traditional methods such as cosine, Pearson correlation coefficient or adjusted cosine do not consider the number of co-rated pairs. By neglecting the quantity of co-rated pairs, the similarity between items calculated based on many users' opinions are considered equally as important as one calculated based on only one user's opinion. Therefore, pairwise similarities between items can be biased towards few users when there are few who have rated both items. A shrunk correlation coefficient $simShrunk(\boldsymbol{i}_i, \boldsymbol{i}_j)$ was introduced to penalize similarity measured by few users [50]:

$$simShrunk(\boldsymbol{i}_i, \boldsymbol{i}_j) = \frac{|C_{ij}|}{|C_{ij}| + \lambda} S(\boldsymbol{i}_i, \boldsymbol{i}_j). \tag{3.4}$$

Where a typical λ is suggested to be 100 in [50].
However, (3.4) only considers the number of users who have rated both items $|C_{ij}|$, not the total number of users who have rated each item $|\boldsymbol{i}_i|$ or $|\boldsymbol{i}_j|$. So, similarities between all pairs of items are penalized by a fixed parameter λ, not with regard to how many co-rated users each pair could possibly have. It penalize opinions aggregated based on few users but ignores that the popularity of items varies.

- **Curse of dimensionality.** As stated in Section 3.2.2, with a fixed number of training samples, as a result of the Hughes Phenomenon, predictive power decreases as the dimension of the user-item matrix increases [111]. Two consequences may result when calculating similarity in high-dimensional space: 1. Pairwise distances between all pairs tend to be similar, known as distance concentration; 2. Items with high similarity will frequently occur in other item's nearest neighbor lists, which is known as hubness [112].

- **Take a part for the whole.** By calculating similarity measurement using co-rated users, traditional CFs only consider the ratings from co-rated users. Dissimilar items do not usually tend to share co-rated users, yet traditional CFs aim to find whether they are similar by only focusing on the intersection C_{ij}, but similarity calculated can then be very unreliable or biased towards to C_{ij}.

Structural Similarity Measurement

Instead of just focusing on the co-rated items, structural similarity measurement focuses on vectors of $\boldsymbol{i}_i$ and $\boldsymbol{i}_j$. In this way, pairwise similarity is not biased to only those co-rated users C_{ij}. However, by using only structural information, we can find highly correlated items that users tend to rate together, but we can not know whether users like both items without considering ratings $\mathbf{R}_{ui}$. Therefore, the underlying assumption of collaborative filtering, which is that a person who likes item A is likely to like its similar item B, will not be valid. Moreover, the curse of dimensionality could also become an issue since we are ingoring rating information by only focusing on structural information [114].

Predictions made by traditional item-based memory-based CF with rating-based similarity measurement or structural similarity measurement alone can be inaccurate since they only focus on either the structural part or rating part of the relationship between items, as shown in Section 3.3.2. Studies have tried to combine rating-based similarity measurement with structural information to form a better similarity measurement. As illustrated in Section 3.3.2, [50] used a fixed variable to penalize similarities that are calculated based on a very few users as illustrated in Equation (3.4) without considering the local information such as the upper bound of the number of co-rated users $\min(|\boldsymbol{i}_i|, |\boldsymbol{i}_j|)$. [60] used only the structural information about the number of users who rated the targeted item $|\boldsymbol{i}_i|$ without considering structural information about the other neighbor item $|\boldsymbol{i}_j|$ or the number of co-rated users between them C_{ij}. None of them considered how "strong" the structural relationship is between two items relatively to other neighbors locally.

3.3.3 Preference Prediction

After similarity $S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ is calculated, the top K most similar items are chosen to form a nearest neighbor list for the targeted item $\boldsymbol{i}_i$. Item-based memory-based CFs calculate the weighted aggregate of similarities from the K nearest neighbors on the rating from user u on those neighbor items to make the prediction of user u on item i:

Simple Weighted Average

The simplest way to predict the rating from user u on item i using simple weighted average is [47]:

$$\hat{\mathbf{R}}_{iu} = \frac{\sum_{c=1}^{K}(\mathbf{R}_{cu}) \cdot S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}{\sum_{c=1}^{K} S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}. \tag{3.5}$$

Weighted Sum of Others' Ratings

We can also use the weighted sum of others' ratings:

$$\hat{\mathbf{R}}_{iu} = \bar{\boldsymbol{i}}_i + \frac{\sum_{c=1}^{K}(\mathbf{R}_{cu} - \bar{\boldsymbol{i}}_c) \cdot S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}{\sum_{c=1}^{K} S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)}. \tag{3.6}$$

3.4 Proposed Approach

We propose a new similarity measurement to quantify the similarity between items. Unlike traditional CFs that use rating-based similarity measurement, such as adjusted cosine, Pearson correlation coefficient, or structural similarity measurement alone, we propose to combine both measurements. Rating-based similarity measurements $S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ can tell us whether two items are positively or negatively correlated based on opinions of users C_{ic} who have rated both items, while structural similarity measurements $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ can tell us how correlated two items are based on the number of users who have rated either one of them without telling us whether they are positive or negative correlated since they can only be positive. By combining structural measurement and rating-based measurement together, we aim to know how correlated they are and how similar they are from users who have rated not only both items but also any of the two items. That is, we expand the focus from only the users who have rated both items C_{ic} to any user that has rated any of the two items so that similarities are not biased towards to users C_{ic}.

In Equation 3.7, the structural similarity measurement $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ is used to search for strongly correlated items by focusing on all users who have rated any of those two items and then rating-based similarity measurement $S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ is used to find out whether those two items are positively or negatively correlated based on the options from users C_{ic}. We assume that in practice, the popularity of each item varies, so that the number of users who rate item $|\boldsymbol{i}_i|$ varies, and the number of co-rated users between each pair of items $|C_{ic}|$ varies. Therefore, we combine structural similarity measurement as a weight on a rating-based similarity measurement to compensate the differences of popularity among items. In this way, opinions of items aggregated based on few co-rated users are penalized. To calculate the similarity between item i and its neighbor item c, the similarity measurement $S_{combined}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ becomes:

$$S_{combined}(\boldsymbol{i}_i, \boldsymbol{i}_c) = S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)^{\alpha} \cdot S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c). \tag{3.7}$$

where $\alpha \geq 1$ is the amplification parameter on structural similarity measurement.

3.4.1 Choice of structural and rating-based similarity measurements

The rating-based similarity measurement we choose is Adjusted Cosine as literatures show that it gives us the most accurate prediction result among other rating-based similarity measurements [47]. A list of some structural similarity measurement is in Table 3.2. The structural similarity measurement we choose fulfills the following 2 requirements:

Approximated locally

As KNN is a type of instance-based learning, where the function is only approximated locally. In this way, we do not need additional global information from the data to calculate this new introduced structural-based similarity measurement. We keep the advantage of memory-based CF which doesn't need the entire item-user rating matrix to make a prediction. Moreover, by approximating locally, we do not use a fixed parameter estimated globally across all pairs of items to penalize similarities among items with various popularities. The combined similarity measurement only requires the information from the two item vectors $\boldsymbol{i}_i$, $\boldsymbol{i}_c$ about the number of co-rated users $|C_{ic}|$ and the number of users who rated each of those 2 items $|\boldsymbol{i}_c|$, $|\boldsymbol{i}_i|$. The structural similarity measurement can be any structural similarity measurements approximated locally.

A ratio of intersection to union

Some structural similarity measurements only consider part of the structural information. Common neighbor only considers the number of co-rated users. PA only considers the number of users who rated each item. Those measurements are not suitable to compare across pairs of items as the size of the local network formed for each item varies. A ratio such as Ochiai that utilizes both the intersection and the union of 2 vectors is more suitable to compare across different local networks of items.

Table 3.2.: Lists of some structural similarity measurements approximated locally. Those measurements do not require global information such as the total number of users. All information required for the calculation can be gathered from those 2 associated vectors $\boldsymbol{i}_c$ and $\boldsymbol{i}_i$.

Structural similarity measurement	Definition $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$
Common neighbor	$\|C_{ic}\|$
PA	$\|\boldsymbol{i}_i\| \cdot \|\boldsymbol{i}_c\|$
Jaccard	$\|C_{ic}\|/\|\boldsymbol{i}_i \cup \boldsymbol{i}_c\|$
Salton/Ochiai	$\|C_{ic}\|/\sqrt{\|\boldsymbol{i}_i\| \cdot \|\boldsymbol{i}_c\|}$
Sorensen	$2\|C_{ic}\|/(\|\boldsymbol{i}_i\| + \|\boldsymbol{i}_c\|)$
HPI	$\|C_{ic}\|/min(\|\boldsymbol{i}_i\|, \|\boldsymbol{i}_c\|)$
HDI	$\|C_{ic}\|/max(\|\boldsymbol{i}_i\|, \|\boldsymbol{i}_c\|)$
LHN1	$\|C_{ic}\|/(\|\boldsymbol{i}_i\| \cdot \|\boldsymbol{i}_c\|)$

Specifically, the structural similarity measurement we have chosen is Ochiai similarity where cosine similarity measurement is applied to binary data since it is a ratio of the number of co-rated users $|C_{ic}|$ to the number of users who rated each item $|\boldsymbol{i}_c|$, $|\boldsymbol{i}_i|$ approximated locally. Results of adjusted cosine combined with other structural similarity measurement such as common neighbor, Sorensen, and Ochiai ($\alpha = 1$) are shown in Appendix.

3.4.2 Improvements over using rating or structural similarity alone

As constraint of using rating or structural based similarity measurement alone shown in Section 3.3.2, the proposed method with structural similarity measurements chosen fulfilling requirements in Section 3.4.1 improves over the following aspects:

- **Compensate unpopular but similar items**: if both items are not popular but most users have rated both items, we do not penalize items' similarity because of their unpopularity. Instead of using a fixed global shrinkage factor [50] to penalize pairs of items with small number of co-rated users, the structural similarity measurement uses local ratio of $|C_ij|$ to the local possible co-rated users $|\boldsymbol{i}_i|$ and $|\boldsymbol{i}_j|$ to compensate unpopular but highly co-rated items. We expect to see the prediction accuracy higher than using rating-based similarity measurement alone.

- **Reduce hubness**: When calculating based on rating-based similarity measurement only, big hubs are those universally liked popular items. Those items do not contribute much personal preferences but highly referred as neighbor items. By multiplying rating-based similarity measurement with structural-based similarity measurement, big hubs caused by using rating-based similarity measurement alone are penalized since those big hubs are rated by lots of users, which ends with a relative small structural-based and combined similarity. We expect to see the hubness of using the local ratio structural-based similarity weighted rating-based similarity lower than using rating-based similarity alone.

  When calculating similarities based on structural data only, big hubs appear when some user rates unpopular items that are rarely rated by other users. So the structural-based similarity will be high. But the nearest neighbor list of collaborative filtering contains only items that the user has rated on. So those big hubs could only appear in the nearest neighbor list of users who like rating unpopular items. Using those nearest neighbor lists that contain unpopular items, it would recommend unpopular items to users who like unpopular items. so we don't expect the hubness of overall users would be change much from using structural similarity measurement alone to using the combined measurement.

- **Full picture**: By combining both, we are able predict whether the user likes an item or not from his or her opinions on highly correlated neighbor items. Structural similarity measurement finds the highly correlated items and rating-based similarity measurement tells us whether the user likes the item.

3.4.3 Amplification parameter α

Structural similarity measurement is less biased than rating-based similarity measurement since structural similarity measurements consider opinions from more users $\boldsymbol{i}_i \cup \boldsymbol{i}_j$ while rating-based similarity measurements focus only on the intersection $\boldsymbol{i}_i \cap \boldsymbol{i}_j$. The probability that dissimilar items are co-rated by many users is rarely low. Therefore, the structural similarity measurement is raised to the power of α to enlarge the differences of structural similarities between each nearest neighbor. In this way, rankings can only change when there is a large difference between their rating-based similarities as illustrated in Table A.3 in Appendix. The higher power we raise structural similarity measurement to, we emphasize more on the ranking calculated based on structural similarity.

Amplification parameter also reduces noise in the data [10]. It tends to favor high weights as small values become negligible when being raised to a power.

Amplification parameter α is determined by the data so that the top K nearest neighbors of the targeted item can contribute equal weight on their opinions so that the prediction is not biased towards opinions of the first few top ranked neighbor items. In this way, the similarity measurement is mainly used for finding top K nearest neighbors. Once the top K nearest neighbor list is chosen, by assigning equal weight to each neighbor item. Otherwise, if one of the opinions from the first few top ranked neighbor items does not truly represent user's opinion, the prediction can be inaccurate. A typical value for α is around 2.5 for MovieLens and Netflix datasets estimated by cross validation illustrated in Section 4.5.

3.5 Experimental Setup

3.5.1 Data Sets

For comparison purposes, the MovieLens and Netflix datasets are used as they are the most widely referenced in literature. (See Table 4.2)

Table 3.3.: Data sets. Datasets vary in different sizes (number of users, number of items, number of ratings, and density). MovieLens have been pre-processed-users who had less than 20 ratings or did not have complete demographic information were removed [115]. MovieLens 100K dataset contains data collected through MovieLens web site (movielens.umn.edu) from 1997/09/19 through 1998/04/22. MovieLens 1M dataset contains data collected during from 2000/04/25 through 2003/02/28. Netflix dataset contains data collected from 1998/10 through 2005/12 and reflects all ratings received during this period. Density is the ratio of the number of actual ratings to the possible maximum number of ratings

Data Set	Number of Users	Number of Movies	Number of Ratings	Density
MovieLens 100K [115]	943	1,682	100,000	6%
MovieLens 1M [115]	6,040	3,952	1,000,209	4%
Netflix Prize [4]	480,189	17,770	100,480,507	1%

The distribution of ratings are shown in Figure 3.2. We can see all three datasets have similar shape of distribution. They are all right skewed with peak at rating of 4.

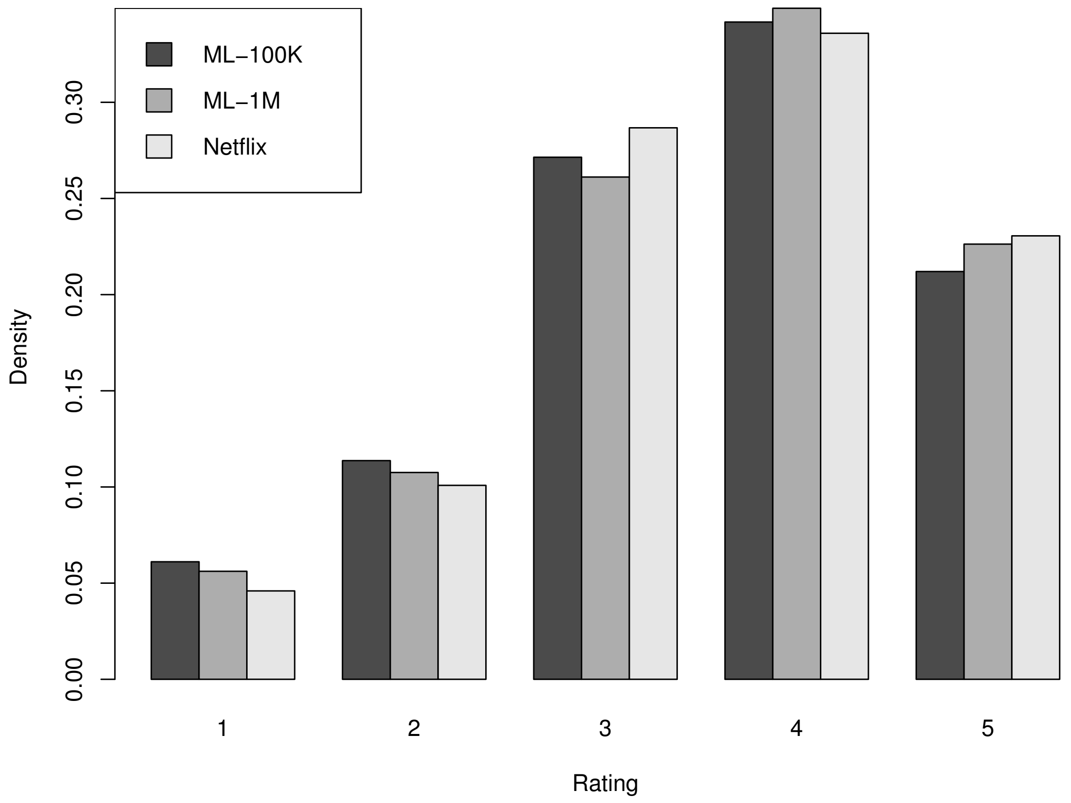
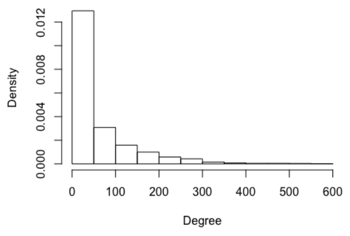
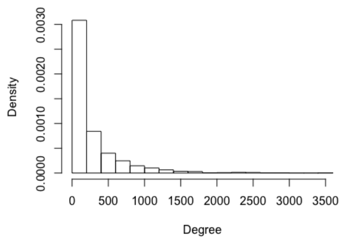


Figure 3.2.: Density Distribution of Ratings of MovieLens, Netflix, where x-axis is the rating from 1 to 5, y-axis is the density of number of users who rated given rating.

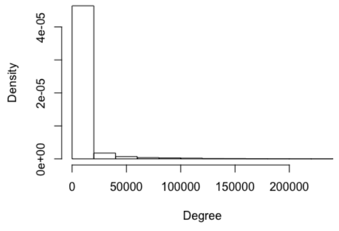
Combined similarity measurement is expected to work better with the degree distribution of items of those datasets shown in Figure 3.3. Although MovieLens datasets are not as skewed as Netflix dataset, all three datasets have a long tail. Some items are rated only by a few users. Using rating-based similarity measurement only, many pairwise similarities are based on few co-rated users. Those similarities will be biased towards those few users. There are popular items rated by lots of users too. Using rating-based similarity measurement alone, the similarities will tend to be the same known as the concentration problem. High rated popular items will also become big hubs that frequently appear in other items' nearest neighbor lists. Those will cause CF not able to make a reliable prediction as pointed in Section 3.2.2.



(a) ML-100K



(b) ML-1M



(c) Netflix

Figure 3.3.: Degree Distribution of Items of MovieLens, Netflix, where x-axis is the degree of the number of users, y-axis is the density of number of items gets rated by given number of users. Rating-based similarity measurement will not be able to capture and utilize those structural information.

3.5.2 Accuracy Evaluation

Both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) were used to evaluate prediction accuracy since most literature on MovieLens dataset use MAE and most literature on Netflix dataset use RMSE. MAE measures the average over the sample of the absolute differences between the actual rating and the prediction, where all individual differences have equal weight:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{R}_{iu} - \hat{\mathbf{R}}_{iu}|. \tag{3.8}$$

where n is the number of total predictions.

RMSE measures the average of squared differences between the actual observation and the prediction:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{R}_{iu} - \hat{\mathbf{R}}_{iu}\right)^2}. \tag{3.9}$$

Both MAE and RMSE are negatively-oriented scores, which means that lower values are more desirable. The difference is that RMSE squares the errors before they are averaged, so RMSE gives a higher weight to larger errors. RMSE does not necessarily increase with the variance of the errors; it increases with the variance of the frequency distribution of error magnitudes. See Appendix for examples.

Another implication is that RMSE can be problematic when compared results calculated based on different sample sizes because of the square root of the number of errors. RMSE tends to be increasingly larger than MAE when the size of the sample size increases. So it would be inappropriate to compare RMSE across datasets with different sample sizes.

Overall, RMSE is ambiguous since it reflects three characteristics of a set of error [116] [117]: (1) variability of the distribution of error magnitudes, (2) the square root of the number of errors, (3) the average-error magnitude (MAE). It could be an inappropriate and misinterpreted measure of average error because sums-of-squared-based statistics do not satisfy the triangle inequality. We present both results below and leave it to the reader to determine which is more appropriate in their context.

3.5.3 Hubness Evaluation

To compute hubness, first, $p_i^K(x)$ is defined as:

$$p_i^K(x) = \begin{cases} 1, & \text{if item } x \text{ is among the } K \text{ nearest neighbors of item } i \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

Second, $O^K(x)$ is defined as the number of times item x occurs in the K-nearest neighbor lists of all other objects:

$$O^K(x) = \sum_{i=1}^{n} p_i^K(x) \tag{3.11}$$

Then, hubness is defined as the skewness of the distribution of O^K [118]:

$$H^K = \frac{E[(O^K - \mu_{O^K})^3]}{\sigma_{O^K}^3}. \tag{3.12}$$

A dataset with few hub items that occurs frequently among other items' K-nearest neighbor lists and many anti-hubs with occurrence of 0 yields high hubness [114]. However, this doesn't capture whether those hubs are overall liked items. For Collaborative Filtering, overall liked items that occurs frequently among other items' K-nearest neighbor lists are the hubs that do not contribute any personal preferences. To capture hubness of overall liked items, $p_i^K(x)$ in Equation (3.10) is redefined as:

$$p_i^K(x) \begin{cases} \bar{\mathbf{R}}_{\boldsymbol{i}_x}, & \text{if item } x \text{ is among the } K \text{ nearest neighbors of item } i \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

Let's call this rating conditioned hubness H_R^K.

3.6 Experimental Results

Experiments over MovieLens dataset (ml-100K and ml-1M) were conducted using a 4 GHz Intel i7 CPU, 16 GB 1600MHz DDR3 memory iMac with OSX 10.13. Experiments over Netflix dataset were conducted on computing clusters of two 12-Core Intel Xeon Gold CPU, 96 GB memory, 793.2 TeraFLOPS. Time measured were in wall time. Memory-based CFs are implemented in C++. Model-based CFs implemented by MyMediaLite library [119] are written in C#. Hyperparameters of model-based CFs suggested by MyMediaLite over the MovieLens dataset are calculated using 5-fold cross validation on the training dataset. Hyperparameter selection is supported using grid search and Nelder-Mead algorithm [120]. The exact partitioning of the 5-fold test is not provided by MyMediaLite. To make a fair comparison among different methods, we generated 5-fold training and testing dataset randomly. The same partitioning of training and testing dataset is used across all experiments in this paper.

The rest of this section is structured as follows: Prediction accuracy in terms of MAE and RMSE over MovieLens and Netflix datasets were compared among state-of-the-art methods in Section 3.6.1. We proposed a different testing method besides using the probe dataset provided in the Netflix prize challenge over the Netflix dataset. We excluded ratings that are in the probe dataset from the training dataset. In Section 3.6.2, we compared time and memory resources required by memory-based and model-based methods. In Section 3.6.3, we compared the prediction accuracy and coverage of memory-based methods by cutting off cold start users.

3.6.1 Prediction Accuracy

MovieLens Datasets

5-fold test is performed on the MovieLens (ml-100K and ml-1M) datasets. MAE performance follows a concave shape for both K (size of the nearest neighbor list) ranging from 10 to 50 with increment of 1 and α(case amplification parameter) ranging from 1 to 5 with increment of 1 over ml-100K dataset as shown in Figure 3.5. MAE is the lowest when K=25. By fixing K=25, it is observed MAE follows a concave shape in Figure 3.5a. α ranging from 2 to 3 gives the lowest MAE value. By letting α=2.6, MAE decreases as K increases from 10 to 25. After K=25, MAE increases as shown in Figure 3.5b. By setting K=25 and α=2.6, memory-based collaborative filtering with the proposed similarity measurement outperforms other state-of-the-art memory-based and model-based methods over ml-1k and ml-1M datasets with respect to MAE as shown in Tables 3.4 and 3.5.

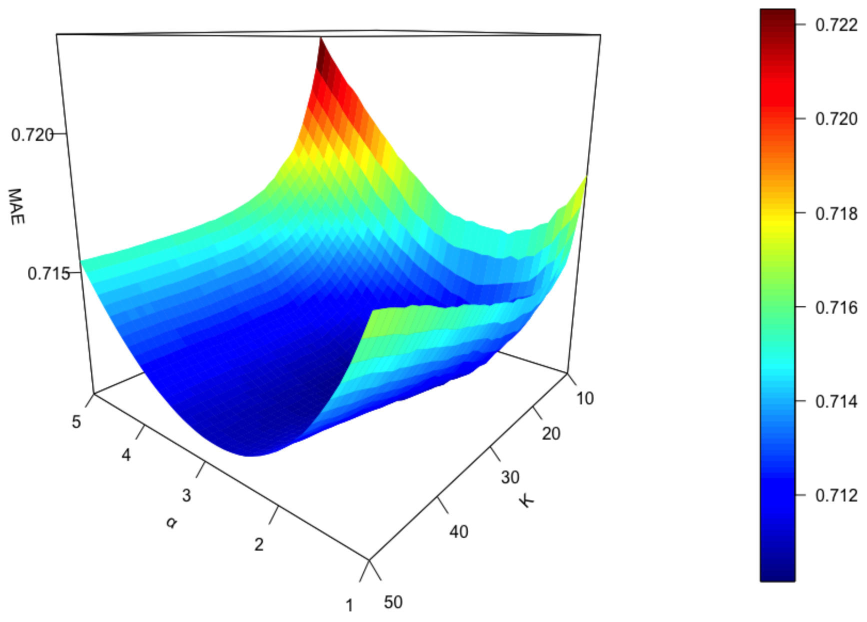
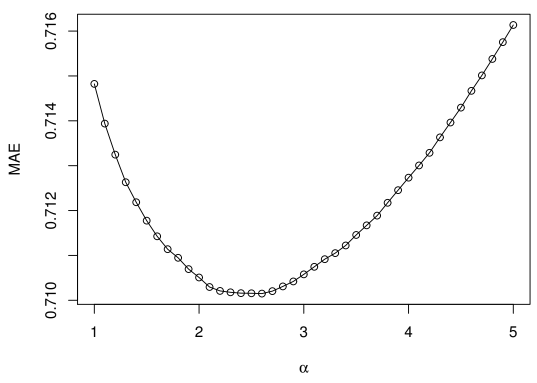
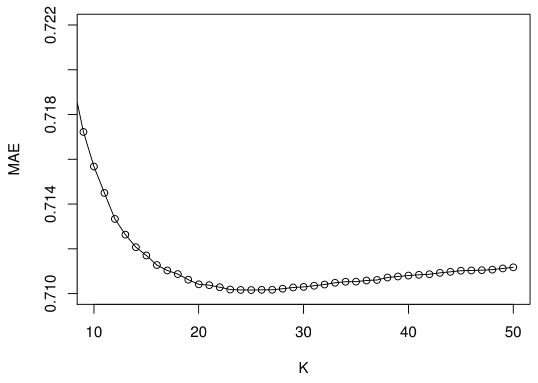


Figure 3.4.: MAE performance for different size of nearest neighbor list and case amplification parameter, where x-axis is the power (case amplification parameter) raised on structural similarity measurement, y-axis is the size of the nearest neighbor list, z-axis is the MAE(Mean Absolute Error). We can see MAE follows a concave shape respect to K and α. We choose the parameters with MAE at the lowest point.



(a) MAE performance for different power when K=25, where we can see that MAE follows a concave shape respect to α.



(b) MAE performance for different K when α=2.6, where we can see that MAE follows a concave shape with respect to K.

Figure 3.5.: Accuracy performance for different nearest neighbor list size and amplification parameter. Those two figures are transections of Figure 3.5.

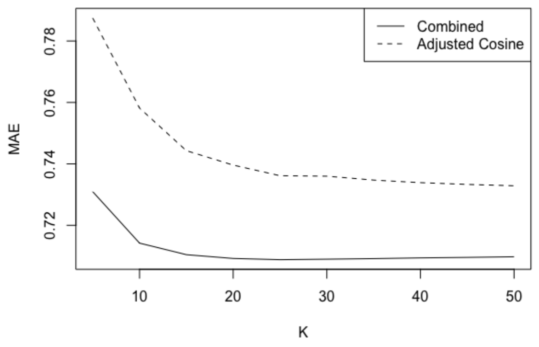
Table 3.4.: Comparison of prediction accuracy among state-of-the-art CFs on ML-100K dataset. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119].

Method	Parameterization	MAE	RMSE
ItemKNNPearson	K=60	0.736125	0.936507
ItemKNNAdjustedCosine	K=60	0.733618	0.935528
ItemKNNPearsonReg	reg_I=1, reg_U=12, K=60	0.733399	0.932648
ItemKNNAdjustedCosineReg	reg_I=1, reg_U=12, K=60	0.729538	0.929991
ItemKNNBCosine	K=25	0.725362	0.924898
BiasedMatrixFactorization	num_factors=40, bias_reg=0.1, reg_u=1.0, reg_i=1.2, learn_rate=0.07, num_iter=100, frequency_regularization=true, bold_driver=true	0.721715	0.912668
ItemKNNPearsonShrink	reg_I=1, reg_U=12, K=40, shrinkage=2500	0.718144	0.915559
ItemKNNAdjustedCosineShrink	reg_I=1, reg_U=12, K=40, shrinkage=2500	0.717390	0.915710
SigmoidUserAsymmetricFactorModel	num_factors=5, regularization=0.003, bias_reg=0.01, learn_rate=0.006, bias_learn_rate=0.7, num_iter=70	0.716550	0.910353
SVDPlusPlus	num_factors=50, regularization=1, bias_reg=0.005, learn_rate=0.01, bias_learn_rate=0.07, num_iter=50, frequency_regularization=true	0.715941	0.909214
ItemKNNCombined	K=25, power=2.6	0.708815	0.909131
ItemKNNCombinedReg	RegU=1, RegI=4, K=25, power=2.6	0.704588	0.903682

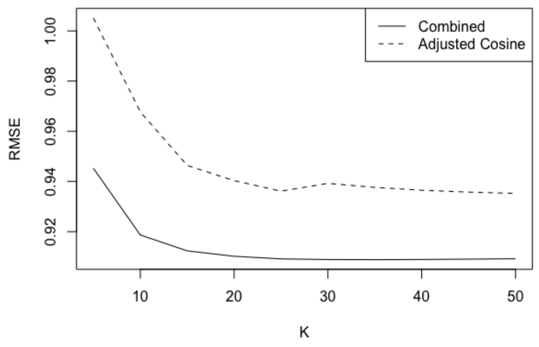
Table 3.5.: Comparison of prediction accuracy among state-of-the-art CFs on ML-1M dataset. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119].

Method	Parameterization	MAE	RMSE
ItemKNNAdjustedCosine	K=80	0.690876	0.879840
ItemKNNAdjustedCosineReg	K=80, reg_u=2, reg_i=3	0.690622	0.879012
BiasedMatrixFactorization	num_factors=120, bias_reg=0.001, regularization=0.055, learn_rate=0.07, num_iter=100, bold_driver=true	0.675454	0.853102
SVDPlusPlus	num_factors=20, num_iter=80, reg=0.05, learn_rate=0.005	0.667725	0.853002
ItemKNNCombinedReg	RegU=2, RegI=3, K=20, power=2.6	0.664384	0.852439
ItemKNNCombined	K=20, power=2.6	0.664270	0.852514

Memory-based collaborative filtering with the proposed similarity measurement also uses 1/3 to 1/4 total number of neighbors to predict compared to traditional KNN methods as shown in Tables 3.4 and 3.5. This means that it can predict 3 to 4 times faster than traditional KNN methods with pre-calculated similarities in the preference prediction phase. Over the ml-100K dataset, we use K=25 whereas traditional KNN with adjusted cosine uses K=60. For the ml-1M dataset, we use K=20 whereas traditional methods with adjusted cosine use K=80. For comparison purposes, when using the same K for both CFs with combined similarity and adjusted cosine, we can see that both MAE and RMSE of the CF with the combined similarity measurement are at least 3% (for both MAE and RMSE) lower than traditional KNN method using adjusted cosine as similarity measurement with K ranging from 5 to 50 in Figure 3.6 over ml-100K dataset.



(a) MAE performance with different nearest neighbor size K, where x-axis is K, y-axis is the MAE.



(b) RMSE performance with different nearest neighbor size K, where x-axis is K, y-axis is the RMSE.

Figure 3.6.: Prediction accuracy with different nearest neighbor size K

Netflix Dataset and Modified Dataset

Prediction accuracy in terms of MAE and RMSE is tested on the Netflix dataset using predefined Netflix probe and testing dataset [4]. CFs with the proposed similarity measurement performs better than traditional memory-based CF and model-based CF in terms of 0.3%-1.7% lower MAE as shown in Table 3.6. Note that the ratings in the probe dataset are also contained in the training dataset, which can benefit model-based CFs more than memory-based CFs since model-based CFs are trained with all testing data while memory-based CFs are trained with part of the testing data. More importantly, in practice, The actual rating that we are trying to predict is usually unknown. Therefore, another testing method is proposed by removing all ratings of probe testing datasets from the training datasets so that CFs are not benefiting from using testing data to train. It can be observed that the predication accuracy of all algorithms became worse in terms of higher MAE and RMSE compared to the result of using training dataset that contains probe dataset to train in Tables 3.6 and 3.8. However, using the proposed method, the MAE of the proposed method compared to traditional KNN with adjusted cosine lowered from 0.3% to 1.7% (from 0.002177 to 0.012415).

Table 3.6.: Comparison of prediction accuracy among state-of-the-art CFs on Netflix probe testing dataset. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119].

Method	Parameterization	MAE	RMSE
UserItemBaseline	reg_u=4.5, reg_i=1.137, num_iter=10,	0.768320	0.982610
BiasedMatrixFactorization	num_factors=80, learn_rate=0.005, reg=0.035, num_iter=26	0.712600	0.916900
ItemKNNAdjustedCosine	K=20	0.702826	0.920311
ItemKNNCombined	K=20, power=2	0.700649	0.927615
ItemKNNCombinedReg	K=20, power=2, RegU=10, RegI=25, iter=100	0.700598	0.927516

Table 3.7.: Comparison of prediction accuracy among state-of-the-art CFs on Netflix dataset with training dataset containing no ratings from the testing dataset. The table is ranked by MAE. The proposed method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119].

Method	Parameterization	MAE	RMSE
ItemKNNAdjustedCosine	K=20	0.728092	0.953960
BiasedMatrixFactorization	num_factors=80, learn_rate=0.005, reg=0.035 num_iter=26	0.726186	0.927115
ItemKNNCombined	K=20, power=2	0.715677	0.947513

Hubness experiments are conducted on MovieLens and Netflix Challenge datasets to see if prediction accuracy increased by using combined similarity measurement over using rating or structural based similarity measurements alone is due to the reduction of the hubness. Overall, with $K = 20$, the rating conditioned hubness H_R^{20} of combined similarity measurement is approximately 2 times lower than adjusted cosine similarity measurement across all 3 datasets. It is slightly lower than Ochiai similarity measurement except for ml-1M dataset.

Table 3.8.: Comparison of hubness and rating conditioned hubness between rating and structural based similarity measurements on MovieLens and Netflix dataset. Both the hubness and rating conditioned hubness decrease as the size of the neighbor list increases, with K=20, the rating conditioned hubness H_R^{20} of combined similarity measurement is approximately 2 times lower than adjusted cosine similarity measurement across all 3 datasets.

	Adjusted Cosine						Ochiai						**Combined**					
Dataset	H^5	H^{10}	H^{20}	H_R^5	H_R^{10}	H_R^{20}	H^5	H^{10}	H^{20}	H_R^5	H_R^{10}	H_R^{20}	H^5	H^{10}	H^{20}	H_R^5	H_R^{10}	H_R^{20}
ml-100K	6.2	5.2	3.7	6.9	5.4	**3.9**	4.1	3.1	2.4	3.6	3.0	**2.7**	6.0	4.9	3.8	3.4	2.7	**2.4**
ml-1M	12.2	9.2	7.5	8.1	9.0	**8.0**	6.5	4.4	2.9	7.2	4.5	**3.1**	7.5	5.2	3.5	7.5	5.1	**3.4**
Netflix	21.7	17.6	15.2	24.9	19.5	**16.3**	38.5	23.8	13.1	22.9	13.2	**13.4**	45.7	25.3	14.0	31.6	16.1	**8.9**

Netflix Dataset and Modified Dataset without cold start users

Since collaborative filtering methods, especially memory-based CFs, are prone to cold start problem [45] [46], experiments are performed to see how much prediction accuracy improvement can be achieved on non-cold start users. experiments in Section 3.6.1 were reconducting using both the Netflix training-probe dataset and training-probe separated dataset to predict on users with at least a certain number of ratings. Since the MovieLens datasets are preprocessed by selecting users who rated at least 20 items [115], with the larger Netflix dataset, users who rated at least 50 are selected to be non-cold start users. By comparing Tables 3.9 and 3.10, the predication accuracy of all algorithms became worse in terms of higher MAE and RMSE with predefined training-probe datasets in Table 3.9 compared to the result of using training-probe separated dataset in Table 3.10. Model-based CF (BiasedMatrixFactorization) performs better than the proposed method with training dataset that contains ratings from testing dataset as shown in Table 3.9. But when using the training dataset that doesn't contain ratings of the testing dataset, which is usually the case in practice, the proposed method still performs better than both traditional memory-based CF and model based CF in terms of lower MAE in Table 3.10. Therefore, memory-based CF with the proposed similarity measurement can perform better than model-based CFs on non-cold-start users on this dataset.

Table 3.9.: Comparison of prediction accuracy among state-of-the-art CFs on Netflix probe testing dataset for users with more than 50 ratings. The table is ranked by MAE. The proposed method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119].

Method	Parameterization	MAE	RMSE
ItemKNNAdjustedCosine	K=20	0.673137	0.879225
ItemKNNCombined	K=20, power=2	0.670398	0.885109
BiasedMatrixFactorization	num_factors=80, learn_rate=0.005, reg=0.035, num_iter=26	0.658173	0.839787

Table 3.10.: Comparison of prediction accuracy among state-of-the-art CFs on Netflix dataset with training dataset containing no ratings from the testing dataset for users with more than 50 ratings. The table is ranked by MAE. Our method is in bold font. Note: Model-based collaborative filtering methods are calculated with suggested hyperparameters using MyMediaLite library [119].

Method	Parameterization	MAE	RMSE
ItemKNNAdjustedCosine	K=20	0.698783	0.912899
BiasedMatrixFactorization	num_factors=80, learn_rate=0.005, reg=0.035, num_iter=26	0.692484	0.887858
ItemKNNCombined	K=20, power=2	0.682575	0.900724

3.6.2 Computational Resources

In this section, memory-based CFs and model-based CFs are compared with respect to computational time in order to show how much time is required versus model-based CFs, it is observed that memory-based CF takes 2-39 times less wall time to predict: SVDPlusPlus takes 78 mins, whereas ItemKNNCombined takes only 2 mins as shown in Table 3.11. Note that in order to make one single prediction, model-based methods take the same amount of time as shown in the Table 3.11 and entire item-user matrix to train while memory-based methods are able to predict using only the item vectors that a particular user has rated on in real time.

Table 3.11.: Comparison of time consumption between memory-based and model-based algorithms on ml-1M dataset. We choose one method that performs well in accuracy prediction from previous section since other methods are derived from those. We can see that memory-based CFs are much faster than model-based CFs. Model-based CF Library provided by [119]. Note that: all methods in this section are single-threaded coded. No parallelization is done for any method.

Algorithm	Wall Time (mins)
SVDPlusPlus	78
BiasedMatrixFactorization	4
ItemKNNCombined	2

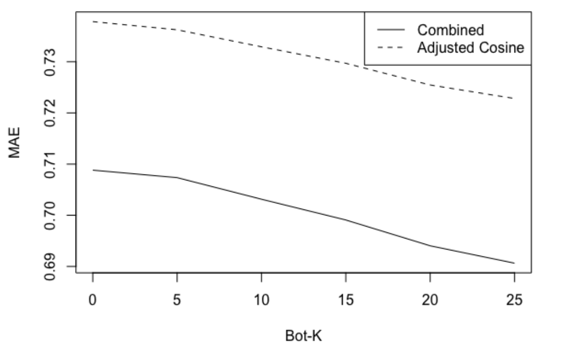
3.6.3 Performance without cold-start users

In this section, memory-based item-based CF with adjusted cosine and the proposed similarity measurement is compared with respect to prediction accuracy and coverage to see if there is a difference in prediction accuracy between the proposed method vs. traditional methods. Since memory-based CFs are prone to the Cold Start Problem, a better performance can also be achieved on prediction accuracy by not making predictions based on less than a certain number (5-25) of neighbors instead of cutting users with less than certain number (50) of ratings as shown in Section 3.6.1. Both the proposed method and KNN with adjusted cosine sacrifice prediction coverage (the number of actually predictions made w.r.t. total number of predictions expected to be made for testing dataset) as items with fewer than Bot-K neighbors are not predicted, but both methods share the same coverage with the same Bot-K.

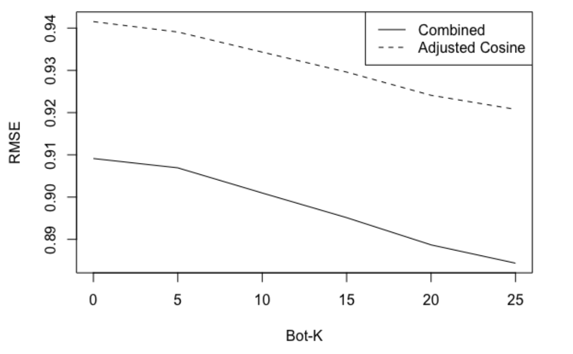
The proposed method is approximately 4% better (for both MAE and RMSE) than KNN using adjusted cosine as similarity measurement with Bot-K ranging from 5 to K-1 as shown in Table 3.12 and Figure 3.7. Furthermore, The differences of MAE and RMSE between KNN with adjusted cosine and the method stay the same with Bot-K ranging from 0 to K-1.

Table 3.12.: Prediction accuracy and coverage with bottom-K cutoff. Bot-K is the number of minimum neighbors used to make a prediction. ItemKNNCombined is using K=25, α=2.6.

Bot-K cut off	**Combined-MAE**	AdjCos-MAE	**Combined-RMSE**	AdjCos-RMSE	Coverage(%)
0	0.708815	0.737840	0.909131	0.941535	100
5	0.707348	0.736244	0.90692	0.939081	99.3828
10	0.703131	0.732921	0.900971	0.934320	95.826
15	0.699072	0.729698	0.895132	0.929595	90.9246
20	0.694031	0.725457	0.888688	0.924096	86.1294
25	0.690634	0.722816	0.884356	0.920776	81.3963



(a) MAE Performance For Different Bot-K Cutoff



(b) RMSE Performance For Different Bot-K Cutoff

Figure 3.7.: Accuracy Performance For Different Bot-K Cutoff

3.7 Discussion

3.7.1 Prediction Accuracy

The proposed similarity measurement can improve the prediction accuracy of traditional KNN method for the MovieLens and Netflix datasets in terms of lower MAE as shown in Section 3.6.1. Since the number of co-rated users between each pair of items $|C_{ic}|$ and the degree distribution among items varies as shown in Figure 3.3, the structural similarity $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ varies among the K nearest neighbors as well. Thus, the MAE of CF using the proposed similarity measurement $S_{combined}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ would be lower than the traditional CF method using rating-based similarity measurements $S_{rating}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ only. The performance of CF with the proposed similarity measurement and adjusted cosine similarity would yield the same MAE or RMSE only when the structural similarity measurement $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ is the same between all pairs of items. The structural similarity $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ in the numerator and denominator in Equations 3.5 and 3.6 will be canceled and make the new proposed similarity measurement equal to adjusted cosine. This would happen when the popularity of each item $|\boldsymbol{i}_i|$ is the same and the number of co-rated users between all pair of items $|C_{ic}|$

is the same, which could hardly be true in practice as we can see from the degree distribution of MovieLens and Netflix dataset follows a power law distribution in Figure 3.2.

The combined similarity measurement also requires 2/3 to 3/4 times fewer neighbors to achieve a better (4% lower in MAE) prediction accuracy compared to traditional KNN methods that uses structural or rating-based similarity measurement alone using MovieLens datasets. Since structural similarity measurement and rating-based similarity measurement are combined together to compensate each other, big hubs that occur when calculating using rating-based similarity measurement or structural similarity measurement alone, but do not contribute much personal preferences, are not frequently among the nearest neighbor lists. The rating conditioned hubness of combined similarity measurement is approximately 2 times lower than adjusted cosine similarity measurement across MovieLens and Netflix Datasets. 2/3 to 3/4 times fewer nearest neighbors are able to represent the user's opinion, a 2/3 to 3/4 shorter nearest neighbor list is needed to make an accurate prediction compared to traditional CFs over the MovieLens Dataset.

Although model-based CFs are known to have better prediction accuracy in terms of lower MAE than memory-based CFs on MovieLens and Netflix datasets, we can see that KNN memory-based CF with the proposed similarity measurement can achieve lower MAEs than sophisticated model-based CFs on MovieLens and Netflix Datasets. Although model-based CFs that utilize the entire user-item space can be potentially more unbiased with more information included in the training, with the nearest neighbor list ranked by the proposed similarity measurement, local neighbor items of the predicting item can be more representative of the user's opinion, thus achieve lower MAE. Therefore, local KNN CFs can perform better in prediction accuracy than model-based CFs with similarity measurements that are more representative of true relationships between items.

Since RMSE penalizes large errors, model-based CFs with entire dataset to train performs better on error regularization than memory-based CFs, thus achieving lower

RMSE. Comparing to memory-based CFs that only use the associated user information and items this user has rated on to predict. Memory based CFs can get higher RMSE compared to model-based CFs when users tend to give extreme ratings more often. With pre-processed datasets such as MovieLens datasets, memory based CF can perform better than model-based CFs in terms of both lower MAE and RMSE.

The prediction accuracy of memory-based CFs with proposed combined similarity measurement can achieve better prediction accuracy than most state-of-the-art Graph Convolutional Network (GCN) approaches and identical to Multi-Component graph convolutional Collaborative Filtering (MCCF) (MAE: 0.7050, RMSE: 0.9070) on MovieLens100K dataset reported in [105]. Moreover, except MCCF, the prediction accuracy of those GCN approaches reported in [105] perform slightly worse than SVD++ over MovieLens100K dataset. Though GCN approaches have good performance, they generally require more computational resources or information to train the model compared to memory-based CFs.

3.7.2 Computational Resources

Model based CFs require the entire user-item matrix $\mathbf{R}$ to train. Consequently, they require 2-39 times more time and memory in order to make one single prediction compared to memory-based CFs as shown in Section 3.6.2. Although once the training is done, the prediction can be made in real time. However, in order to make the most accurate prediction, the training process needs to be repeated after each new rating is made. On the other hand, memory-based CFs only require the associated user and the item information that this user has rated on to make a prediction. They require much less memory and time compared to model-based approaches which require the entire matrix to train. More importantly, the prediction can be done in real time when new ratings are made. Furthermore, with the proposed similarity measurement, we can predict with fewer neighbors versus traditional KNN CFs that use adjusted cosine, and so require less time to predict.

3.7.3 Performance without cold-start users

KNN CF with the proposed similarity measurement perform better with sufficient amount of items rated by this predicting user as shown in Section 3.6.3. Memory-based CFs are very intuitive to understand compared to model-based CFs such as SVD that use hidden factors to describe the underlying reasons of why a user likes an item. Although memory-based CFs suffer from the Cold Start Problem, it is more easily understandable and explained why a user likes an item and why a particular prediction is made. Then, active learning can be conducted [45] to specifically gather more data to better understand user preference and then make a more reliable prediction. While with model-based CFs, it is unable to explain why a user likes a item and why predictions are made based on hidden factors which are not descriptive, it would be challenging to form a strategy on how we could understand user preference better by gathering ratings of which items from users.

3.8 Conclusion

In this research, we proposed a general framework to combine similarity measurement to be used in KNN memory-based collaborative filtering methods that improves the prediction accuracy over state-of-the-art CF methods. This is accomplished without losing the advantages that memory based CF methods require less memory and time compared to model-based CF methods. Furthermore, with the proposed similarity measurement, KNN memory-based CFs use fewer neighbors to predict compared to traditional KNN memory-based CFs as well.

Future work will incorporate time into recommendation systems by utilizing dynamic models of user preference dynamics. We will investigate on under what conditions time would affect recommendation systems.

4. IMPROVING MEMORY-BASED COLLABORATIVE FILTERING BY CONSIDERING CONTRAST EFFECTS, ANCHORING AND ADJUSTMENT BEHAVIORS

4.1 Abstract

Consumers rate items anchoring based on the ratings they have given to similar items recently or based on the overall average ratings of this item given by other consumers. The score of rating for an item can be a relative measure of how consumers think of this item based on their own past experiences, or whether they believe this item is overrated or underrated by all other consumers. Therefore, the score of the rating given by a user can be biased because of the user's past experience such as the sequence of the ratings, or the average of the item rated at the time when the user rates this item. Two techniques are proposed in this paper to reduce the biases caused by those user comparing, anchoring and adjustment behavior by modifying the similarity measurements used in memory-based collaborative filtering.

4.2 Introduction

4.2.1 Time-aware Recommendation Systems

A product or service is not judged only by its own characteristics, but also by the characteristics of other products or services offered concurrently [11]. It can also be judged by anchoring based on users' memories [12–14]. Rating or satisfaction is viewed as a function of the discrepancy or contrast between expected and obtained outcomes [15, 16]. This is documented as contrast effects [17]. Thus, a rating given to an item by a user is a comparative opinion based on the user's past experiences.

Therefore, the score of ratings (e.g., 1-5) can be affected by the sequence and time of ratings. However, in traditional collaborative filtering, pairwise similarities measured between items do not consider time factors such as the sequence of rating, which could potentially introduce biases caused by contrast effects. Furthermore, the user averages used in adjusted cosine or item averages used in Pearson Correlation Coefficient similarity measurement are overall averages of all ratings. In those similarity measurements, item averages or user averages are used as a threshold of whether a user liked an item. Those averages can shift over time. Using a fixed overall average as a threshold can give us inaccurate opinions of items from users at that given time. Thus, 2 techniques are proposed in this paper to incorporate time factors into memory-based collaborative filtering by modifying similarity measurements to reduce biases introduced by consumer behaviors such as contrast effects, anchoring and adjustment.

4.2.2 Summary of Main Contributions

In this paper, works about recommendation systems that considers time factors and propose a new approach based on memory-based collaborative filtering method are reviewed. How hubness and prediction accuracy of item-based collaborative filtering in terms of MAE (Mean Absolute Error) improved by using our approach. The main contributions of this paper are:

- A time-aware similarity measurement for both structure-based and rating-based similarity measurements to be used in memory-based collaborative filtering is proposed to overcome the limitations of: 1) unreliable measurements of pairwise relationships between items being rated over a long time apart due to user interest shifts; 2) highly rated but dissimilar items being measured as similar items by traditional similarity measurements due to broad user preferences.

- The experimental results of our proposed approach, user anchoring and adjustment behavior with recent or all data anchoring on user average or item average using Pearson Correlation Coefficient and Adjusted Cosine as similarity measurements on 3 most widely referred datasets: MovieLens100K, MovieLens-LatestSmall, and Netflix Challenge datasets are provided. The results show: 1) The prediction accuracy and hubness of memory-based CF improve by using our time-aware similarity measurements; 2) Time affects the prediction accuracy for both datasets collected via offline survey (users rates all movies at once) or on-line rating (users rate one movie after they watched) system ; 3) Users compare items and enlarge the differences in ratings that are rated close to each other in time for online rating systems; 4) The differences between items rated within minutes to each other for offline survey system are narrowed due to the rigidity of the integer rating scale; 5) Hubness of memory-based CF does not change via random removal except under some certain conditions; 6) Users anchor and adjust their ratings not only on simple measures such as item or user averages.

4.3 Background

4.3.1 Related works

One of the most widely used approaches to deal with user interest shifts is by time window, or referred as instance selection, time truncation [26,121]. It only learns the user interests from the most recent observations [122] [123]. There are studies on how prediction accuracy can be improved by adjusting the window size. [124] showed that the prediction accuracy can be further improved by using heuristics to adjust the size of the window . [125] showed that a time-based function to determine which old observations are outside a certain time window should be neglected. However, there are some cases that old observations are also important compared to new ones, to avoid loss of useful information, some of the systems also learn from old observations [122]. Hybrid models combining both short-term and long-term model of user interests were also developed [126] to include observations that are old, but important. The method learns from the recent observations first. If a story cannot be classified using short-term model, then it uses long-term model to learn from old observations.

As time-window approach can lose too much signal [31], models that assign weights on the ratings with different decay functions are also developed. For example, a gradual forgetting function with linear decay is developed as natural forgetting is a gradual process [98]. With recent data being more important, [28] used an exponential weight function to assign data importance based on time. Different clusters of items or users can also have different decay rates.

There are also approaches that model time as one of the factors trained in model-based approaches. Global effects, such as ratings may fall with time after the movie's initial release dates are also considered in deriving interpolation weights [127]. Time-factors can also be modeled as biases [100]. Users are profiled based different time cycles such as day, month or year [99]. Model-based SVD++ is introduced to capture gradual concept drift [31]. Multiple time attributes can be derived, such as season, week, or time of the day, to recommend [128]. [19] found that users tend to rate items in similar context in a short period of time, which they modeled as sessions. [129] considered rating sequence into matrix factorization model to improve prediction accuracy. Users shift preferences due to : 1) new item exploration, 2) users' past experience, 3) popular item bias, 4) neighbors' influence. [22] used a tensor-matrix factorization method to include all those factors to capture user preference dynamics. [130] used a overlapping community detection algorithm to include time factors. [131] factorized the coupled tensor by weighting the importance of past user preference by giving more weights on users with more rare side information.

There are also some additional approaches. Instead of assigning weights on the rating, weights of similarities between items can also be assigned differently. It can be based on the deviation of the rating on the item from the most recent rating [29]. It can also be a measure of time similarity between items [132].

4.3.2 Memory-based collaborative filtering

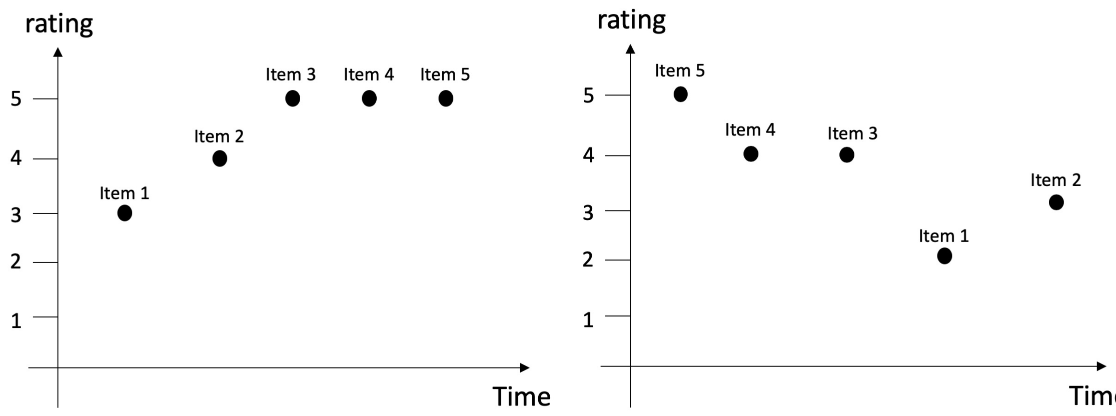
Our approach is based on memory-based CFs. Memory-based CFs can be item-based or user-based [10]. There are 2 steps for both approaches: similarity calculation and preference prediction. Without loss of generality, taking item-based approach, the detailed steps are in previous chapter, Sections 3.3.2 and 3.3.3. To help readers understand better, terminologies and notations are defined in previous chapter, Section 3.3.1.

4.4 Proposed Approach

Our approaches focus on modifying similarity measurement to reduce biases introduced by consumer behaviors such as contrast effects, anchoring and adjustment. In contrast, most literatures apply exponential decays on ratings to make predictions more reflective of recent user preferences. However, Koren pointed that time-window and decay cannot work as those techniques loose too much information about the data [31]. in our approaches, instead of applying time-window and decay during prediction phase for each user, they are applied in similarity measurement phase only. In this way, each user's individual subjective opinions are preserved as rating data are not filtered or ignored for each user based on how recent this rating is during the prediction phase. Instead, Time windows are applied based on how far apart those two items are rated in time from one user. Therefore, if two movies that are considered similar from one user will be considered similar in the future as well if those two movies are rated close in time. Since users may shift their preferences and dislike a movie they used to like, they are likely to dislike the other similar movie as well if those 2 movies are rated close in time, as they reflects the preference of that user from the same time period. Furthermore, as similarity measurements are an aggregation of opinions between two items among all users who have rated both items. Ignoring parts of opinions would not affect the prediction coverage. So 2 approaches are proposed to estimate similarities between items and compare the hubness after applying our approach. At last, the comparison between using item average and using user average as anchoring point in rating-based similarity measurements is performed.

4.4.1 Reducing contrast effects in rating-based similarity measurements

As items are judged by anchoring based on users' memories [12–14], the contrast effects that users give ratings by comparing recent items, becomes the first bias to ignore. Figure 4.1 illustrates the behaviors caused by contrast effects(anchoring and adjustment). By looking at the figure, the cores of the ratings may be affected by the sequences of the ratings.



(a) Ratings of User X who watched and rated items in the order of item 1,2,3,4,5

(b) Ratings of User X who watched and rated items in the order of item 5,4,3,1,2

Figure 4.1.: The scores of ratings affected by the sequence of ratings. Ratings given may be anchored and adjusted based on user memories. In Fig 4.1a, User X rated item 3, and later found that item 4 and item 5 were better, but the highest rating this user can give is 5. So the user gave item 4 and 5 with ratings of 5 as well. While in Fig. 5.6a, User X rated the best item 5, and then found item 4 and 3 were not as good as item 5, so he gave lower ratings to those items. The score of the ratings are affected by the sequence of ratings.

To capture the phenomena that the scores of ratings could be affected by the sequences of ratings caused by contrast effects, the part of the Eq. 4.1 for the item being rated later are raised to the power of β, where $\beta > 0$, to enlarge or reduce the differences between the later rating and the item average to capture the phenomena that opinions of later rated items may be anchored based on previous rated items. Rating-based similarity measurements such as adjusted cosine now becomes Eq. 4.1. By choosing $\beta > 1$, contrast effects are enlarged. The difference between the later rating and the reference average are enlarged. In this way, the rigidity of the integer rating scale can be compensated. If 2 items are rated with the same score, the later item now becomes more or less favorable by the user depending on both ratings are rated higher or lower than the average. On the other hand, by choosing $\beta < 1$, the contrast effects are reduced. The phenomena that the difference of the scores between two items that are rated close in time are enlarged due to comparison is compensated. By adjusting β, the relationship between 2 items measured from the ratings from the user become less biased if users give comparative opinions anchored on recent rated items.

$$S^t_{rating,\beta,\gamma}(\boldsymbol{i}_i, \boldsymbol{i}_c) = \frac{\sum_{x \in C_{ic}} z_i^{\gamma_i} \left((|\mathbf{R}_{ix} - \bar{\mathbf{R}}\boldsymbol{u}_x| + 1)^{\beta^{\gamma_i}} - 1 \right) \times z_c^{\gamma_c} \left((|\mathbf{R}_{cx} - \bar{\mathbf{R}}\boldsymbol{u}_x| + 1)^{\beta^{\gamma_c}} - 1 \right)}{\sqrt{\sum_{x \in C_{ic}} \left(z_i^{\gamma_i} \left((|\mathbf{R}_{ix} - \bar{\mathbf{R}}\boldsymbol{u}_x| + 1)^{\beta^{\gamma_i}} - 1 \right) \right)^2} \sqrt{\sum_{x \in C_{ic}} \left(z_c^{\gamma_c} \left((|\mathbf{R}_{cx} - \bar{\mathbf{R}}\boldsymbol{u}_x| + 1)^{\beta^{\gamma_c}} - 1 \right) \right)^2}}, \tag{4.1}$$

where $\beta \in (0, +\infty)$, $z_i = sgn(\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}) = \frac{\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}}{|\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}|}$,$z_c = sgn(\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}) = \frac{\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}}{|\mathbf{R}_{cx} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}|}$,$\gamma$ is the binary parameter to determine which item is later rated, which is defined as:

$$\begin{cases} \gamma_i = 0, \gamma_c = 1 & t_c > t_i \\ \gamma_i = 1, \gamma_c = 0 & t_c < t_i \end{cases} \tag{4.2}$$

Two time distance measurements of how far between co-rated items in time are introduced to be used as a threshold to determine whether to apply the modified $S^t_{rating,\beta,\gamma}(\boldsymbol{i}_i, \boldsymbol{i}_c)$. They are: time distance and item distance. Time distance measures the natural time distance (ex. days) between the time when each user rated those 2 items. Item distance measures how many items were rated between those 2 items in time by this user. Similarity measurement is modified only on pairs of items that are rated close to each other in time determined by those 2 measurements as in Eq. 4.1.

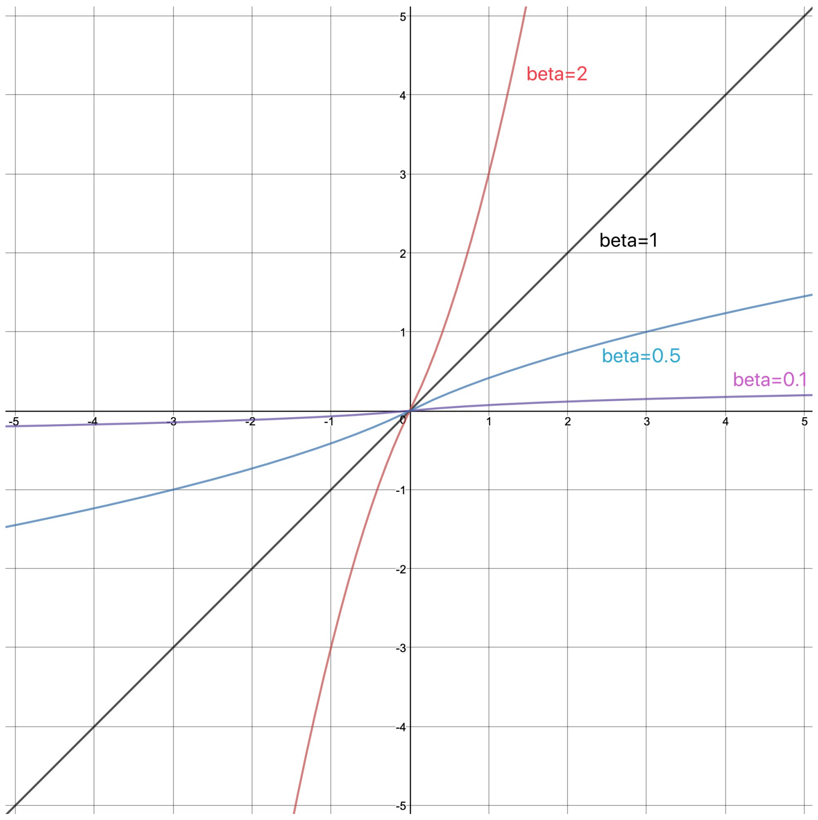


Figure 4.2.: Plot of $y = z_i \left(\left(|\mathbf{R}_{ix} - \bar{\mathbf{R}}_{\boldsymbol{u}_x}| + 1 \right)^{\beta} - 1 \right)$. Contrast effects can be enlarged or reduced by choosing different β. when $\beta > 1$, the contrast effects are enlarged, when $\beta < 1$, the contrast effects are reduced.

For pairs of items that are not close to each other in time, traditional adjusted cosine is used as contrast effects happen among items rated close to each other in time.

4.4.2 Ignoring relationship between items rated long time apart in structural similarity measurements

Structural similarity measurements can be biased because of item popularity since structural similarity measurements consider two items similar if those two items share lots of co-rated users. However, two popular items may not be highly correlated. But using traditional structural similarity measurements would make them seem so popular items usually share lots of co-rated users. But users' preferences may change over time, items being rated by the same user may not be similar if the time distance between the ratings of those two items from the same user is far. Traditional structural similarity doesn't consider time distance to capture whether those two items are rated far in time. So it doesn't consider user preference shifts across time or item popularity. Therefore, to lower the similarities between popular but not necessarily similar items, if the ratings of 2 items by the same user are rated far in time based on time distance measurements, the relationship of this pair of items formed by this user is neglected, in which those pairs of items are excluded in structural similarity measurements. The time distance between the rating of pairs of items are again: *natural time distance* and *item distance*. Structural similarity measurements now become in Table. 4.1. With this modified structural similarity measurement, The hubness of items is expected to reduce and the prediction accuracy is expect to increase as similarities between popular but not necessarily similar items are reduced.

Table 4.1.: Lists of some time-aware structural similarity measurements approximated locally. $|C_{ic}^t|$ denotes the pairs of items that are rated by the same user within a certain natural time distance or item distance. Note that those measurements do not require global information such as the total number of users. All information required for the calculation can be obtained from those 2 associated vectors $\boldsymbol{i}_c$ and $\boldsymbol{i}_i$.

Structural similarity measurement	Definition $S_{struct}^t(\boldsymbol{i}_i, \boldsymbol{i}_c)$
Common neighbor	$\|C_{ic}^t\|$
Jaccard	$\|C_{ic}^t\|/\|\boldsymbol{i}_i \cup \boldsymbol{i}_c\|$
Salton/Ochiai	$\|C_{ic}^t\|/\sqrt{\|\boldsymbol{i}_i\| \cdot \|\boldsymbol{i}_c\|}$
Sorensen	$2\|C_{ic}^t\|/(\|\boldsymbol{i}_i\| + \|\boldsymbol{i}_c\|)$
HPI	$\|C_{ic}^t\|/min(\|\boldsymbol{i}_i\|, \|\boldsymbol{i}_c\|)$
HDI	$\|C_{ic}^t\|/max(\|\boldsymbol{i}_i\|, \|\boldsymbol{i}_c\|)$
LHN1	$\|C_{ic}^t\|/(\|\boldsymbol{i}_i\| \cdot \|\boldsymbol{i}_c\|)$

By applying time-aware modifications on the structural and rating based similarity measurements incrementally to reduce those biases mentioned in previous 2 sections is expected to lead to an improvement in prediction accuracy in terms of lower MAE.

However, as each technique is limiting the amount of data used during calculation. Using less and less data could make similarity measurement calculation more prone to Cold Start Problems, which too few data are used that could leads to unreliable representation of the relationship between items too. So the performance of prediction accuracy will improve at first by ignoring biased opinions of pairs of items and then decrease when too many pairs of items are ignored in the structural or rating-based similarity measurements, which similarities between items are biased towards opinions from very few users.

4.4.3 Hubness

Hubness of item-based CF using time-aware rating-based similarity measurements will not change compared to the ones without considering time. This is because that rating-based similarity measurements are consensuses among all users who have rated both items as those measurements are aggregated over opinions of all pairs of items rated by the same user. They consider 2 items similar if most users rate both items above or below user averages or item averages. Time-aware rating-based similarity measurements exclude opinions of items rated close to each other in time from the same user. It is very unlikely that 2 items are rated close to each other in time by most users in reality as it is not likely that most users watch movies in the same sequence. Therefore, it removes only a small portion of opinions on pairs of items that are rated close to each other. How much exactly is the portion of pairwise opinions removed is shown in Thm. 4.4.1 and 4.4.2. Those removed pairwise opinions can be unreliable due to contrast effects as mentioned in the previous section. Again, rating-based similarities are consensuses over co-rated users. By removing a small portion of unreliable consensuses, the structure of the item-based network isn't affected. Only the edges with rating-based similarities as weights get adjusted without considering those unreliable opinions. Therefore,hen choosing $x \ll n$, what has been removed is a portion of the opinions from the consensuses from all co-rated users over the pairwise items. As total number of ratings n grows, the portion being removed gets even smaller. So not much change in the hubness of item-based CF using time-aware rating-based similarity measurements compared to the ones without considering time is expected.

Theorem 4.4.1. *The total number of pairwise opinions over items from one user rated within item distance of x is: $n_2^x = (x-1)n - \frac{(x-1)x}{2}$, where n is the total number of items this user rated, x is the total number of items rated between 2 items with regard to time, $x \in \mathbb{N}_{>1}$, $x \leq n$.*

Proof. Let x be the item distance, n_x be the total number of pairwise opinions, then:

$$\begin{array}{cc} x & n_x \\ 2 & n-1 \\ 3 & (n-1)+(n-2) \\ 4 & (n-1)+(n-1)+(n-3) \\ \vdots & \vdots \\ x & (n-1)+(n-2)+(n-3)+\cdots \\ & +(n-x-1) = (x-1)n - \frac{(x-1)x}{2} \end{array}$$

□

Given that x is a constant, the rate of number of pairwise opinions grows in $\mathcal{O}(n)$. However, the total number of pairwise opinions n_2^x from one user grows exponentially with regard to the total number of ratings this user gives. To be exact, that is:

Theorem 4.4.2. *The rate of total number of pairwise ratings n_2 from one user grows in $\mathcal{O}(n^2)$, where n is the total number of ratings from this user.*

Proof. The number of pairwise opinions from one user is:

$$n_2 = \binom{n}{2}, \tag{4.3}$$

where n is the total number of ratings from this user. As

$$n_2 = \binom{n}{2} = \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2}, \tag{4.4}$$

then,

$$n_2 = \mathcal{O}(n^2). \tag{4.5}$$

$\square$

However, hubness of item-based CF using time-aware structural similarity measurements will decrease compared to the ones without considering time as popular but not necessarily similar pairs of items are ignored. Structural similarity measurements consider two items similar if those two items share a lot of users who rated both of them. As mentioned in previous section, popular items tend to share lots of co-rated users. Therefore, with structural similarity measurements, two items that are not necessarily similar in content but are both popular will be considered similar. One user can have variety of different interests and those preferences may shift over time. by considering all pairwise opinions, all those items that are not necessarily similar in content, but popular, will be considered to be similar due to that they share lots of co-rated users determined by structural similarity measurements caused by the varieties and shifts of user preferences. In item-based network, all items will be considered similar eventually as the network grows. Therefore, assuming that users may shift preferences gradually, the time-aware structural similarity measurements are proposed to remove pairwise opinions that are rated over a long period of time in terms of how many days or item distance they were rated apart by the same user to reserve the true dynamics between items. From Table. 4.1, structural similarity measurements are a function of the number of co-rated users. Taking Ochiai as an example, to reduce the hubness of popular items, the reduction is only applied over the nominator $|C_{ic}^t|$. Therefore, Similarities of popular items that are rated over a long period of time will not increase but be penalized by their popularities as the denominator stays the same. Therefore, the more popular the items are, the more reduction they will get in similarity measurements. Hubness is expected to reduce in this way if our assumption that users shift their preferences gradually and eventually rate more popular items is true. If user preferences do not change or user preferences change abruptly within the time distance threshold, there will be no change in hubness by using this approach.

Reduction in hubness of item-based CF using time-aware structural similarity measurements is not simply due to random reduction in the number of co-rated users. To further verify that hubness reduction is contributed by our strategical time-related removal approach, it can be compared with random removal by approving those following theorems:

Theorem 4.4.3. *Given a weighted graph $G(V, E, W)$, where $[w_{ij} = f(x_{ij})] \in W$, x_{ij} is an attribute of $e_{ij} \in E$, $\forall x_{ij} \geq 0$, $f(x_{ij}$ is a non-negative monotonically increasing function. For $G'(V, E, W)$, such that $\forall x'_{ij} = \alpha x_{ij}$, where $\alpha > 0$. Then the hubness $H(G') = H(G)$.*

Proof. we We are going to prove by contradiction. the hubness, which is the skewness of the distribution of the number of times each node occurs in the nearest neighbor lists of all other objects can change if the nearest neighbor lists change. The nearest neighbor lists can change because the rankings of the nearest neighbor lists changes. Assume that the ranking of the nearest neighbor list can be changed by decreasing or increasing $x'_{ij} = \alpha x_{ij}$ on every edge, this can not be true as it contradicts that w_{ij} is a non-negative monotonically increasing function. Therefore, nearest neighbor lists does not change, the hubness does not change.

First, let $\alpha \in (0, 1]$, that is $x'_{ij} \leq x_{ij}$, since $x'_{ij} = \alpha x_{ij}$. Before decreasing x_{ij} to x'_{ij}, 2 nearest neighbors of v_1: v_2 and v_3, ranked by:$f(x_{13}) \leq f(x_{12}) \implies x_{13} \leq x_{12}$ since w_{ij} is a non-negative monotonically increasing function.

When decreasing x on all edges to x' by the same ratio α, x' over all edges are decreasing at the same rate, $x_{13} \leq x_{12} \implies \alpha x_{13} \leq \alpha x_{12}$.

Assume that the ranking of the nearest neighbor list can be changed by decreasing $\forall x'_{ij} = \alpha x_{ij}$ on every edge, that is: after decreasing, $\alpha x_{13} \leq \alpha x_{12} \implies f(\alpha x_{13}) \geq f(\alpha x_{12})$.

However, this contradicts that $w_{ij} = f(x_{ij})$ is a non-negative monotonically increasing function. Thus, this can not be true.

Therefore, the nearest neighbor list can not be changed by decreasing x_{ij} on every edge. Thus, the hubness, which is the skewness of the distribution of the number of times each node occurs in the nearest neighbor lists of all other objects, does not change.

The proof is similar for $\alpha > 1$, where $\forall x_{ij}$ being increased to x'_{ij} by the same rate α.

$\square$

Note that if the function $f(x)$ is not monotonic, the hubness is possible to change. That is:

Remark. *Given a weighted graph $G(V, E, W)$, where $[w_{ij} = f(x_{ij})] \in W$, x_{ij} is an attribute of $e_{ij} \in E$, $\forall x_{ij} \geq 0$, W_{ij} is NOT a non-negative monotonically increasing function. For $G'(V, E, W)$, such that $\forall x'_{ij} = \alpha x_{ij}$, where $\alpha > 0$, Then the hubness $H(G')$ is not guaranteed to equal to $H(G)$.*

However, for those structure similarities used in CF - $S_{struct}(\boldsymbol{i}_i, \boldsymbol{i}_c)$ listed in the Table. 4.1 are all non-negative monotonically increasing function w.r.t. $|C_{ic}|$. So this phenomenon will not happen.

But the hubness of item-based CF using structural similarity measurements can be reduced by random removal under some condition. That is:

Theorem 4.4.4. *Given a weighted graph $G(V, E, W)$, where $[w_{ij} = f(x_{ij})] \in W$, x_{ij} is an attribute of $e_{ij} \in E$, $\forall x_{ij} \geq 0$, w_{ij} is a non-negative monotonically increasing function, $x < \theta \implies f(x) = 0$. For $G'(V, E, W)$, such that $\forall x'_{ij} = \alpha x_{ij}$, where $\alpha > 0$, Then the hubness $H(G') \leq H(G)$.*

Proof. We are going to proof by direct proof by showing that as the weights W_{ij} on each edge decreases at the same rate, the smallest weight will always have the highest likelihood to decrease to become under the threshold θ to let $f(x_{ij})$ become 0 first. Then these two nodes will become disconnected with each other and disappear among each other node's nearest neighbor list. Then the hubness decreases. With more weights become 0, the hubness continues to decrease until it becomes 0 with the graph becomes completely empty with no edge exists.

As Theorem 4.4.3 shows, when decreasing x_{ij} on all edges to x'_{ij} by the same ratio α, the hubness $H(G') = H(G)$.

However, $x_{ij} < \theta \implies f(x_{ij}) = 0$. The smallest weight W_{ij} among all other weights will always have the highest likelihood to decrease to 0 first since $f(x_{ij})$ is a non-negative monotonically increasing function, $f(x_{min}) = min(f(x_{ij})) \implies x_{min} = min(x_{ij}) \implies x'_{min} = \alpha x_{min} = min(x'_{ij}) \implies f(x'_{min}) = min(f(x'_{ij}))$.

With $W'_{ij} = f(x'_{ij})$ becomes 0, two nodes become disconnected. Thus, if one node is among the other node's nearest neighbor list, this node will no longer be among the other's nearest neighbor list now.

As hubness is the skewness of the distribution of the number of times each node occurs in the nearest neighbor lists of all other objects as described in Section **??**. The hubness will decrease.

With x_{ij} continuing to decrease on all edges, the 2nd smallest x_{ij} becomes under the threshold θ. The weight $w'_{ij} = f(x'_{ij})$ becomes 0. As x_{ij} continues to decrease over all edges, more weights on more edges become zero and become disconnected. More nodes disappear from the end of other nodes' nearest neighbor lists with the size of the nearest neighbor lists large enough to hold those nearest neighbors before decreasing x_{ij}.

With more 0s appearing as the number of times each node occurs in the nearest neighbor lists of all other objects, the hubness continues to decrease.

Eventually, the hubness of the graph is decreased to 0 as it becomes a completely empty graph. □

Although, the hubness of item-based CF using structural similarity measurements can be reduced by random removal according to Theorem 4.4.4, The prediction accuracy in expected to decrease as less neighbors are used to make a prediction.

The prediction accuracy and hubness between random removal and time-aware removal will be compared to see if the proposed strategy reduces hubness and increases prediction accuracy simply due to reducing the density of the data. By removing the same amount of co-rated users, the proposed strategy is expected to reduce more hubness and achieve better prediction accuracy.

4.4.4 Anchoring and Adjustment

In traditional collaborative filtering, all pairwise opinions are used to calculate similarity measurements. Reference points to determine whether a user likes an item or not, such as user rating averages or item averages during calculation, are approximated based on all data. In adjusted cosine or Pearson Correlation Coefficient(PCC), ratings of 2 items i, j from the same user u above the user average $\mathbf{R}_{iu}, \mathbf{R}_{ju} > \bar{\boldsymbol{u}}_u$ or each item's average $\mathbf{R}_{iu} > \bar{\boldsymbol{i}}_i, \mathbf{R}_{ju} > \bar{\boldsymbol{i}}_j$ are considered to be positively correlated. However, those reference points can shift over time $\bar{\boldsymbol{u}}_u^t \neq \bar{\boldsymbol{u}}_u^{t+1}$ as user rating scales or preferences shift. As user averages $\bar{\boldsymbol{u}}_u$ shift over time, the user average at the time when this user rated each item or the overall user average can all be different. A rating above the user average at time t the user rated $\mathbf{R}_{iu} > \bar{\boldsymbol{u}}_u^t$ can later at time $t+1$ become below the user average $\mathbf{R}_{iu} < \bar{\boldsymbol{u}}_u^{t+1}$ as the user average shifts over time. Then an item liked by the user at the time t the user rated can later be interpreted as the user disliked the item at time $t+1$. Using a static overall user average as a reference point during the similarity calculation to determine whether a user liked an item, the opinions of an item from a user at a previous timestamp can be interpreted totally different based on an average calculated at a later time using all data as shown in Fig 4.3. Thus, similarity measurements such as adjusted cosine or PCC calculated using static user or item averages can be unreliable as they are approximated with all opinions from a user to evaluate an opinion made earlier. Unreliable similarities used in CFs can lead to inaccurate prediction accuracy.

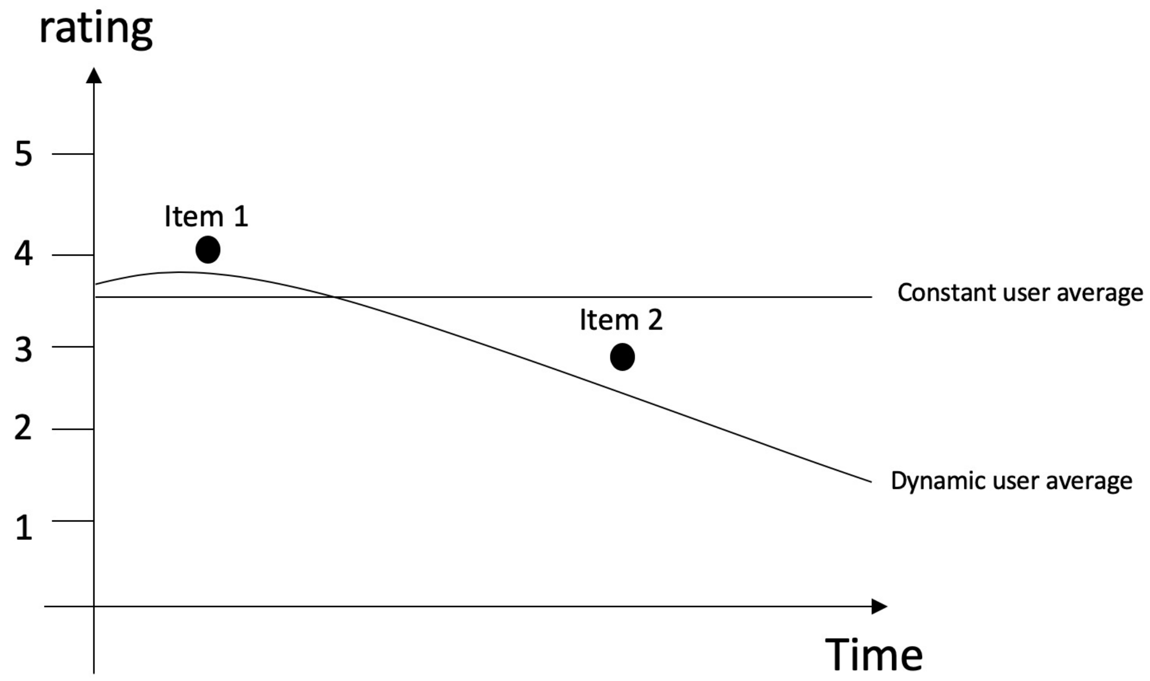


Figure 4.3.: Static average v.s. dynamic average. Using constant user average as the reference point, item 1 and item 2 will yield a negative relationship, since item 1 is above the average while item 2 is below. Using dynamic user average as the reference point, item 1 and item 2 will yield a positive relationship.

Users may be anchoring based on how others think of the item (item averages) or their own memories, which is the user averages in this case. A comparison of both PCC using accumulative item average with ratings before the user made the rating v.s. the static overall item average and Adjusted Cosine using user averages calculated using only most recent ratings from the user v.s. static overall user average can be performed to show if there is any difference. A comparison of the performance of prediction accuracy of CF with PCC v.s. Adjusted Cosine over datasets which users do or do not know the item average ratings when they made the rating can show if users rate anchoring on item averages.

CF with PCC is expected to perform better over the datasets which users know the item average rating when they made the rating, as users may anchor on the item average during rating, while CF with Adjusted Cosine will perform better over the datasets which users do not know the item average rating when they made the rating, as users may be anchoring on their user experiences, which may be more related to user averages.

4.5 Experiment Design

4.5.1 Data Sets

For comparison purposes, the MovieLens and Netflix datasets are used as they are the most widely referenced in literature (See Table 4.2). For ML100K and Netflix257, the last rating of each user based on timestamp is separated into testing dataset and the rest is in training dataset. For MLNew and Netflix2761, the last 10% rating of each user based on timestamp is separated into testing dataset and the rest is in training dataset. One difference is that over ML100K dataset, it was collected in a survey fashion. Users rate at most hundreds of movies within a day. While over the Netflix and MLNew datasets, Assume users rate each movie right after they watched the movie. The other difference is that: users in ML100K dataset don't know the item average ratings when they made the ratings. Users in Netflix and MLNew datasets know the item average ratings when they made the ratings.

4.6 Experimental Results

4.6.1 Time affecting prediction accuracy

First, the real-world data is used to examine how the contrast effects indicated by literatures affect the accuracy of recommendation systems. The dataset chosen is

Table 4.2.: Data sets. Datasets vary in different sizes (number of users, number of items, number of ratings, and density). MovieLens datasets have been pre-processed with users who had less than 20 ratings or did not have complete demographic information were removed [115]. ML100K dataset contains data collected through MovieLens web site (movielens.umn.edu) from 1997/09/19 through 1998/04/22. MLNew contains data collected between March 29, 1996 and September 24, 2018. Netflix datasets are selected sub-dataset from Netflix Prize dataset which contains data collected from 1998/10 through 2005/12 and reflects all ratings received during this period. Netflix contains users who have rated at least 20 items and been active for more than 1 day. Netflix257 contains less ratings with less users compared to Netflix2761. Density is the ratio of the number of actual ratings to the possible maximum number of ratings.

Data Set	Number of Users	Number of Movies	Number of Ratings	Density
MovieLens 100K (ML100K) [115]	943	1,682	100,000	6%
MovieLens Latest Small (MLNew) [115]	9,742	610	100,836	1.7%
Netflix257 [4]	257	1,000	10,008	4%
Netflix2761 [4]	2,761	1,000	104,697	4%

a well-known dataset used by most recommendation systems as a benchmark called MovieLens 100K. More details about this dataset can be referred in Section 4.5.1. Three sets of experiments on 4 sub-datasets generated from MovieLens 100K were conducted. Each dataset is separated into training and testing datasets by users. Each testing dataset contains the last rating of each user based on the timestamp. The rest of the ratings are in training dataset. The training and testing dataset $S_{U>1}$ contains ratings from 291 users who are active for more than 1 day. The training and testing dataset $S_{U\leq 1}$ contains ratings from 636 users who are active for less than 1 day.

Then, the users were ranked based on the number of ratings each user has and separated users into 2 datasets: S_{U+} and S_{U-}. The training and testing datasets of S_{U+} contains ratings from top 292 users based on the ranking of the number of ratings, which contains more rating. The training and testing datasets of S_{U-} contains the ratings from the rest 645 users, which contains less rating. The size of dataset $S_{U>1}$ is close to that of S_{U+}. The size of dataset $S_{U\leq 1}$ is close to that of S_{U-}. Model-based CF SVD++ and memory-based CF with adjusted cosine and our proposed similarity measurement from [133] were conducted on those 4 datasets. The results are shown in Table 4.3.

By comparing the MAE of $S_{U>1}$ and $S_{U\leq 1}$, users who are active for more than 1 day yield about 13-15% (Adjusted Cosine, SVD++, Combined) better prediction accuracy. But it may be because of that users who are active for more than 1 day also have more ratings to train with. To investigate whether more ratings leads to better prediction accuracy. By comparing the MAE of S_{U+} and S_{U-}, it shows that more ratings do contribute to better prediction accuracy. But when looking at the differences of MAE between $S_{U>1}$ and $S_{U\leq 1}$ and the differences of MAE between S_{U+} and S_{U-}, differences of MAE between $S_{U>1}$ and $S_{U\leq 1}$ is approximately 5% larger than the differences of MAE between S_{U+} and S_{U-}, which indicates that the number of ratings is not the only factor that contributes to better prediction accuracy. Users who are active for less than one day tend to rate all movies within a short period

Table 4.3.: Prediction accuracy of memory-based and model-based CF performed on 4 datasets: $S_{U>1}$: users active for more than 1 day, $S_{U\leq 1}$: users active for less than 1 day, S_{U+}: users with more ratings, S_{U-}: users with less ratings. By comparing the MAE of S_{U+} and S_{U-}, more number of ratings can potentially contribute to better prediction accuracy of CFs(but not necessarily, by comparing the MAE of S_{U-} and $S_{U\leq 1}$ of SVD++). By comparing the MAE of $S_{U>1}$ and S_{U+} users who are active for longer time have better prediction accuracy than users who have more ratings.

	Adjusted Cosine		SVD++		Combined	
Dataset	MAE	RMSE	MAE	RMSE	MAE	RMSE
$S_{U>1}$	0.713728	0.918298	0.710057	0.921500	0.698495	0.894182
$S_{U\leq 1}$	0.834777	1.075180	0.828599	1.046482	0.827064	1.061490
S_{U+}	0.769130	1.006780	0.752907	0.972794	0.735208	0.970204
S_{U-}	0.835165	1.052060	0.817157	1.025473	0.825090	1.051230

of time. Some rate movies every few minutes or seconds. They may rate movies based on their memories and take the rating as a survey. Their memories may not be reliable to reflect their true ratings. While users who are active for more than 1 day tends to rate movies at least every few hours or days. Assume those users rate movies right after they watched the movie. So the rating may be more reliable. By further comparing the MAE of $S_{U\leq 1}$ and S_{U-}, SVD++ and CF with combined similarity measurement get lower MAE for S_{U-}, which indicates that more rating doesn't necessarily contributes to better prediction accuracy. Therefore, separating users by active time shows that time is a factor that contribute to prediction accuracy.

4.6.2 Reducing contrast effects in rating-based similarity measurements

The approach in Section. 4.4.1 is applied to the pairs of items rated by the same user within a time distance and item distance on dataset $S_{U>1}$ (explained in Section 4.6.1) and got the results in Table 4.4. Traditional memory-based CF with adjusted cosine can now perform 0.5% better than model-based CF SVD++. Memory-based CF with our combined similarity measurement improves 1% in terms of lower MAE when enlarging the contrast effects with $\beta > 1$ compared to the one (S_{U+} Combined in Table 4.3) without considering the time. Over Netflix257 dataset, which has approximately the same number of users as dataset $S_{U>1}$, reducing contrast effects lowered MAE. Choosing β close to 0 neglects the pairs of items that are rated close to each other yields the best result. Compared to the MAE without time in Table. **??**, our combined+ method that neglects pairs of items rated within item distance of 4 gives us a 1.4% improvement in MAE.

	MAE	RMSE	Parameter
AdjCos	0.726202	0.937236	topN=30
Combined	0.707023	0.916561	topN=30 α=2.6

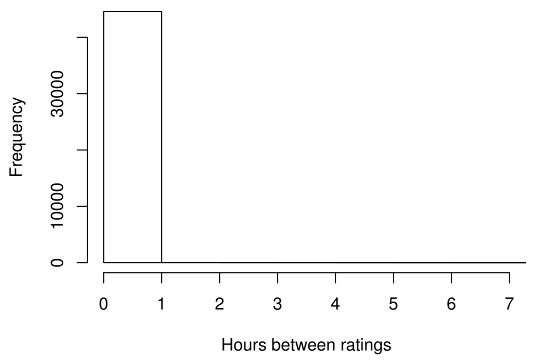
Table 4.4.: Results for enlarging contrast effects in rating-based similarity measurements. Traditional memory-based CF with adjusted cosine can now perform better than sophisticated model-based CF SVD++. Memory-based CF with our combined similarity measurement improves 1% in terms of lower MAE compared to S_{U+} Combined in Table 4.3.

	MAE	RMSE	Parameter
AdjCos	0.713728	0.918298	
Combined	0.698495	0.894182	
AdjCos+	0.709950	0.913432	power = 1.15 session= 1hr
Combined+	0.690399	0.907679	power=4 session=1hr
Combined+	0.686851	0.901488	power=8 session=3 items

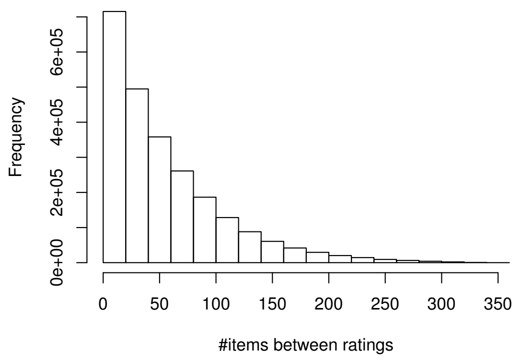
Table 4.5.: MAE of combined+ on Netflix257 dataset with α=2.6, topN=30: best prediction accuracy is achieved by setting β close to 0 to neglect similarities of pairs of items rated close to each other(within a time distance, or within an item distance) in final similarity aggregation. Using item distance as selecting criteria performances better than using time distance.

	β=0.01	0.05	2
1 day	0.705928	0.706	0.713306
1 week	0.70474	0.705594	0.708456
1 month	0.704851	0.703818	0.713187
1 item	0.701893	0.703934	0.715352
2 items	0.705934	0.705315	0.711787
3 items	0.702577	0.703745	0.713396
4 items	**0.696801**	0.701337	0.713051
5 items	0.697233	0.699512	0.713963

The distribution of item distance and time distance over S_{U+} and Netflix257 dataset is examined to understand why enlarging the contrast effects works better over S_{U+}, while reducing or neglecting the pairs rated too close to each other in time works better over Netflix257 dataset (See Figs. 4.4,4.5). It turned out that users in MovieLens dataset rate movies very frequently like a survey, while users in Netflix257 rate movies at least a few hours or days after watching each movie.

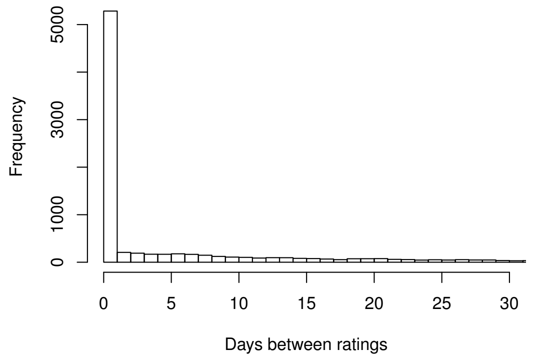


(a) Distribution of time differences of items rated next to each other by the same user for ML100K dataset

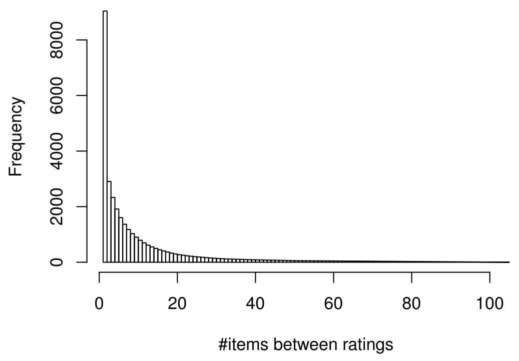


(b) Distribution of number of items being rated within 1 hour by the same user for ML100K dataset

Figure 4.4.: Distribution of time distance and item distance of all pairs of items rated by the same user over ML100K dataset



(a) Distribution of time differences of items rated next to each other by the same user for Netflix257 dataset



(b) Distribution of number of items being rated within 1 day by the same user for Netflix257 dataset

Figure 4.5.: Distribution of time distance and item distance of all pairs of items rated by the same user over Netflix257 dataset

Table 4.6.: MAE of Ochiai with structural similarity cutoff on Netflix257 dataset with α=2.6, β=0.01, topN=30.

N	itemDist<N	timeDist<N	itemDist<N%*$\lvert \boldsymbol{u}_i \rvert$	timeDist<N%*$T(\boldsymbol{u}_i)$
10	0.706093	0.709055	0.710428	0.719146
20	0.708270	0.711792	**0.704031**	0.711037
30	0.710478	0.712353	0.709794	0.714870
40	0.710022	0.715786	0.709625	0.712801
50	0.709791	0.719761	0.711649	0.718054
60	0.711901	0.719217	0.713215	0.716313
70	0.714322	0.718554	0.713919	0.712742
80	0.711829	0.717690	0.715343	0.711432
90	0.711277	0.715113	0.713619	0.710775
100	0.712398	0.716477	0.713996	0.713996
Static Ochiai	0.713996			

4.6.3 Ignoring relationship between items rated long time apart in structural similarity measurements

Next, relationship formed long time apart is ignored in structural similarity measurements introduced in Section. 4.4.2 over Netflix257 to examine if the prediction accuracy can be improved. Two item and time distance thresholds were used: static and dynamic. Static item distance threshold ignores pairs of items that are rated more than a certain item distance apart. Dynamic item distance threshold ignores items that are rated apart more than a certain percentage of total number of items this user has rated. While static time distance threshold ignores pairs of items that are rated more than a certain days apart. dynamic time distance threshold ignores items that are rated more than a certain percentage of active days (days between first and last rating) of a user. Ochiai similarity is used as an example of structural similarity measurement. The results are shown in Table. 4.6 and Fig. 4.6. Using a dynamic item distance with 20 items yield the lowest MAE of 0.704031, which is about 1.4% lower than static Ochiai without considering time.

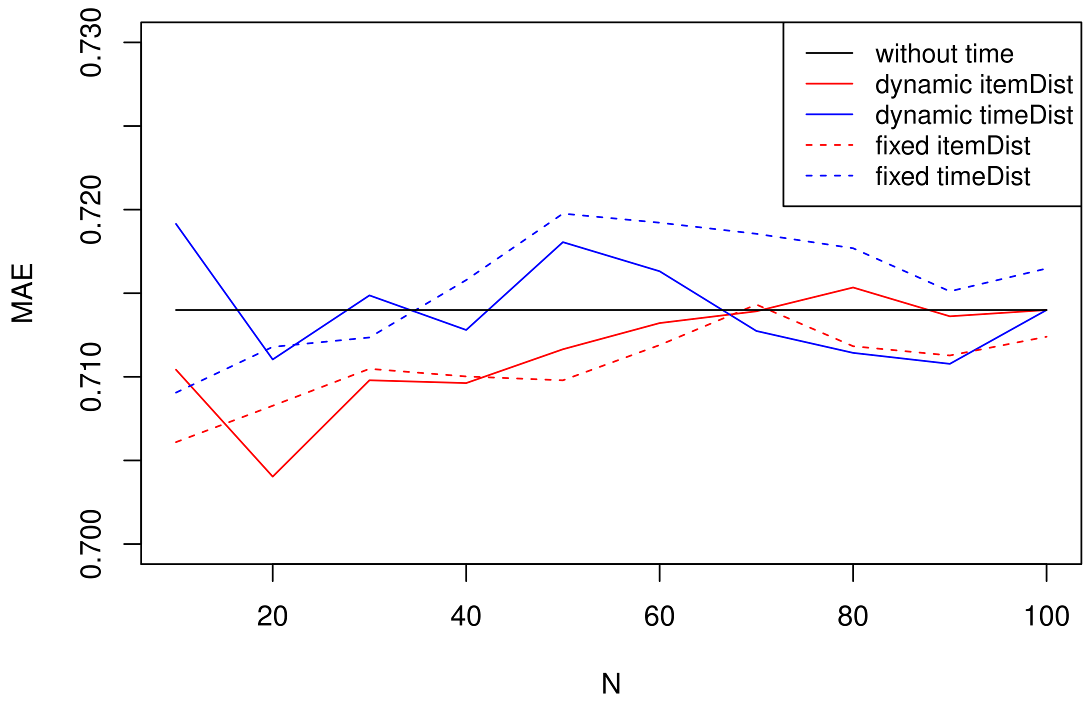


Figure 4.6.: MAE of Ochiai with structural similarity cutoff on Netflix257 dataset with β=0.01, topN=30. Using a dynamic item distance with 20 items yield the lowest MAE of 0.704031, which is about 1.4% lower than static Ochiai without considering time.

Then, the same approach with combined similarity measurement is performed over Netflix257 to see if relationship formed long time apart is ignored in structural similarity measurements could improve the prediction accuracy. The results are shown in Tab. 4.7 and Fig. 4.7. There is only an improvement of 0.3% in MAE by using dynamic time distance.

Table 4.7.: MAE of combined+ with structural similarity cutoff on Netflix257 dataset with α=2.6, topN=30. There is only an improvement of 0.3% in MAE by using dynamic time distance.

N	itemDist<N	timeDist<N	itemDist<N%*$\lvert \boldsymbol{u}_i \rvert$	timeDist<N%*$T(\boldsymbol{u}_i)$
10	0.726780	0.741209	0.734249	0.730346
20	0.718433	0.730557	0.734641	0.737068
30	0.705206	0.726725	0.716630	0.732055
40	0.706586	0.721525	0.717111	0.723287
50	0.711158	0.720189	0.707323	0.710828
60	0.708677	0.714149	0.717750	0.705783
70	0.710804	0.720011	0.711336	**0.704815**
80	0.706399	0.720809	0.707582	0.705452
90	0.705485	0.719271	0.706103	0.706489
100	0.704925	0.725100	0.707023	0.707023
Static Combined	0.707023			

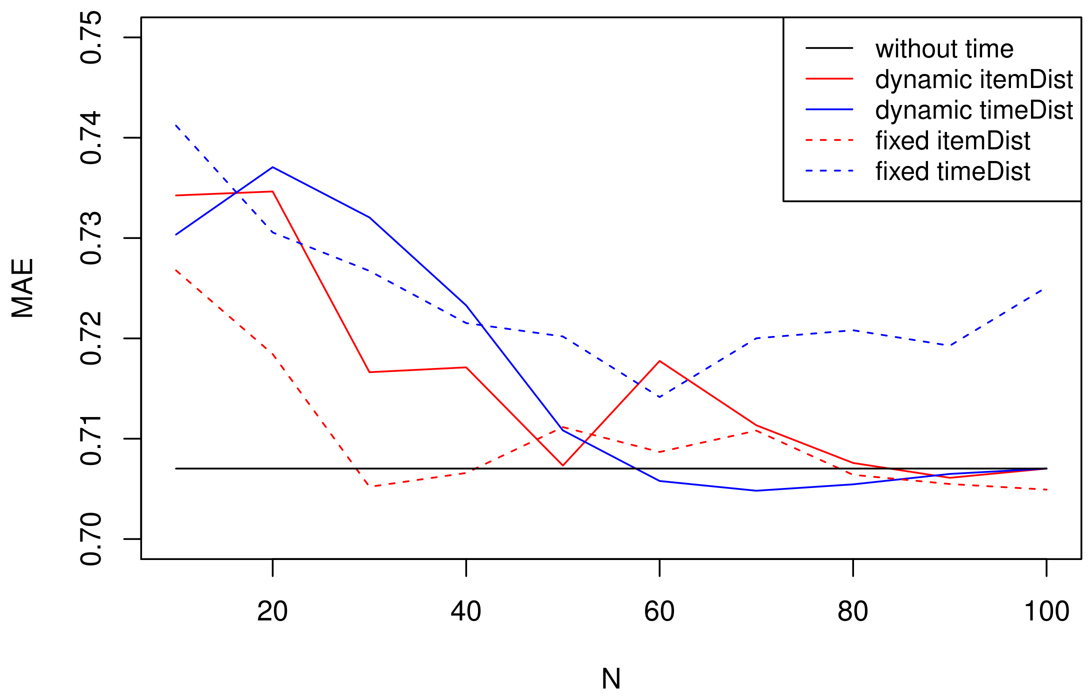


Figure 4.7.: MAE of combined+ with structural similarity cutoff on Netflix257 dataset with α=2.6, topN=30.

Finally, the approaches of reducing contrast effects in rating-based similarity measurements and ignoring relationship formed long time apart in structural similarity measurements were combined together to examine if there is an improvement in prediction accuracy over the Netflix257 dataset. The results are shown in Table. 4.8 and Fig. 4.11. The MAE is improved by 1.3% over the combined method with neglecting items rated close to each other within item distance of 4 in rating-based similarity measurement and 2.7% over the static combined method.

Table 4.8.: MAE of combined+ with structural and rating similarity cutoff on Netflix257 dataset with α=2.6, β=0.01(itemDist=4), topN=30.

#	itemDist¡#	timeDist¡#	itemDist¡%*$\|\boldsymbol{u}_i\|$	timeDist¡%*$T(\boldsymbol{u}_j)$
10	0.718879	0.702213	0.716909	0.706304
20	0.700994	0.704779	0.722187	0.713683
30	0.691385	0.708916	0.693583	0.715001
40	**0.687667**	0.700999	0.698405	0.707568
50	0.692490	0.707992	0.690667	0.695967
60	0.692871	0.705808	0.699012	0.690531
70	0.697553	0.701944	0.696385	0.692553
80	0.694137	0.701593	0.696463	0.690865
90	0.693688	0.696626	0.695633	0.694505
100	0.693211	0.701254	0.696801	0.696801
Static Combined	0.707023			
Combined+ Rating neglect(itemDist=4) 0.696801				

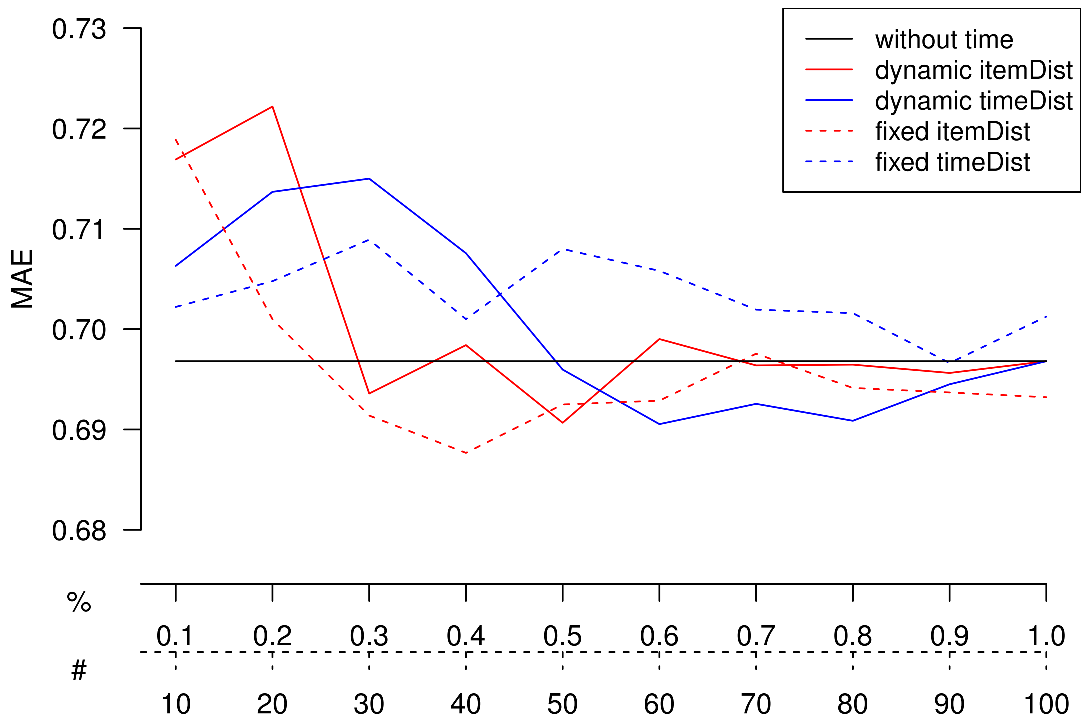


Figure 4.8.: MAE of combined+ with structural and rating similarity exclusion on Netflix257 dataset with α=2.6, β=0.01(itemDist=4), topN=30. The MAE is improved by 1.3% over the combined method with neglecting items rated close to each other within item distance of 4 in rating-based similarity measurement and 2.7% over the static combined method.

4.6.4 Coverage and Hubness

Although pairs of items rated close to each other are excluded to reduce contrast effects in rating-based similarity measurements and pairs of items rated far to each other are excluded to reduce relationship formed due to popularity of items and user preference shifts in structural similarity measurements. The coverage, which is how many predictions that are able to make, does not decrease as shown in Table. 4.9. Because similarities are aggregated over multiple users. Excluding some pairs from one user does not affect the overall coverage.

However, the hubness is dramatically reduced at most 70% with CF using combined+ compared to traditional adjusted cosine. The result is shown in Table. 4.9 and Fig. 4.9. Combined+ method with both techniques achieves the lowest hubness. That means the most popular highly rated items are least frequently among the nearest neighbor lists when using combined+ as similarity measurement. Hubness of combined+ decreased up to 43% compared to static combined method. However, this may also be caused by excluding pairs of items alone. By excluding pairs of items rated within item distance of 4 in rating-based similarity measurement, 14% of pairs are excluded. By excluding pairs of items rated more than item distance of 40 away in structural similarity measurement, 22% of pairs are excluded.

14% and 22% (same amount of pairs being ignored using threshold cutoff) pairs of items are randomly excluded in rating-based similarity measurement and structural similarity measurement accordingly to see if hubness is reduced simply due to less pairs of items considered. The hubness of all rating-based, structural-based and combined similarity measurements with excluding the same amount of pairs of items randomly do not change much compared to the ones without random excluding. However, by comparing random excluding and our approach with the same density of pairs excluded, hubness is reduced over structural (Ochiai+) and Combined+ (Both) similarity measurements. Hubness of strategically excluded Ochiai+ is reduced up to 25% compared to Ochiai with randomly excluding the same amount of pairs of items. Our proposed Combined+ (Both) with excluding pairs of items in both rating-based and structural similarity measurements further reduced the hubness up to 45% compared to Combined (Both) with random excluding the same amount of pairs. By comparing randomly and strategically excluding pairs of items in rating-based and structural similarity measurements separately, excluding strategically in structural similarity measurement contributes to the hubness reduction in Combined+ method since the hubness of Combined+ (Rating itemDist<4) is the same as randomly excluded Combined(random exclude 22% Rating), while the hubness of Combined+

Table 4.9.: Comparison of hubness between rating-based, structural-based and combined similarity measurement with and without considering time effects. Combined+ with both techniques achieves the lowest hubness.

Similarity Measurement	H_R^5	H_R^{10}	H_R^{15}	H_R^{20}	H_R^{25}	H_R^{30}
AdjCos	5.8	3.9	3.1	2.5	2.2	1.9
Ochiai	5.2	4.1	3.3	3.0	2.8	2.5
Combined	3	2	1.7	1.4	1.2	1.1
AdjCos (random exclude 14%)	6	4	3	2.5	2.1	1.8
Ochiai random exclude 22%)	5.2	3.9	3.2	2.8	2.5	2.2
Combined (random exclude 22% Rating)	2.9	2	1.6	1.4	1.2	1
Combined (random exclude 14% Struct)	3.1	2.2	1.6	1.3	1.1	1
Combined (both)	3.1	2.2	1.6	1.3	1.1	1
AdjCos+	5.8	3.9	3.1	2.5	2.2	1.8
Ochiai+	3.9	2.7	2.4	2.3	2.3	2.1
Combined+ (Rating itemDist¡4)	3	2	1.7	1.4	1.2	1
Combined+ (Struct itemDist¿40)	1.7	1.2	1.1	1.1	1	1.1
Combined+ (Both)	**1.7**	**1.2**	**1**	**1.1**	**1**	**1**

(Struct itemDist>40) is reduced compared to Combined (random exclude 14% Struct) and is the same as Combined+ (Both).

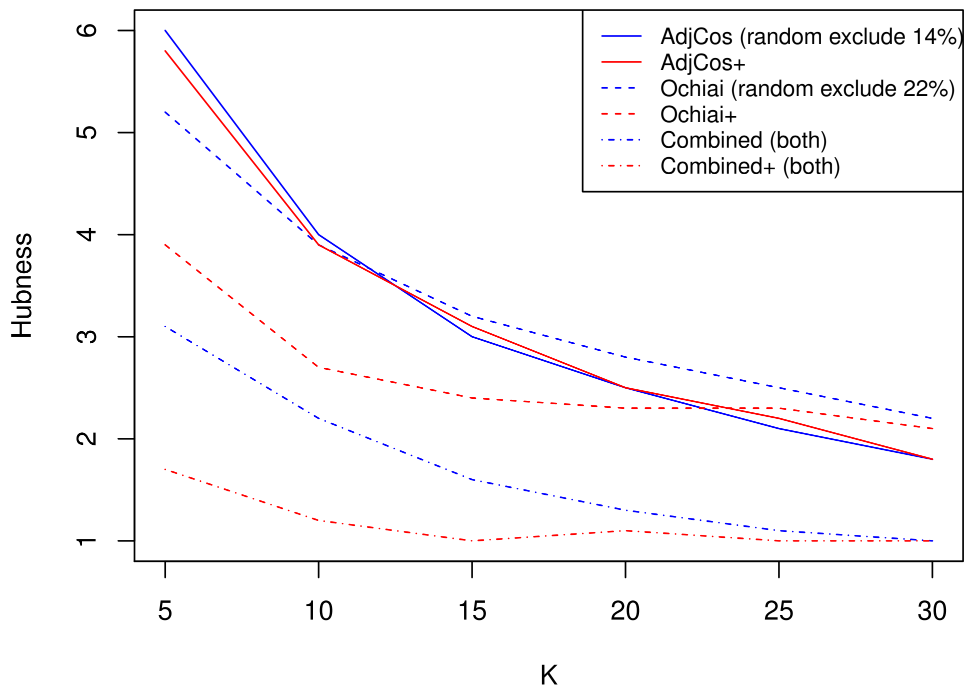


Figure 4.9.: Comparison over Hubness (H_R^K) of KNN list with similarities randomly and strategically exclusion. Combined+ (both) achieves the lowest hubness. Hubness of Ochiai+ is less than Ochiai with random exclusion, while there is no significant difference in hubness between strategically and random exclusion over rating-based AdjCos (random exclude 14%) and AdjCos+.

The experiment of randomly removing pairs of items in Ochiai similarity measurement is performed to see if the hubness can be reduced to the same level of Ochiai+, which was removed strategically based on one of the time factors. The result is showed in Fig. 4.10. Ochiai+ (Struct itemDist<40) excluded 14% of pairs of items, which achieved hubness (H_R^5) of 3.9. Different amount of pairs over Ochiai ranging from 0% to 90% were randomly excluded to see how much density is needed to be excluded in order to achieve the same hubness of Ochiai+. Note that the threshold θ in Theorem 4.4.4 here is equal to 1 as at least 1 co-rated user is needed to create an edge between nodes. The result shows that the same hubness of Ochiai is achieved by removing 80% of the pairs of items randomly over Ochiai similarity measurement. However, the prediction accuracy in terms of MAE of Ochiai start to decrease dramatically after excluding 40% of pairs.

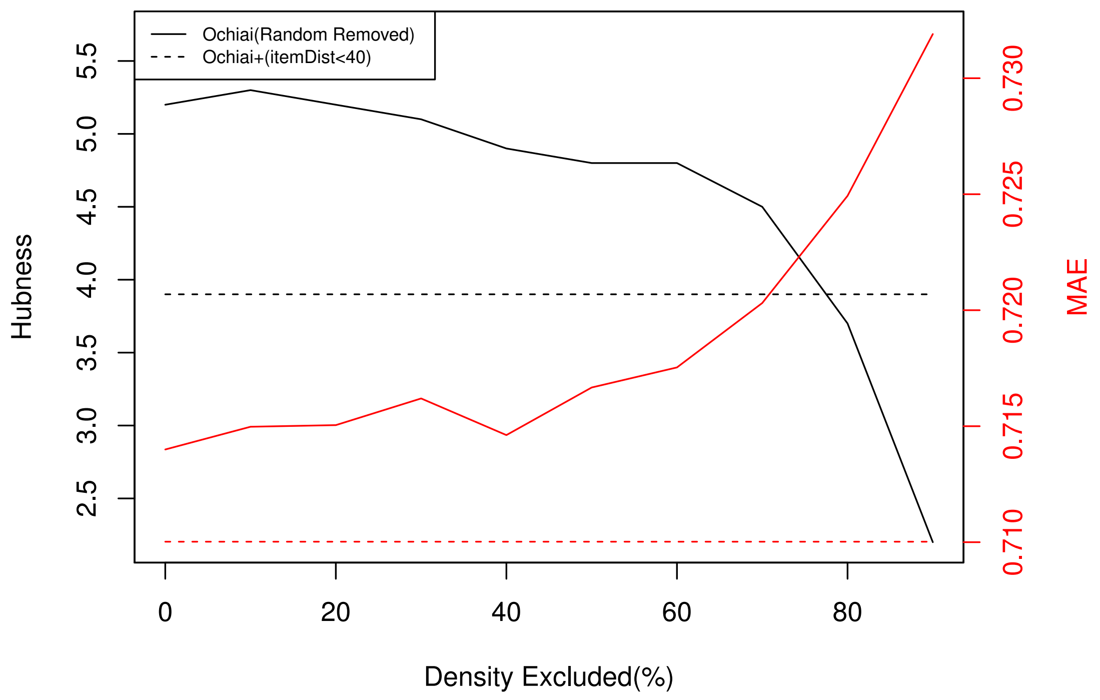


Figure 4.10.: Comparison of prediction accuracy and hubness (H_R^5) of KNN list with Ochiai+ and Ochiai with different density of pairs of items excluded. To achieve the same hubness of Ochiai with 14% pairs excluded, Ochiai needs to remove at least 80% of pairs. However, Prediction accuracy (MAE) of Ochiai is dramatically decreased after excluding 50% of pairs.

4.6.5 Anchoring and Adjustment

The prediction accuracy of CFs using PCC between with accumulative item averages (PCC+) with ratings till the time the user made the rating and with static overall item averages (PCC) over the Netflix2761 dataset were first compared. Different numbers of items (0-100) as the warm up period were used for the accumulative averages to examine if different warm up period would improve the prediction accuracy. Then the same comparison was performed over Adjusted Cosine between with accumulative user averages (AdjCos+) with ratings till the time the user made the rating and with static overall user averages (AdjCos). The performances over the combined methods with PCC (CombPCC+) and Adjusted Cosine(CombAdjCos+) was also compared. At last, the prediction accuracy of CFs between using adjusted cosine with user averages calculated with most recent items each user had rated before the user rated this item and using adjusted cosine with overall static user averages was compared. The results are shown in Fig. 4.11. There is not any significant improvement in prediction accuracy in terms of lower MAE by using dynamic user or item averages in rating-based similarity measurements. But PCC or CombPCC with item averages as the reference points usually works better than adjusted cosine or CombAdjCos with user averages as the reference points over this Netflix2761 dataset, which means users knowing the item averages when rating, may anchor more on item average than user averages.

Then the PCC and Adjusted Cosine over $S_{U>1}$ (extracted from ML100K) and Netflix257 were compared. Both datasets contains around 200-300 users. As seen in 4.4b and 4.5b, over MovieLens dataset, it was collected in a survey fashion. Users rate at most hundreds of movies within a day. While over the Netflix dataset collected by the content provider, assume users rate each movie right after they watched the movie. The results of prediction accuracy of CFs with PCC and Adjusted Cosine in terms of MAE are shown in Table 4.10. Without knowing the item averages when rating, CF with adjusted cosine using user averages as reference points performs better than with PCC. With knowing the item averages when rating, CF with PCC using item averages as reference points performs better than with adjusted cosine. The same sets of experiments were performed over the larger Netflix2761 and MLNew datasets which was collected over a longer period of time with users knowing the item averages when rating for both datasets. CF with PCC performs better than adjusted cosine over both datasets which users know item averages when rating, indicating that users may anchor more on item averages than user averages when knowing item averages.

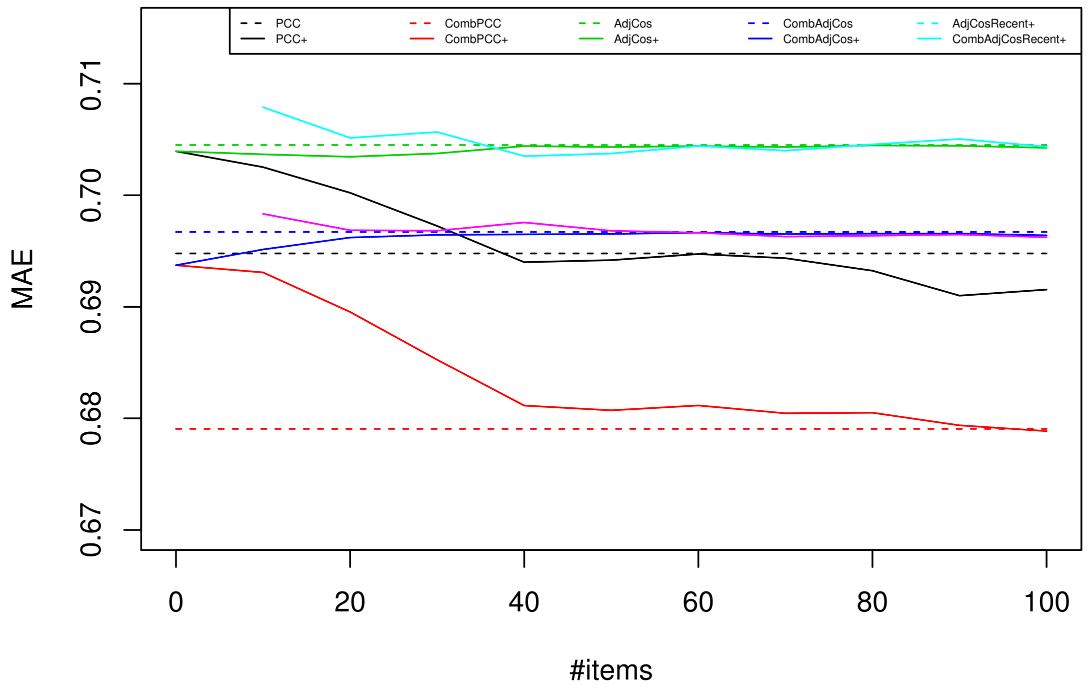


Figure 4.11.: MAE of Scene building techniques. #items means the number of items excluded as warm up period in accumulative average calculation in PCC, AdjCos methods, and number of items included in AdjCosRecent methods. There isn't much difference in MAE of CFs using similarity measurements with dynamic averages compared to static averages.

Table 4.10.: Comparison of prediction accuracy (MAE) of CFs using PCC v.s. Adjusted Cosine over the ML100K and Netflix257 datasets. CF with PCC performs better over Netflix257, Netflix2761 and MLNew, which users know item averages when rating. CF with adjusted cosine performs better over ML100K which users don't know item averages when rating.

	ML100K	Netflix257	MLNew	Netflix2761
PCC	0.714789	**0.70424**	**0.68525**	**0.704707**
AdjCos	**0.708824**	0.726202	0.69611	0.718571

4.7 Discussion

The prediction accuracy of item-based memory-based collaborative filtering(CF) can be improved by neglecting pairs of items rated close to each other in time by the same user in rating-based similarity measurements. For survey-like rating datasets such as MLens100K, users watched all movies and then rated all movies at once like taking a survey. Users tend to have lower contrast effects. However, the rating differences between movies are limited by the rigidity of the integer rating scale. Thus, enlarging contrast effects improves the prediction accuracy of CFs by enlarging the difference between ratings of items rated close to each other to restore the true differences of ratings between those compared items. For content subscribing websites such as Netflix, assume users rate right after they watch the movie. Users may compare movies they watched more recently and introduce biases when rating. Therefore, by reducing pairs of items rated close to each other in similarity measurement improves the reliability of the similarities. Thus, it improves the prediction accuracy of CFs with those time-aware rating-based similarity measurements over the traditional rating-based similarity measurements.

The prediction accuracy of item-based memory-based CF can also be improved by neglecting pairs of items rated far from each other in time by the same user in structural similarity measurements. For structural similarity measurements, two items are considered similar if two items share large portion of users who rated both items. However, as users may shift preferences and develop new interests over time, items rated by the same user but over a long period of time doesn't necessarily mean those 2 items are similar. However, traditional structural similarity measurements without incorporating time factors may consider them correlated. Thus, neglecting pairs of items rated across a long period time reduces the possibilities of dissimilarity items being considered similar in structural similarity measurements. Therefore, it reduces the possibility that user opinions over pairs of items rated over long period of time from each other get the chance to represent the same opinions from the same user but over 2 items that are actually dissimilar. Thus, it improves the prediction accuracy of CFs with those time-aware structural similarity measurements over the traditional structural similarity measurements.

Hubness can also be reduced to improve prediction accuracy by strategically excluding pairs of items in structural similarity measurements. Again, for structural similarity measurements, whether two items are considered similar is based on the ratio of the number of co-rated users over total number of users who rated either item. If the ratio is high, then 2 items are considered similar. Big hubs in

By excluding pairs of items, the number of co-rated users get reduced, while the total number of users who rated either item stays the same. Thus, pairs of two popular items co-rated by lots of users, which are considered big hubs in structural similarity measurements, have higher chance of being excluded randomly. Therefore, by randomly excluding pairs of items in structural similarity measurement aggregation, hubness of KNN list using structural similarity measurements gets reduced, while rating-based similarity measurement is not affected by random exclusion since rating-based similarity measurement is a consensus measured over only co-rated users on whether users like both items. It is not affected by change in the ratio of the number of co-rated users over users who rated either item.

Hubness can also be reduced as highly rated items, which may be introduced by contrast effects, are now neglected in rating-based similarity measurements and highly correlated while dissimilar pairs of popular items are also neglected in structural similarity measurements. Lower hubness reserves better user preferences as popular liked items are not referred again and again. Thus, reducing the hubness can also potentially improves the prediction accuracy of CFs.

Users give ratings anchoring based either on user memories or how others think of this item and then make adjustments. If they think this item is worse than how others think of, they may give a rating lower than the item average. If item average is not provided, they may give ratings based on their past experiences. This phenomenon can be seen by comparing the prediction accuracy of CFs using PCC or adjusted cosine with item average or user average accordingly over different datasets, which item averages are provided or not. With item averages provided, CFs with PCC using item averages tend to yield a better prediction accuracy, while with item averages not provided, CFs with adjusted cosine using user averages tend to yield a better result.

4.8 Conclusion

In this research, a general framework is proposed to consider time in similarity measurements used in KNN memory-based collaborative filtering methods that improves the prediction accuracy and reduces hubness over traditional methods without considering time. This is accomplished by adjusting contrast effects introduced by user comparing items rated close to each other in time and removing pairs of items rated long time apart in time as user preference and rating scale may shift over time. The comparison of prediction accuracy of CFs between using PCC and using adjusted cosine with item average and user average over datasets which users are aware of item averages when rating or not is shown. With item averages provided, the prediction accuracy of CFs is better using PCC with item averages.

Future work will consider how to utilize the contrast effects into user preferences and rating scale. When does the user anchor more on user experiences and when does the user anchor more on item averages will be investigated. How much peer review pressure affects the user rating scale.

5. QUANTIFICATION AND VISUALIZATION OF USER MOVIE GENRE SHIFTS AND DYNAMIC RECOMMENDATION USING COLLABORATIVE FILTERING

5.1 Introduction

User preference dynamics refer to questions such as: whether and how users change their preference across different categories? If people do change their preference across different categories, do they prefer different categories at the same time period or they prefer only 1 category at a time? Are the user dynamic patterns of the same user across different categories different? Do they stop shopping from the previous category instantly or move to a new category gradually? Do they ever come back to the previous category? If so, how frequent is it? It is therefore desirable to have a transparent recommendation system that could shed light on these questions in an intuitive manner.

Users may or may not shift their preferences due to various reasons: [134] studied how a mental disorder could shift a user's musical preference. The study found that although most patients did not change music preferences after the onset of a mental disorder, a group of patients who had a shifts reported that music had impaired them during the time of illness, and a third group stopped listening to music completely. Musical preferences may also shift as users age [135–137]. [137] found that musical preferences are subject to change throughout the life span: some music preference dimensions may decrease with age (e.g., Intense, Contemporary), whereas some music preference dimensions may increase with age (e.g., Unpretentious, Sophisticated). [138] studied how user exposed to foreign culture might shift their movie genre preferences. It found that users might shift to more comprehensive movies when cultural literacy is required.

In this chapter, real-world datasets are used to examine if there is a preference shift in movie genre with regards to aging and how to quantify those shifts. This rest of this chapter is organized as following: In Section 5.2, how to visualize and quantify the preference shifts are illustrated. In Section 5.3, the real-word datasets are introduced and preference shifts among users on movie genres using those methods are shown. in Section 5.4, how to potentially use those shifts to generate dynamic recommendation lists with those user preference dynamics is discussed.

5.2 Method

Three steps are proposed to identify user preference dynamics: **1) sampling, 2) visualization, 3) quantification**. Sampling prepares the data in a time series manner to be used for visualization. Then the data is visualized and projected into a lower-dimensional space. At last, those shifts if any are quantified over the original space and compare it with the projection on the lower-dimensional space.

5.2.1 Sampling

The numbers of ratings are considered, not the scores of ratings. Unusually, rating data describes which **user** rates which **item** at what **timestamp** with what **score**. Here, assume that a user rating a movie indicates that the user is interested in this genre of movies. Therefore, the actual scores of the ratings are ignored in identifying user preference dynamics. For example, a user likes the movie ¡Star Wars¿ series very much. No matter how low the score of the rating he gives to one episode, he would still watch and rate the next episode, which means that the preferences of movie genres don't necessarily correlate with the shifts of scores of ratings.

The number of ratings are aggregated every once a while. Let's call the time intervals when ratings are sampled as **Sampling Window**, and the time intervals sampling are skipped as **Skipping Window** as shown in Fig. 5.1. The size of sampling windows and skipping windows determines how well the visualization and how much each shift is. The size of the sampling windows need be large enough to capture enough number of ratings to reflect the genre preferences, while the size of the skipping window need be large enough to distinguish a preference shift if any exists. If the sampling window size is too small, then a short-time period shift be misidentified as a long-time period shift. If the skipping window size is too small, then shifts may not be identified as differences are too subtle. The sizes of sampling and skipping windows depend on how many movies and how frequent each user rates. Ratings during the first time interval named **Warm Up Period** are ignored, since users may rate lots of movies they have watched before at once when they enter the system at the beginning. The timestamps associated with the rating during the warm up period doesn't necessarily align with the actual preference at that moment.

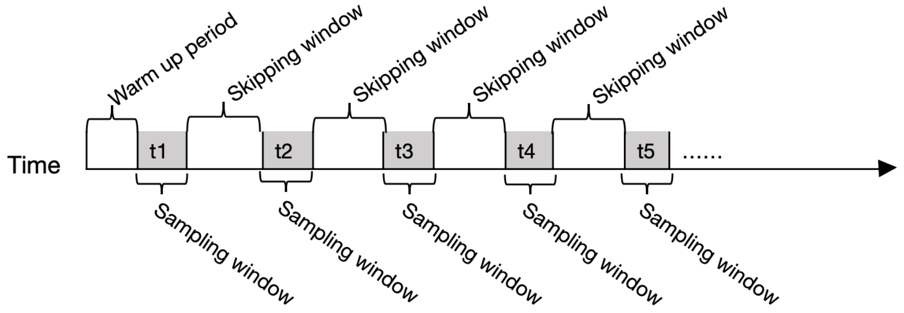


Figure 5.1.: Time sampling: the ratings are sampled during sampling windows once every skipping windows. The warm up period is also skipped as users may rate lots of movies they have seen before at once at the beginning. But the time he watched the movie may be a long time ago.

Next, the proportion of movies belonging to each genre during each sampling window for each user is calculated. During each sampling window t, the number of ratings of movies $|N_{i,g}^{t}|$ belonging to each genre g given by each user i with m total genres is counted. Then the percentage of movies belonging to each genre is calculated by dividing by the total number of all movies rated during this time window by this user: $P_{i,g}^t = \frac{|N_{i,g}^t|}{\sum_{c=1}^m |N_{i,c}^t|}$, where m is the total number of genres. Then $P_i^t = \{P_{i,1}^t, P_{i,2}^t ... P_{i,x}^t\}$ contains the elements of proportions of numbers of movies belonging to each genre from user i during sampling window t.

5.2.2 Visualization

The method to compare those multi-dimensional genre proportion vectors across different time $P_i = \{P_i^1, P_i^2, P_i^3, ... P_i^t\}$ from user i, and cross different users $P = \{P_i, P_j, ...\}$ is by using Multidimensional Scaling (MDS).each sampling P_i^t from each user is treated as one vector data entry in the matrix P and later group data points by users. MDS performs non-linear dimensionality reduction by taking the input of distance matrix between each object and places each object into a N-dimensional space such that the relative distance between each object are preserved as much as possible through optimization procedures. In this way, the preference shifts within each user and trend among all users across time can be projected to a 2D or 3D dimension that can interpreted. A useful loss function of MDS called stress, which is often minimized to ensure the distance between objects are preserved. [139] suggests that stress values below 15% represent a good fit.

5.2.3 Quantification

The quantification of user preference shifts can be done in the original data space or the reduced space after performing MDS. Since each P_i^t is sampled at the same rate as the skipping window size is constant. The distance between adjacent data sampling point P_i^t and P_i^{t+1} from the same user is actually how fast this user's preference shift. Let's call this **Shifting step size**. The average shifting step size $\Delta\bar{G}_i$ for user i can be measured using similarity measurement mentioned in Chapter 2.2.2. Taking the most common manhattan distance as an example, each shifting step size can be calculated as:

$$\Delta G_i^{t+1,t} = ||P_i^{t+1} - P_i^t||_1 = \sum_{g=1}^{m} |P_{i,g}^{t+1} - P_{i,g}^t|, \tag{5.1}$$

where m is the number of all genres.

Then the average shifting step size for user i is calculated as:

$$\bar{\Delta G_i} = \frac{\sum_{t=1}^{t_n-1} \Delta G_i^{t+1,t}}{t_n - 1}, \tag{5.2}$$

where t_n is the total number of sampling windows for user i.

Shifting step size can also be approximated after MDS visualization. How well it is approximated depends on the goodness-of-fit of MDS, which can be indicated by the loss function stress. Taking a 2D MDS to illustrate the process, each P_i^t will have a data point on the MDS plot with x and y coordinate on the plane. Let's call the coordinate x_i^t, y_i^t from user i at time t. As explained earlier, since each data point is sampled by the same skipping window, the distance between 2 data point on the MDS plot with the adjacent sampling time (e.g., t and $t+1$) from the same user is also approximated as how fast this user's preference shifts. Each shifting step size $\Delta G_i^{t+1,t}$ can be approximated using Manhattan distance or Euclidean distance. Taking Manhattan distance as an example:

$$\Delta G_i^{t+1,t} \approx |x_i^{t+1} - x_i^t| + |y_i^{t+1} - y_i^t|. \tag{5.3}$$

Let's say that if the user shifts his or her preference along the same direction as the previous shift, the user is predictable. Let's call this measurement as **stability** or **predictability** of a user. The way to measure how stable users are shifting their preferences along the same direction compared to previous shifts is by calculating the change of direction of any adjacent shift such as from P_i^t to P_i^t and from P_i^{t+1} to P_i^{t+2}. This is calculated only in the original space, not in reduced lower dimensional space since MDS preserves the relative distance between points, but not the angle between

2 vectors formed by those 3 points $P_i^t, P_i^{t+1}, P_i^{t+2}$. In a euclidean space, the similarity of direction between 2 previous adjacent shifts at time t for user i is calculated as:

$$\Delta\omega_i^t = \cos(\theta_i^t) = \frac{(P_i^t - P_i^{t-1}) \cdot (P_i^{t-1} - P_i^{t-2})}{||P_i^t - P_i^{t-1}|| \cdot ||P_i^{t-1} - P_i^{t-2}||}. \tag{5.4}$$

Then the angular distance $(1 - \Delta\omega_i^{t,t-2})$ is the angular difference between 2 shifts. The stability or predictability of user i, which is the average differences of shifts in direction of a user is aggregated as:

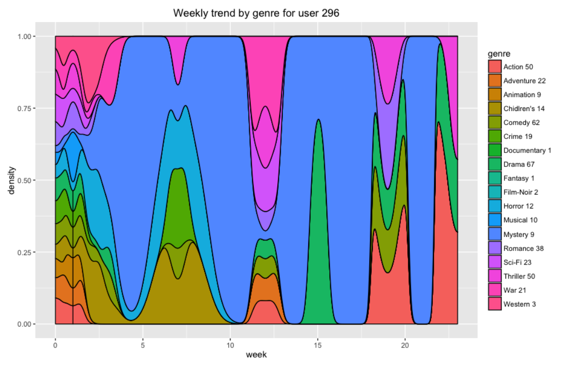
$$\Delta\bar{\omega}(i) = \frac{\sum_{t=3}^{t_n}(1 - \Delta\omega_i^t)}{t_n - 2}. \tag{5.5}$$

5.3 Experiment

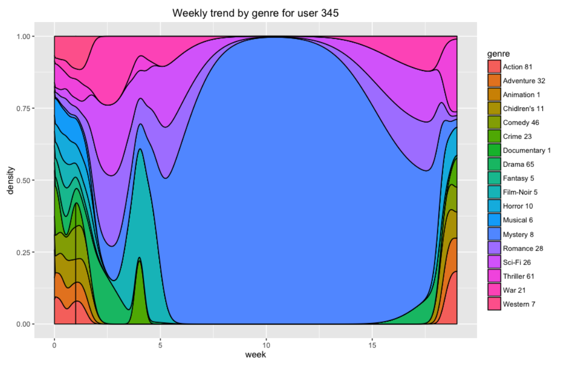
The most widely used MovieLens100K and MovieLens25m were used for our experiments. The time span of MovieLens100K is about half year, while MovieLens25m is about 25 years. The preliminary experiments were performed on the shorter time span MovieLens100K.

Evidence showing that users switch preferences among different genres by looking at the changes of propotions of number of movies belonging to different genres over the MovieLens100K dataset [115] is illustrated in Figure 5.2. Some users switches among different genres more frequently than others, while some users does not switch among different genres and stays in one genre.

Next the non-MDS with euclidean distance as distance matrix on MovieLens25m with longer time span was performed to examine how users' preferences shift in a long run. There are 18 genes in total. Each user was sampled for one year every 5 years starting at the 2nd year after each user joins MovieLens. Each user has to rate at least 20 movies during all sampling windows and has to be active for more than 20 years. Although the dataset contains 162,541 users, only 9 users are left above this threshold. The result is shown in 5.3. Dots on the plot are colored by users and connected sequentially with arrows pointing in chronological order. Some users shift



(a) Pattern of user watching movies belonging to different genres over time



(b) Pattern of user watching movies from the same genre over time

Figure 5.2.: User switching movie genres over time. X-axis is the time in week, week 0 is the date when the user started to rate in the database. Y-axis is the density of movie genres. Those are two users selected randomly from the MovieLens100K [115] dataset to show two different shifting patterns. In Figure 5.2a, at different time period, the user watched movies belonging to different genres, which indicates a user preference shift over time.

a lot such as user 4 with longer distance compared to others, while some users shift a little such as user 2.

The stress for 2D MDS is 12%, which indicates a good fit. The goodness-of-fit can also be checked by using Shepard diagram as shown in Fig. 5.4. The plot shows that most points are spread around the fitted function, which also indicates a good fit of the 2D MDS. The scree plot for up to 18 dimensions is shown in Fig. 5.5. 2D MDS gives us a good fit with stress under 15%.

The proportion of each genre over the 25 years for user 2 and user 4 was plotted to examine if that's really the case as shown in MDS plots in Fig. 5.6. The changes of the proportions of each genre for user 2 is not as dramatic as user 4. User 4 has gone into a different direction at year 15 and then go back to the same direction in year 20 as shown in Fig. 5.3. The proportion of genres in the lower bottom in violet colors goes down and then goes up, while the proportion of comedy in green color in the middle gets large in year 15 and then smaller in year 20. This may be the reason why the user shifts in a very different direction at year 15.

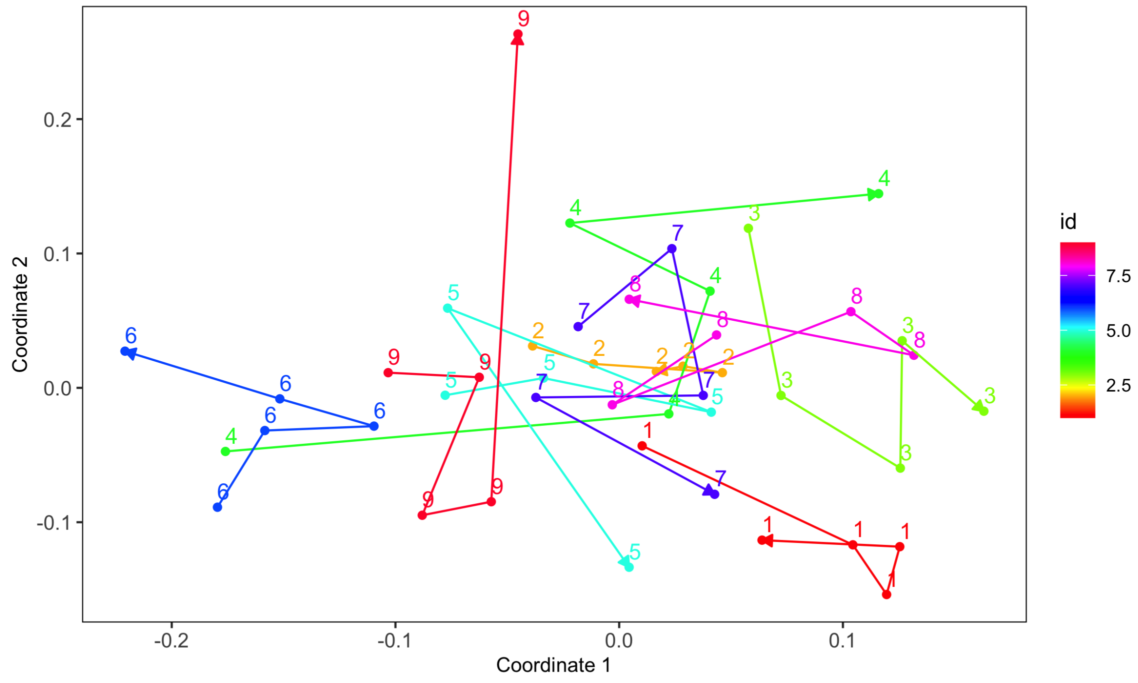


Figure 5.3.: MDS plot for user genre preference shifts: Each dot is colored, labeled by user from 1 to 9, connected sequentially with arrows pointing in chronological order. Users don't shift too much such as user 2, while some users shift a lot such as user 4 based on distances between dots.

The average shifting step size was calculated using euclidean distance in both original and reduced space (2D MDS) and the predictability of those 9 users in original space as shown in Table. 5.1. The users with the largest and smallest average step sizes are the same users calculated in the original space and those in the reduced space after performing MDS. Since the stress of 2D MDS is 12%, it is not expected to preserve the ranks of users with distance that are too close to each other. With more dimensions and lower stress for MDS, the ranks can be preserved better. For predictability, user 4 is ranked highest, though its average shifting step size is ranked lowest. There is a clear trend from Fig. 5.6b that this user is watching more action movies at a steady increasing rate. While the plot for user 2 doesn't show a clear pattern of the shifting trend, who is ranked 4 for predictability.

Table 5.1.: Average step size and predictability of each user: MDS successfully preserves the distance between points as the largest and least shifts are the same as calculated in original space and reduced space. User 4 shifts the most with the largest average shifting step size. But the shifting is the most predictable as the predictability of user 4 is ranked the highest.

	Average Shifting Step Size				Predictability	
User id	Original Space	Rank	Reduced Space	Rank	Original Space	Rank
1	0.229	7	0.122	3	1.198	8
2	0.129	1	0.065	1	1.159	6
3	0.235	8	0.175	6	0.986	4
4	0.379	9	0.245	9	0.813	1
5	0.214	5	0.232	8	0.925	2
6	0.201	4	0.118	2	0.956	3
7	0.190	3	0.171	4	1.263	9
8	0.165	2	0.174	5	1.138	5
9	0.228	6	0.218	7	1.172	7

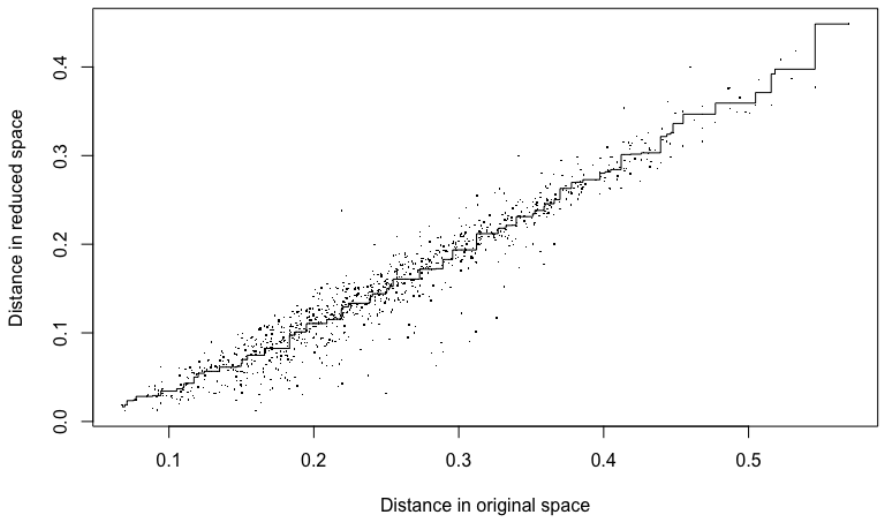


Figure 5.4.: Shepard plot for 2D MDS: most points are spread around the fitted function, which indicates a good fit of the 2D MDS

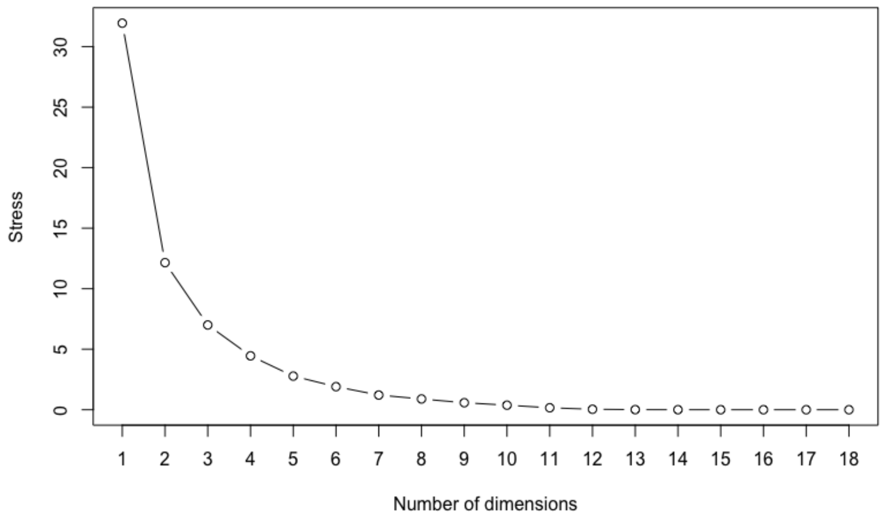
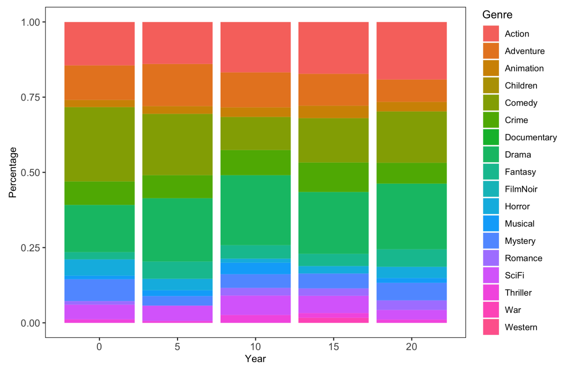
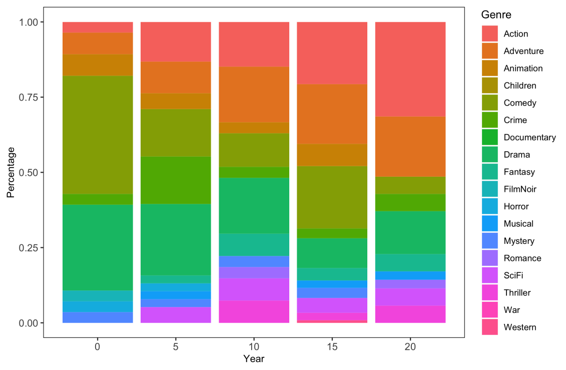


Figure 5.5.: Stress with different dimensions: 2D MDS gives us a good fit with stress under 15%



(a) User 2



(b) User 4

Figure 5.6.: User preference shifts comparison: the changes of the proportions of each genre for user 2 is not as dramatic as user 4. But user 4 is shifting at some pattern such as the proportion of action movies are watched more at a steady increasing rate.

5.4 Dynamic Recommendation List - Future Work

Now there is a way to identify whether there's a preference shift among each user. But how to utilize this information to generate dynamic recommendation list that best reflects the user's most recent preferences?

Let U be the items the user has interacted with (purchased, watched, etc.). Assuming some similarity measure between items, each item $j \in U$ is analyzed and N of the most similar items are found by summing the similarities of the candidate item and most similar k items from U [140]. To include user preference changes, a Category Index (CI) $\alpha_{g,u,t}$ is introduced on the item category g for user u at (current) time t, where $\alpha_{g,u,t}$ represents how each user interacted (e.g., rated, reviewed) with each category over time. $\alpha_{g,u,t}$ is calcualted as:

$$\begin{aligned}\alpha_{g,u,t} &= \frac{\text{Number of items belonging to category g at time t that user u has rated}}{\text{Total number of items at time t that user u has rated}} \\ &= \frac{\sum_{I_x \in I_{u,t}} [Category(I_x) = g]}{|I_{u,t}|},\end{aligned} \tag{5.6}$$

where I_x denotes the item, $I_{u,t}$ denotes the items rated by user u within the current time window t. The size of time window t can vary among different users as users may not switch genre preferences with the same frequency and rate. User historic data is used to find the best time window for each user.

By tracking whether the Category Index increases or decreases for each category at each time interval t, when a user switches his or her preferences from one category to another can be detected. This may be a switch that only happens in one session, which indicates that this user prefers some certain categories over others at that specific time. Then the recommendation list is adjusted to contain more items that belongs to that certain genre during that session time. It could also detect a trend as the user gradually switches his or her preferences towards other categories. The recommendation list will gradually contain fewer items belonging to the category that the user no longer prefers. This is achieved by letting $\alpha_{g,u,t}$ be the vector of all genre preferences of user u at the current time window. Instead of selecting the N most similar items for each item $j \in U$, $N \cdot \lceil \alpha_{g,u,t} \rceil$ of the most similar items for each item $j \in U$ are selected based on j's category. $\alpha_{g,u,t}$ can also be used as a weight on neighboring items by multiplying $\alpha_{g,u,t}$ to the similarities of candidate item and $j \in U$, before summing and ranking of similarities. In this way, either more items that are similar to current users' category preference are selected or more weights are assigned on the user opinions on the items that are similar to current users' category preference . In both ways, the recommendation list will be composed of more items that are related to the category that the user prefers at the current time.

5.4.1 Future Expected Setup and Expected Results

A number of validation experiments can be conducted. For example, randomly selecting a number of users with all their ratings from real-world dataset such as MovieLens, Netflix datasets and analyzing their preference dynamics to determine parameters (e.x., time window size t). Then, the Users Preference Dynamics(UPD)

[22] of recommendation lists of the current time window t and previous time window $t-1$ between our time-aware category-aware approach and the traditional approaches can be compared to determine how the improvements our algorithm can provide by considering user preference changes. The UPD is defined as:

$$UPD_u = 1 - \frac{I^u_{cur} \cap I^u_{prev}}{I^u_{cur} \cup I^u_{prev}} \tag{5.7}$$

where I^{cur}_u denotes the recommendation list generated for current time window for user u, I^{prev}_u denotes to the recommendation list generated for previous time window for user u. Lower UDP_u values indicated low dynamics of the recommendation list.

The recommendation list is expected to generate for those two time windows by our algorithm will vary more (higher UPD) compared to traditional approaches, if the user has indeed changed his or her preferences from time window $t-1$ to t. This means that at each time window, recommendations are made reflective of the user's most recent preferences.

Bibliography

[1] Christopher C Yang, Hsinchun Chen, and Kay Hong. Visualization of large category map for internet browsing. *Decision support systems*, 35(1):89–102, 2003.

[2] Martin Hilbert and Priscila López. The world's technological capacity to store, communicate, and compute information. *science*, page 1200970, 2011.

[3] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[4] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA, 2007.

[5] Aleksandar Ilic Maja Kabiljo. Recommending items to more than a billion people, 2015.

[6] Erik Brynjolfsson, Yu Hu, and Michael D Smith. Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. *Management Science*, 49(11):1580–1596, 2003.

[7] M Benjamin Dias, Dominique Locher, Ming Li, Wael El-Deredy, and Paulo JG Lisboa. The value of personalised recommender systems to e-business: a case study. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 291–294. ACM, 2008.

[8] Linyuan Lü, Matúš Medo, Chi Ho Yeung, Yi-Cheng Zhang, Zi-Ke Zhang, and Tao Zhou. Recommender systems. *Physics Reports*, 519(1):1–49, 2012.

[9] Yehuda Koren. How useful is a lower rmse?, 2007.

[10] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

[11] John U Farley, Jerrold Katz, and Donald R Lehmann. Impact of different comparison sets on evaluation of a new subcompact car brand. *Journal of Consumer Research*, 5(2):138–142, 1978.

[12] Harry S Upshaw. Own attitude as an anchor in equal-appearing intervals. *The Journal of Abnormal and Social Psychology*, 64(2):85, 1962.

[13] Monica Biernat, Melvin Manis, and Thomas E Nelson. Stereotypes and standards of judgment. *Journal of Personality and Social Psychology*, 60(4):485, 1991.

[14] Daniel Kahneman and Dale T Miller. Norm theory: Comparing reality to its alternatives. *Psychological review*, 93(2):136, 1986.

[15] Richard L Oliver. A cognitive model of the antecedents and consequences of satisfaction decisions. *Journal of marketing research*, pages 460–469, 1980.

[16] Robert B Woodruff, Ernest R Cadotte, and Roger L Jenkins. Modeling consumer satisfaction processes using experience-based norms. *Journal of marketing research*, pages 296–304, 1983.

[17] John G Lynch Jr, Dipankar Chakravarti, and Anusree Mitra. Contrast effects in consumer judgments: Changes in mental representations or in the anchoring of rating scales? *Journal of Consumer Research*, 18(3):284–297, 1991.

[18] Wendy W Moe and Michael Trusov. The value of social dynamics in online product ratings forums. *Journal of Marketing Research*, 48(3):444–456, 2011.

[19] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy.

In *Proceedings of the fifth ACM conference on Recommender systems*, pages 165–172. ACM, 2011.

[20] Xiaoying Zhang, Junzhou Zhao, and John Lui. Modeling the assimilation-contrast effects in online product rating systems: Debiasing and recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 98–106. ACM, 2017.

[21] Sangkil Moon, Paul K Bergey, and Dawn Iacobucci. Dynamic effects among movie ratings, movie revenues, and viewer satisfaction. *Journal of Marketing*, 74(1):108–121, 2010.

[22] Dimitrios Rafailidis and Alexandros Nanopoulos. Modeling users preference dynamics and side information in recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):782–792, 2016.

[23] Joshua L Moore, Shuo Chen, Douglas Turnbull, and Thorsten Joachims. Taste over time: The temporal dynamics of user preferences. In *ISMIR*, pages 401–406, 2013.

[24] Negar Hariri, Bamshad Mobasher, and Robin Burke. Adapting to user preference changes in interactive recommendation. In *IJCAI*, volume 15, pages 4268–4274, 2015.

[25] Mohsen Jamali, Gholamreza Haffari, and Martin Ester. Modeling the temporal dynamics of social rating networks using bidirectional effects of social relations and rating patterns. In *Proceedings of the 20th international conference on World wide web*, pages 527–536. ACM, 2011.

[26] Pedro G Campos, Alejandro Bellogín, Fernando Díez, and J Enrique Chavarriaga. Simple time-biased knn-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, pages 20–23. ACM, 2010.

[27] Neal Lathia, Stephen Hailes, and Licia Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 796–797. ACM, 2009.

[28] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 485–492. ACM, 2005.

[29] Yi Ding, Xue Li, and Maria E Orlowska. Recency-based collaborative filtering. In *Proceedings of the 17th Australasian Database Conference-Volume 49*, pages 99–107. Australian Computer Society, Inc., 2006.

[30] Pei Wu, Chi Ho Yeung, Weiping Liu, Cihang Jin, and Yi-Cheng Zhang. Time-aware collaborative filtering with the piecewise decay function. *arXiv preprint arXiv:1010.3988*, 2010.

[31] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[32] Jussi Karlgren. An algebra for recommendations. *SyslabWorking Paper*, 179, 1990.

[33] Jussi Karlgren. Newsgroup clustering based on user behavior-a recommendation algebra. *SICS Research Report*, 1994.

[34] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[35] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the*

SIGCHI conference on Human factors in computing systems, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[36] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[37] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[38] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

[39] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[40] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[41] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[42] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[43] Yi Ding. Effective and efficient collaborative filtering. 2011.

[44] Charu C Aggarwal et al. *Recommender systems*. Springer, 2016.

[45] Neil Rubens, Mehdi Elahi, Masashi Sugiyama, and Dain Kaplan. Active learning in recommender systems. In *Recommender systems handbook*, pages 809–846. Springer, 2015.

[46] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[47] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

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

[49] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[50] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[51] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[52] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[53] Xiaoyuan Su and Taghi M Khoshgoftaar. Collaborative filtering for multi-class data using belief nets algorithms. In *2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06)*, pages 497–504. IEEE, 2006.

[54] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Aaai/iaai*, pages 187–192, 2002.

[55] Xiaoyuan Su, Russell Greiner, Taghi M Khoshgoftaar, and Xingquan Zhu. Hybrid collaborative filtering algorithms using a mixture of experts. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 645–649. IEEE Computer Society, 2007.

[56] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 627–636. ACM, 2014.

[57] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

[58] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[59] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[60] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth*

conference on Uncertainty in artificial intelligence, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[61] Ming-Sheng Shang, Linyuan Lü, Wei Zeng, Yi-Cheng Zhang, and Tao Zhou. Relevance is more significant than correlation: Information filtering on sparse data. *EPL (Europhysics Letters)*, 88(6):68008, 2010.

[62] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[63] Gobinda G Chowdhury. *Introduction to modern information retrieval*. Facet publishing, 2010.

[64] Andrew Rodriguez, Byunghoon Kim, Mehmet Turkoz, Jae-Min Lee, Byoung-Youl Coh, and Myong K Jeong. New multi-stage similarity measure for calculation of pairwise patent similarity in a patent citation network. *Scientometrics*, 103(2):565–581, 2015.

[65] Paul Jaccard. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579, 1901.

[66] Hao Liao and An Zeng. Reconstructing propagation networks with temporal similarity. *Scientific reports*, 5:11404, 2015.

[67] Th A Sorensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biol. Skar.*, 5:1–34, 1948.

[68] Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

[69] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

[70] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

[71] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[72] Lada A Adamic and Eytan Adar. Friends and neighbors on the web. *Social networks*, 25(3):211–230, 2003.

[73] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[74] Jia-Yu Pan, Hyung-Jeong Yang, Christos Faloutsos, and Pinar Duygulu. Automatic multimedia cross-modal correlation discovery. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 653–658. ACM, 2004.

[75] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[76] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[77] Peter G Doyle and James Laurie Snell. *Random walks and electric networks*, volume 22. Mathematical Association of America Washington, DC, 1984.

[78] Agis Chartsias. *Link prediction in large scale social networks using hadoop.* PhD thesis, PhD thesis, Technical University of Crete, Greece, 2010.

[79] Weiping Liu and Linyuan Lü. Link prediction based on local random walk. *EPL (Europhysics Letters)*, 89(5):58007, 2010.

[80] Alexis Papadimitriou, Panagiotis Symeonidis, and Yannis Manolopoulos. Fast and accurate link prediction in social networking systems. *Journal of Systems and Software*, 85(9):2119–2132, 2012.

[81] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[82] Gabor Takacs, Istvan Pilaszy, Bottyan Nemeth, and Domonkos Tikk. On the gravity recommendation system. In *Proceedings of KDD cup and workshop*, volume 2007, 2007.

[83] Lyle Ungar and Dean P Foster. A formal statistical approach to collaborative filtering. *CONALD'98*, 1998.

[84] Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, pages 114–129, 1998.

[85] Manh Cuong Pham, Yiwei Cao, Ralf Klamma, and Matthias Jarke. A clustering approach for collaborative filtering recommendation using social network analysis. *J. UCS*, 17(4):583–604, 2011.

[86] Ludovico Boratto and Salvatore Carta. Using collaborative filtering to overcome the curse of dimensionality when clustering users in a group recommender system. In *ICEIS (2)*, pages 564–572, 2014.

[87] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[88] Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[89] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11(Jul):2057–2078, 2010.

[90] Raghunandan H Keshavan, Sewoong Oh, and Andrea Montanari. Matrix completion from a few entries. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, pages 324–328. IEEE, 2009.

[91] Raghunandan H Keshavan and Sewoong Oh. A gradient descent algorithm on the grassman manifold for matrix completion. *arXiv preprint arXiv:0910.5260*, 2009.

[92] Zhimin Chen, Yi Jiang, and Yao Zhao. A collaborative filtering recommendation algorithm based on user interest change and trust evaluation. *JDCTA*, 4(9):106–113, 2010.

[93] Paula Cristina Vaz, Ricardo Ribeiro, and David Martins De Matos. Understanding the temporal dynamics of recommendations across different rating scales. In *UMAP Workshops*, 2013.

[94] Nathan N Liu, Min Zhao, Evan Xiang, and Qiang Yang. Online evolutionary collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 95–102. ACM, 2010.

[95] Wenjun Jiang, Jie Wu, Guojun Wang, and Huanyang Zheng. Fluidrating: A time-evolving rating scheme in trust-based recommendation systems using fluid dynamics. In *INFOCOM, 2014 Proceedings IEEE*, pages 1707–1715. IEEE, 2014.

[96] Kristina Lerman. Dynamics of collaborative document rating systems. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 46–55. ACM, 2007.

[97] Neal Kiritkumar Lathia. *Evaluating collaborative filtering over time*. PhD thesis, UCL (University College London), 2010.

[98] Ivan Koychev and Ingo Schwab. Adaptation to drifting user's interests. In *Proceedings of ECML2000 Workshop: Machine Learning in New Information Age*, pages 39–46, 2000.

[99] Linas Baltrunas and Xavier Amatriain. Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems (CARS'09)*, 2009.

[100] Gavin Potter. Putting the collaborator back into collaborative filtering. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, page 3. ACM, 2008.

[101] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM, 2010.

[102] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. *arXiv preprint arXiv:2001.10167*, 2020.

[103] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[104] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, pages 165–174, 2019.

[105] Xiao Wang, Ruijia Wang, Chuan Shi, Guojie Song, and Qingyong Li. Multi-component graph convolutional collaborative filtering. *arXiv preprint arXiv:1911.10699*, 2019.

[106] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[107] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[108] TN Kipf and M Welling. Semi-supervised classification with graph convolutional networks iclr. 2017.

[109] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[110] Daniel Billsus and Michael J Pazzani. Learning collaborative information filters. In *Icml*, volume 98, pages 46–54, 1998.

[111] Gordon Hughes. On the mean accuracy of statistical pattern recognizers. *IEEE transactions on information theory*, 14(1):55–63, 1968.

[112] Alexandros Nanopoulos, Miloš Radovanović, and Mirjana Ivanović. How does high dimensionality affect collaborative filtering? In *Proceedings of the third ACM conference on Recommender systems*, pages 293–296. ACM, 2009.

[113] Dominik Schnitzer, Arthur Flexer, Markus Schedl, and Gerhard Widmer. Local and global scaling reduce hubs in space. *Journal of Machine Learning Research*, 13(Oct):2871–2902, 2012.

[114] Peter Knees, Dominik Schnitzer, and Arthur Flexer. Improving neighborhood-based collaborative filtering by reducing hubness. In *Proceedings of International Conference on Multimedia Retrieval*, page 161. ACM, 2014.

[115] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

[116] Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

[117] Cort J Willmott, Kenji Matsuura, and Scott M Robeson. Ambiguities inherent in sums-of-squares-based error statistics. *Atmospheric Environment*, 43(3):749–752, 2009.

[118] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research*, 11(Sep):2487–2531, 2010.

[119] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 305–308. ACM, 2011.

[120] Martin Piotte and Martin Chabbert. The pragmatic theory solution to the netflix grand prize. *Netflix prize documentation*, 2009.

[121] Sergiu Gordea and Markus Zanker. Time filtering for better recommendations with small and sparse rating matrices. In *International Conference on Web Information Systems Engineering*, pages 171–183. Springer, 2007.

[122] Tom M Mitchell, Rich Caruana, Dayne Freitag, John McDermott, David Zabowski, et al. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):80–91, 1994.

[123] I Barry Crabtree and Stuart J Soltysiak. Identifying and tracking changing interests. *International Journal on Digital Libraries*, 2(1):38–53, 1998.

[124] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.

[125] Marcus A Maloof and Ryszard S Michalski. Learning evolving concepts using partial memory approach. 1995.

[126] Daniel Billsus and Michael J Pazzani. A hybrid user model for news story classification. In *UM99 User Modeling*, pages 99–108. Springer, 1999.

[127] Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 43–52. IEEE, 2007.

[128] Dongjoo Lee, Sung Eun Park, Minsuk Kahng, Sangkeun Lee, and Sang-goo Lee. Exploiting contextual information from event logs for personalized recommendation. In *Computer and Information Science 2010*, pages 121–139. Springer, 2010.

[129] Alexandros Karatzoglou. Collaborative temporal order modeling. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 313–316. ACM, 2011.

[130] Fatemeh Rezaeimehr, Parham Moradi, Sajad Ahmadian, Nooruldeen Nasih Qader, and Mahdi Jalili. Tcars: Time-and community-aware recommendation system. *Future Generation Computer Systems*, 78:419–429, 2018.

[131] Dimitrios Rafailidis and Alexandros Nanopoulos. Repeat consumption recommendation based on users preference dynamics and side information. In *Proceedings of the 24th International Conference on World Wide Web*, pages 99–100, 2015.

[132] Christoph Hermann. Time-based recommendations for lecture materials. In *2010 World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 1028–1033, 2010.

[133] Wang Dawei, Ventresca Mario, and Yuehwern Yih. Improving neighbor-based collaborative filtering by using a hybrid similarity measurement. *Expert Systems with Applications*, 2020.

[134] Stefan Gebhardt and Richard von Georgi. The change of music preferences following the onset of a mental disorder. *Mental illness*, 7(1), 2015.

[135] Frederick S Barrett, Kevin J Grimm, Richard W Robins, Tim Wildschut, Constantine Sedikides, and Petr Janata. Music-evoked nostalgia: Affect, memory, and personality. *Emotion*, 10(3):390, 2010.

[136] Robert D Crowther and Kevin Durkin. Sex-and age-related differences in the musical behaviour, interests and attitudes towards music of 232 secondary school students. *Educational Studies*, 8(2):131–139, 1982.

[137] Arielle Bonneville-Roussy, Peter J Rentfrow, Man K Xu, and Jeff Potter. Music through the ages: Trends in musical engagement and preferences from adolescence through middle adulthood. *Journal of personality and social psychology*, 105(4):703, 2013.

[138] Saham Barza and Mehran Memari. Movie genre preference and culture. *Procedia-Social and Behavioral Sciences*, 98:363–368, 2014.

[139] Pat Dugard, John B Todman, and Harry Staines. *Approaching multivariate analysis: A practical introduction.* Routledge, 2010.

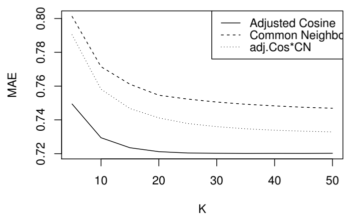
[140] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254. ACM, 2001.

APPENDICES

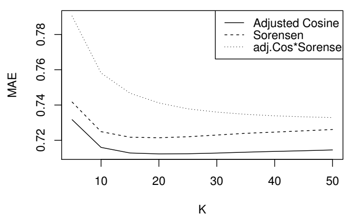
APPENDIX A
AMPLIFICATION PARAMETER

Figure A.1.: MAE of adjusted cosine combined with different structural similarity measurements with $\alpha = 1$ over ML-100K dataset. The performance of the combination of adjusted cosine and structural similarity measurement works better than using adjusted cosine or structural similarity measurement alone.

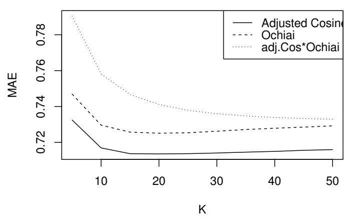
(a) Common Neighbor



(b) Sorensen



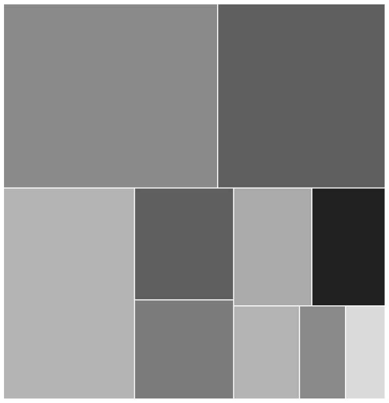
(c) Ochiai



(a) CASE 2: Small α=1 makes structural similarity measurement lose dominance over the ranking process by choosing a small α not to enlarge the differences calculated by structural similarity measurement big enough so that the list can be re-ranked by rating-based similarity measurement. Note that every item is re-ranked.

Ranking	Item ID	S_{struct}	S_{rating}	New Sim
1	3	0.8	0.8	$0.8^1 \times 0.8$=0.64
2	1	1.0	0.5	$1^1 \times 0.5$=0.50
3	2	0.9	0.5	$0.9^1 \times 0.5$=0.45
4	8	0.3	0.6	$0.3^1 \times 0.6$=0.18
5	7	0.4	0.4	$0.4^1 \times 0.4$=0.16
6	6	0.5	0.3	$0.5^1 \times 0.3$=0.15
7	4	0.7	0.2	$0.7^1 \times 0.2$=0.14
8	9	0.2	0.5	$0.2^1 \times 0.5$=0.10
9	10	0.1	0.7	$0.1^1 \times 0.7$=0.07
10	5	0.6	0.1	$0.6^1 \times 0.1$=0.06

(b) CASE 2: Percentage of each neighbor items when α=1: although the list is totally re-ranked by rating-based similarities, the percentage of each neighbor similarity changes a little comparing to the percentage before multiplying rating-based similarities.



(a) CASE 1: Large α=10 makes structural similarity measurement dominant the ranking process by choosing a large α to enlarge the differences calculated by structural similarity measurement so large that rating-based similarity measurement can hardly re-rank.

Ranking	Item ID	S_{struct}	S_{rating}	New Sim
1	1	1.0	0.5	$1^{10} \times 0.5$=5.00e-1
2	2	0.9	0.5	$0.9^{10} \times 0.5$=1.74e-1
3	3	0.8	0.8	$0.8^{10} \times 0.8$=8.58e-2
4	4	0.7	0.2	$0.7^{10} \times 0.2$=5.64e-3
5	5	0.6	0.1	$0.6^{10} \times 0.1$=6.04e-4
6	6	0.5	0.3	$0.5^{10} \times 0.3$=2.92e-4
7	7	0.4	0.4	$0.4^{10} \times 0.4$=4.19e-5
8	8	0.3	0.6	$0.3^{10} \times 0.6$=3.54e-6
9	9	0.2	0.5	$0.2^{10} \times 0.5$=5.12e-8
10	10	0.1	0.7	$0.1^{10} \times 0.7$=7e-11

(b) CASE 1: Percentage of each neighbor items when α=10: new similarity of the first item is more than that of the rest items combined. The prediction would be biased towards to the user opinion of the first item.

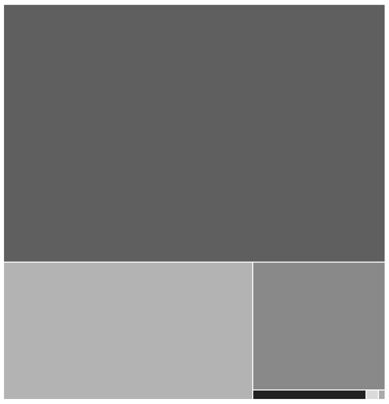


Table A.3.: Choice of amplification parameter α. CASE 1: If α is too large, prediction is biased towards the opinions of the first few neighbor items. CASE 2: If α is too small, prediction can be biased towards very few users since the ranking can be determined by rating-based similarity measurement which could be calculated based on very few users. CASE 3: Ranking is determined mainly by structural similarity while allowing rating-based similarity to adjust on a smaller scale.

(a) CASE 3: α=1.5 makes structural similarity measurement dominant the ranking process by enlarging the differences calculated by structural similarity measurement large enough but also small enough to let rating-based similarity measurement re-rank the list on a smaller scale. Note that only item 3 and item 5 change their ranking.

Ranking	Item ID	S_{struct}	S_{rating}	New Sim
1	3	0.8	0.8	$0.8^{1.5} \times 0.8$=0.5724
2	1	1.0	0.5	$1^{1.5} \times 0.5$=0.5000
3	2	0.9	0.5	$0.9^{1.5} \times 0.5$=0.4259
4	4	0.7	0.2	$0.7^{1.5} \times 0.2$=0.1171
5	6	0.5	0.3	$0.5^{1.5} \times 0.3$=0.1060
6	7	0.4	0.4	$0.4^{1.5} \times 0.4$=0.1011
7	8	0.3	0.6	$0.3^{1.5} \times 0.6$=0.0985
8	5	0.6	0.1	$0.6^{1.5} \times 0.1$=0.0464
9	9	0.2	0.5	$0.2^{1.5} \times 0.5$=0.0447
10	10	0.1	0.7	$0.1^{1.5} \times 0.7$=0.0221

(b) CASE 3: Percentage of each neighbor items when α=1.5: the list is re-ranked a little bit by rating-based similarities, the percentage of each neighbor similarity changes a little comparing to the percentage before multiplying rating-based similarities.

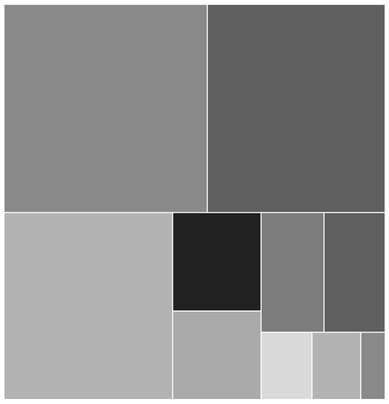


Table A.4.: MAE v.s. RMSE with increasing error variance. MAE stays the same while RMSE increases as the frequency distribution of magnitudes of error variance increases. Large errors will result a higher RMSE.

(a) CASE 1: Evenly distributed errors

ID	$Error$	$\|Error\|$	$Error^2$
1	2	2	4
2	2	2	4
3	2	2	4
4	2	2	4
5	2	2	4
6	2	2	4
7	2	2	4
8	2	2	4
9	2	2	4
10	2	2	4
		MAE:	2.000
		RMSE:	2.000

(b) CASE 2: Small variance in errors

ID	$Error$	$\|Error\|$	$Error^2$
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9
		MAE:	2.000
		RMSE:	2.236

(c) CASE 3: Large error outlier

ID	$Error$	$\|Error\|$	$Error^2$
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400
		MAE:	2.000
		RMSE:	6.325

APPENDIX B
MAE V.S. RMSE

Table B.1.: MAE v.s. RMSE with different variance of errors. RMSE stays the same while MAE differs with error variance of case 4 is greater than that of case 5.

(a) CASE 4: Errors = 0 or 5

ID	$Error$	$\lvert Error \rvert$	$Error^2$
1	5	5	25
2	5	5	25
3	5	5	25
4	5	5	25
5	5	5	25
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
		MAE:	2.500
		RMSE:	3.536

(b) CASE 5: Errors = 3 or 4

ID	$Error$	$\lvert Error \rvert$	$Error^2$
1	3	3	9
2	3	3	9
3	3	3	9
4	3	3	9
5	3	3	9
6	4	4	16
7	4	4	16
8	4	4	16
9	4	4	16
10	4	4	16
		MAE:	3.500
		RMSE:	3.536