Expert Systems with Applications 160 (2020) 113651

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Improving neighbor-based collaborative filtering by using a hybrid similarity measurement

Dawei Wang, Yuehwern Yih, Mario Ventresca *

School of Industrial Engineering, Purdue University, 315 Grant St, West Lafayette, IN 47906, USA

ARTICLE INFO

Article history:
Received 14 September 2019
Revised 24 May 2020
Accepted 8 June 2020
Available online 6 July 2020

Keywords:
Collaborative filtering
K-nearest-neighbor
Recommendation system
Item-based
Memory-based
Similarity measurement

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 paper, 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.

1. 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 (Linden et al., 2003), online movie streaming companies such as Netflix (Bennett et al., 2007), and social media networks such as Facebook (Maja Kabiljo, 2015). 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% and 40% of sales on Amazon are due to recommended products that do not belong to the shop's 100,000 most sold products (Brynjolfsson et al., 2003), 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 (Dias et al., 2008). Hence, a recommendation system can significantly impact a company's revenue (Lü et al., 2012). 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 (Koren, 2007).

Recommendation systems typically generate a list of recommendations to users in one of three ways (Su and Khoshgoftaar, 2009) – (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 ratings, while model-based approaches

* Corresponding author.
E-mail addresses: wang337@purdue.edu (D. Wang), yih@purdue.edu (Y. Yih), mventresca@purdue.edu (M. Ventresca).

[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).

https://doi.org/10.1016/j.eswa.2020.113651
0957-4174/

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 1.1.

This paper takes the approach of item-based and memory-based collaborative filtering due to its simplicity, efficiency, and ability to produce accurate recommendations (Desrosiers and Karypis, 2011). 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 2). 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 (Herlocker et al., 1999) and Netflix datasets (Bennett et al., 2007). 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.

1.1. 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 (Adomavicius and Tuzhilin, 2005; Su and Khoshgoftaar, 2009; Lü et al., 2012):

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) (Hofmann, 1999; Hofmann, 2004), LDA (Latent Dirichlet Allocation) (Blei et al., 2003), 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 (Lü et al., 2012). Content-based filterings are widely used in a variety of domains ranging from recommending webpages, news articles, restaurants, etc (Pazzani and Billsus, 2007). For examples: Pandora Radio[2] recommends users with songs that share similar characteristics (Casey et al., 2008), 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 (Pazzani and Billsus, 2007).
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 Aggarwal et al. (2016). 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 do not 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 have not rated a sufficient number of items, these methods may not perform very well (known as the cold start problem (Rubens et al., 2015; Schein et al., 2002)). Collaborative Filtering could also suffer from scalability or sparsity issues (Sarwar et al., 2001; details are listed in Section 1.2).
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:
   (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) (Goldberg et al., 1992; Sarwar et al., 2001). 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 (Desrosiers and Karypis, 2011), 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 (Lü et al., 2012).
   (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) (Koren, 2008), and Principal component analysis (PCA) (Tipping and Bishop, 1999), 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 (Su and Khoshgoftaar, 2009). 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 (Su and Khoshgoftaar, 2009).
3. Hybrid. This strategy is to combine two or more techniques (Burke, 2002; Su and Khoshgoftaar, 2006; Melville et al., 2002; Su et al., 2007), or with other techniques such as deep learning (Wang and Wang, 2014) or clustering (Xue et al., 2005) 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 (Su and Khoshgoftaar, 2009).

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

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 (Chen et al., 2020) A general framework of Neural network-based Collaborative Filtering (NCF) is proposed to learn user-item interaction via a multi-layer perceptron (He et al., 2017). Neural Graph Collaborative Filtering (NGCF) is proposed embedding propagation layer to leverage high-order connectivities in user-item integration graph of model-based CF (Wang et al., 2019). Multi-Component graph convolutional Collaborative Filtering (MCCF) approach is proposed to distinguish the latent purchasing motivations (Wang et al., 2019). 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 (Ying et al., 2018); Simple Graph Convolution (SGC) (Wu et al., 2019) and Linear Residual Graph Convolutional Collaborative Filtering (LR-GCCF) (Chen et al., 2020) removes the non-linearities. Although most GCN based approaches including GCN based recommendation models achieve the best performance with two layers (Kipf and Welling, 2017; Hamilton et al., 2017; Ying et al., 2018). It still requires training the model with all user-item interaction information iteratively as model-based approaches.

1.2. 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.

1.2.1. Sparsity

In practice, recommendation systems are used with very large datasets such as those from Amazon (Linden et al., 2003), Facebook (Maja Kabiljo, 2015) or Netflix (Bennett et al., 2007). 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 (Rubens et al., 2015; Schein et al., 2002). 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.

1.2.2. 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 (Linden et al., 2003), and memory-based methods can react to new ratings and make a prediction in real time even for extremely large datasets (Sarwar et al., 2001; Linden et al., 2003). While model-based approaches can mitigate scalability problems by using dimensionality reduction techniques such as SVD (Billsus and Pazzani, 1998), 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 (Su and Khoshgoftaar, 2009).

1.2.3. 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 (Hughes, 1968). Two outcomes may result (Nanopoulos et al., 2009).

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 (Nanopoulos et al., 2009). Hubness starts reducing only when intrinsic dimensionality is reached, where further reduction may incur loss of information (Nanopoulos et al., 2009). The concentration and hubness are inherent properties of high dimensionality, not proprieties like sparsity or skewness of the distribution of ratings (Nanopoulos et al., 2009). Both phenomena can negatively affect the accuracy of predictions since they impact the representativeness of nearest neighbor lists (Nanopoulos et al., 2009). 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 (Schnitzer et al., 2012; Knees et al., 2014).

1.3. Summary of main contributions

In this paper, 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 3). We study how hubness appears in nearest neighbor list using rating-based and structural similarity measurement alone. The main contributions of this paper 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 3).
2. We compare two benchmarks for prediction accuracy evaluation: MAE (Mean Absolute Error) and RMSE (Root Mean Squared Error) (Section 4.2) and and provide experimental

results on three most widely referenced datasets: MovieLens100K, MovieLens1M, and Netflix Challenge datasets (Section 4).
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 100 K, 1 M and Netflix datasets (Section 5.1); (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 1 M dataset (Section 5.2) and (3) Our method can achieve 3% lower MAE and RMSE compared to traditional memory-based CFs on non-cold start users on MovieLens 100 K dataset (Section 5.3).

2. Background

As briefly mentioned in Section 1.1, 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 2.2 and 2.3. Before that, let's define some terminologies and notations in Section 2.1.

2.1. Notation

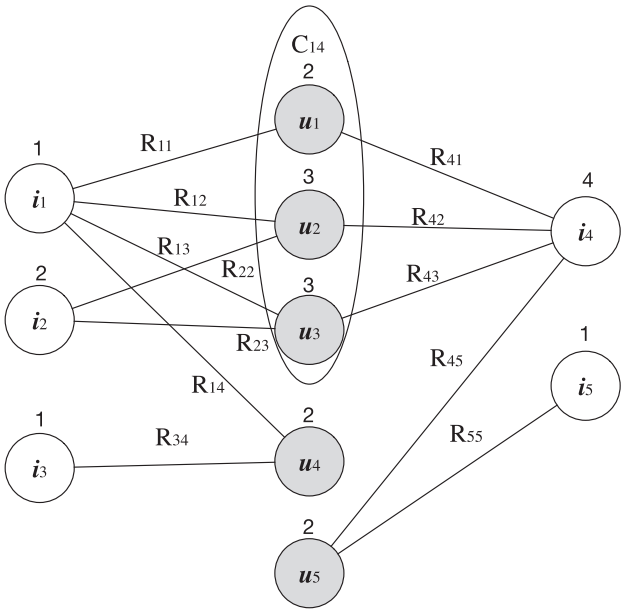
$$\mathbf{R} = \begin{array}{c} \begin{matrix} \boldsymbol{u_1} & \boldsymbol{u_2} & \boldsymbol{u_3} & \dots & \boldsymbol{u_m} \end{matrix} \\ \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} & \mathbf{R}_{13} & \dots & \mathbf{R}_{1m} & \boldsymbol{i_1} \\ \mathbf{R}_{21} & \mathbf{R}_{22} & \mathbf{R}_{23} & \dots & \mathbf{R}_{2m} & \boldsymbol{i_2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{R}_{n1} & \mathbf{R}_{n2} & \mathbf{R}_{n3} & \dots & \mathbf{R}_{nm} & \boldsymbol{i_n} \end{bmatrix} \end{array} \quad (1)$$

Given a dataset containing ratings from m users on n items, we can get an item-user matrix $\mathbf{R}$ as shown in (1). For example, a dataset is given as:

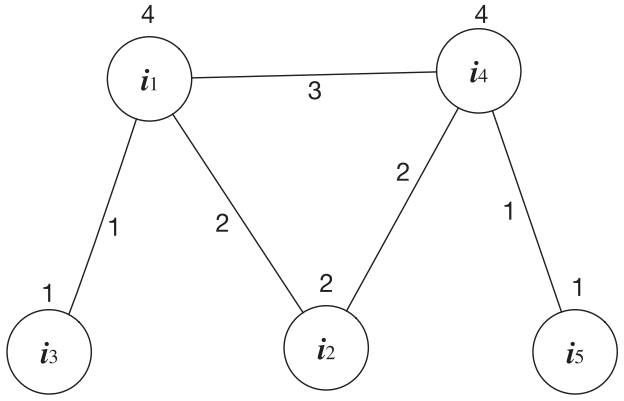
$$\mathbf{R} = \begin{array}{c} \begin{matrix} \boldsymbol{u_1} & \boldsymbol{u_2} & \boldsymbol{u_3} & \boldsymbol{u_4} & \boldsymbol{u_5} \end{matrix} \\ \begin{bmatrix} 4 & 2 & 4 & 1 & 0 & \boldsymbol{i_1} \\ 0 & 4 & 2 & 0 & 0 & \boldsymbol{i_2} \\ 0 & 0 & 0 & 3 & 0 & \boldsymbol{i_3} \\ 5 & 3 & 5 & 0 & 4 & \boldsymbol{i_4} \\ 0 & 0 & 0 & 0 & 5 & \boldsymbol{i_5} \end{bmatrix} \end{array} \quad (2)$$

Then we can draw the item-user network as in Fig. 1a and draw item-based network as in Fig. 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.

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 (Linden et al., 2003). (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 (Herlocker et al., 1999). 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 do not 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



(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.

Fig. 1. Illustration of item-user network and Item-based network: From item-user network as in (a), we draw the item-based network as in (b).

pre-calculated, which could shorten the time needed to make a prediction (Lü et al., 2012).

2.2. 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 (Lü et al., 2012). 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 (Lü et al., 2012).

2.2.1. Rating-based similarity measurement

Traditional methods only use ratings to compute similarity between items (e.g. Pearson correlation coefficient (Resnick et al., 1994), cosine and adjusted cosine (Sarwar et al., 2001)) by only considering ratings from co-rated users C_{ij}. Taking adjusted cosine similarity measurement as an example, it is calculated as:

$$S_{rating}(\boldsymbol{i_i}, \boldsymbol{i_c}) = \frac{\sum_{x \in C_{ic}} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{u_x})(\mathbf{R}_{cx} - \bar{\mathbf{R}}_{u_x})}{\sqrt{\sum_{x \in Cic} (\mathbf{R}_{ix} - \bar{\mathbf{R}}_{u_x})^2} \sqrt{\sum_{x \in C_{ic}} (\mathbf{R}_{cx} - \bar{\mathbf{R}}_{u_x})^2}}. \quad (3)$$

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 (Breese et al., 1998). To compensate, the idea of inverse user frequency was introduced by multiplying $log(\frac{|\boldsymbol{i_i}|}{m})$ to the original item similarity measurement (Breese et al., 1998). Note that if $|\boldsymbol{i_i}| = \boldsymbol{m}$, 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 (Koren, 2008):

$$simShrunk(\boldsymbol{i_i}, \boldsymbol{i_j}) = \frac{|\boldsymbol{C_{ij}}|}{|\boldsymbol{C_{ij}}| + \lambda} \boldsymbol{S}(\boldsymbol{i_i}, \boldsymbol{i_j}). \quad (4)$$

  Where a typical λ is suggested to be 100 in Koren (2008). However, (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 1.2.3, 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 Hughes (1968). 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 (Nanopoulos et al., 2009).
- **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}.

2.2.2. 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 (Knees et al., 2014).

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 2.2. Studies have tried to combine rating-based similarity measurement with structural information to form a better similarity measurement:

Similarities have been combined or modified to improve prediction accuracy. As illustrated in Section 2.2, Koren (2008) used a fixed variable to penalize similarities that are calculated based on a very few users as illustrated in Eq. (4) without considering the local information such as the upper bound of the number of co-rated users $\min(|\boldsymbol{i_i}|, |\boldsymbol{i_j}|)$. Breese et al. (1998) 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. To avoid this drawback, Bobadilla et al. (2010) combined structural similarity measurement Jaccard and mean square differences (JMSD), in which Jaccard is used to capture the proportion of the co-rated users and MSD is used to capture the information of ratings. The equation of JMSD is illustrated in Eq. 5.

$$sim^{JMSD}(\boldsymbol{i_i}, \boldsymbol{i_j}) = \boldsymbol{sim^{Jaccard}}(\boldsymbol{i_i}, \boldsymbol{i_j}) \cdot \boldsymbol{sim^{MSD}}(\boldsymbol{i_i}, \boldsymbol{i_j}). \quad (5)$$

$$sim^{Jaccard}(\boldsymbol{i_i}, \boldsymbol{i_j}) = \frac{|\boldsymbol{C_{ic}}|}{|\boldsymbol{i_i} \cup \boldsymbol{i_c}|}. \tag{6}$$

$$sim^{MSD}(\boldsymbol{i_i}, \boldsymbol{i_j}) = \boldsymbol{1} - \frac{\sum_{\boldsymbol{x} \in \boldsymbol{C_{ij}}} (\mathbf{R_{ix}} - \mathbf{R_{jx}})^{\boldsymbol{2}}}{|\boldsymbol{C_{ij}}|}. \tag{7}$$

The combination of Jaccard and MSD can make up for the partial shortages of Jaccard and MSD, in which the similarities becomes diverse. But there are still similarities between different pairs of items or users that are the same as illustrated in the example in Liu et al. (2014). Another approach introduced triangle similarity and combined with Jaccard similarity (Sun et al., 2017). Mu et al. (2019) improved Common Pearson Correlation Coefficient(COPC) by combining with global similarity Hellinger Distance and Jaccard similarity.

Some researches have combined similarities to alleviate the sparse data or cold start problem. For example, Bobadilla et al. (2012) combined 6 different similarity measurements and assigned weights learned through neural network learning to obtain a global similarity. However, this approach introduced more computational and time complexity with neural network learning process. Ahn (2008) proposed three factors of similarity: proximity, impact and popularity called PIP. But there are disadvantages (Liu et al., 2014): 1. does not consider the absolute ratings and the proportion of the co-rated items; 2. does not consider each user's global rating preference; 3. is not normalized and not convenient to combine with other methods. Liu et al. (2014) improved the PIP model and proposed three new factors: proximity, significance, and singularity and combined with modified Jaccard and user rating preference. Patra et al. (2015) proposed a metric that combined Jaccard and Bhattacharyya coefficient to overcome the sparse data problem by extracting global information when there are few or no co-rated items. Polatidis and Georgiadis (2016) also proposed a multi-level measurement by assigning similarities between users based on different cases depending on Pearson correlation coefficient and the number of co-rated items to provide recommendations to users that do not have at least a number of co-rated items and a certain Pearson Correlation Coefficient similarity value. Another approach proposed a new similarity measurement by combining three impact factors to overcome the data sparsity problem (Feng et al., 2018).

There are also some studies that tried to reduce the computational complexity of computing pairwise similarities. For example, Zhang et al. (2016) proposed a novel data structure to compute Pearson correlation similarities in linear time.

2.3. 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

2.3.1. Simple weighted average

The simplest way to predict the rating from user u on item i using simple weighted average is Sarwar et al. (2001):

$$\hat{\mathbf{R}}_{iu} = \frac{\sum_{c=1}^{K} (\mathbf{R}_{cu}) \cdot S_{rating}(\boldsymbol{i_i}, \boldsymbol{i_c})}{\sum_{c=1}^{K} S_{rating}(\boldsymbol{i_i}, \boldsymbol{i_c})}. \tag{8}$$

2.3.2. Weighted sum of others' ratings

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

$$\hat{\mathbf{R}}_{iu} = \bar{\boldsymbol{i}}_i + \frac{\sum_{c=1}^{K} (\mathbf{R}_{cu} - \bar{\boldsymbol{i}}_c) \cdot S_{rating}(\boldsymbol{i_i}, \boldsymbol{i_c})}{\sum_{c=1}^{K} S_{rating}(\boldsymbol{i_i}, \boldsymbol{i_c})}. \tag{9}$$

3. 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} (see Table 1).

In Eq. 10, 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:

$$S_{combined}(\boldsymbol{i_i}, \boldsymbol{i_c}) = \boldsymbol{S_{struct}}(\boldsymbol{i_i}, \boldsymbol{i_c})^{\alpha} \cdot \boldsymbol{S_{rating}}(\boldsymbol{i_i}, \boldsymbol{i_c}). \tag{10}$$

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

3.1. 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 (Sarwar et al., 2001). A list of some structural similarity measurement is in Table 2. The structural similarity measurement we choose fulfills the following 2 requirements.

3.1.1. 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

Table 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, $\boldsymbol{u_j} = \mathbf{R}_{*j}, \forall * \in [\boldsymbol{1}..\boldsymbol{n}]$
$\lvert\boldsymbol{u_j}\rvert$	Total number of items that user j has rated, $\lvert\boldsymbol{u_j}\rvert \in [\boldsymbol{0}..\boldsymbol{n}]$
$\boldsymbol{i_i}$	Vector of ratings for item i, $\boldsymbol{i_i} = \mathbf{R}_{i*}, \forall * \in [\boldsymbol{1}..\boldsymbol{m}]$
$\lvert\boldsymbol{i_i}\rvert$	Total number of users who has rated item i, $\lvert\boldsymbol{i_i}\rvert \in [\boldsymbol{0}..\boldsymbol{m}]$
$\bar{\boldsymbol{u}}_j$	$\bar{\boldsymbol{u}}_j = \frac{\sum_{x=1}^{n} \mathbf{R}_{xj}}{\lvert\boldsymbol{u_j}\rvert}$, 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, $\boldsymbol{i_i} \cap \boldsymbol{i_j} \in [\boldsymbol{0}..\boldsymbol{m}]$
$S_{rating}(\boldsymbol{i_i}, \boldsymbol{i_c})$	Rating-based similarity between item i and item c Measured based on ratings from users C_{ic} on item i and item c, $S_{rating}(\boldsymbol{i_i}, \boldsymbol{i_c}) = \boldsymbol{I_i} \odot \boldsymbol{I_c} \mapsto \mathbb{R}$ 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" $S_{struct}(\boldsymbol{i_i}, \boldsymbol{i_c}) = \boldsymbol{I_i} \odot \boldsymbol{I_c} \mapsto \mathbb{R}_{\geqslant \boldsymbol{0}}$ The operation $\odot$ can be any structural similarity measurement Such as Jaccard, common neighbor, Sorensen or Ochiai

Table 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	$\lvert C_{ic}\rvert$
PA	$\lvert\boldsymbol{i_i}\rvert \cdot \lvert\boldsymbol{i_c}\rvert$
Jaccard	$\lvert C_{ic}\rvert / \lvert\boldsymbol{i_i} \cup \boldsymbol{i_c}\rvert$
Salton/Ochiai	$\lvert C_{ic}\rvert / \sqrt{\lvert\boldsymbol{i_i}\rvert \cdot \lvert\boldsymbol{i_c}\rvert}$
Sorensen	$2\lvert C_{ic}\rvert / (\lvert\boldsymbol{i_i}\rvert + \lvert\boldsymbol{i_c}\rvert)$
HPI	$\lvert C_{ic}\rvert / min(\lvert\boldsymbol{i_i}\rvert, \lvert\boldsymbol{i_c}\rvert)$
HDI	$\lvert C_{ic}\rvert / max(\lvert\boldsymbol{i_i}\rvert, \lvert\boldsymbol{i_c}\rvert)$
LHN1	$\lvert C_{ic}\rvert / (\lvert\boldsymbol{i_i}\rvert \cdot \lvert\boldsymbol{i_c}\rvert)$

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 $\lvert C_{ic}\rvert$ and the number of users who rated each of those 2 items $\lvert\boldsymbol{i_c}\rvert, \lvert\boldsymbol{i_i}\rvert$. The structural similarity measurement can be any structural similarity measurements approximated locally.

3.1.2. 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.

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 $\lvert C_{ic}\rvert$ to the number of users who rated each item $\lvert\boldsymbol{i_c}\rvert, \lvert\boldsymbol{i_i}\rvert$ 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.

3.2. Improvements over using rating or structural similarity alone

As constraint of using rating or structural based similarity measurement alone shown in Section 2.2, the proposed method with structural similarity measurements chosen fulfilling requirements in Section 3.1 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 (Koren, 2008) to penalize pairs of items with small number of co-rated users, the structural similarity measurement uses local ratio of $\lvert C_ij\rvert$ to the local possible co-rated users $\lvert\boldsymbol{i_i}\rvert$ and $\lvert\boldsymbol{i_j}\rvert$ 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 do not 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.

3.3. 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 15 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 (Su and Khoshgoftaar, 2009). 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.

4. Experimental setup

4.1. Data sets

For comparison purposes, the MovieLens and Netflix datasets are used as they are the most widely referenced in literature (see Table 3).

The distribution of ratings are shown in Fig. 2. We can see all three datasets have similar shape of distribution. They are all right skewed with peak at rating of 4.

Combined similarity measurement is expected to work better with the degree distribution of items of those datasets shown in Fig. 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 1.2.3.

4.2. 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:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\mathbf{R}_{iu} - \hat{\mathbf{R}}_{iu}|. \tag{11}$$

where n is the number of total predictions.

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

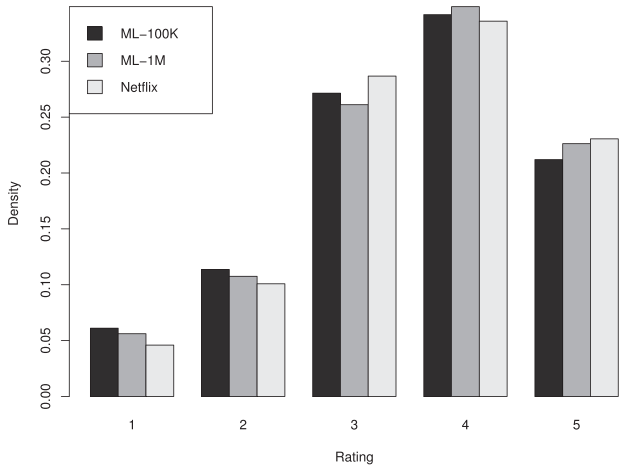


Fig. 2. Density Distribution of Ratings of MovieLens, Netflix, where x-axis is the rating from 1 to 5, y-axis is the density of number of users who rated given rating.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\mathbf{R}_{iu} - \hat{\mathbf{R}}_{iu})^2}. \tag{12}$$

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 (Willmott and Matsuura, 2005; Willmott et al., 2009): (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.

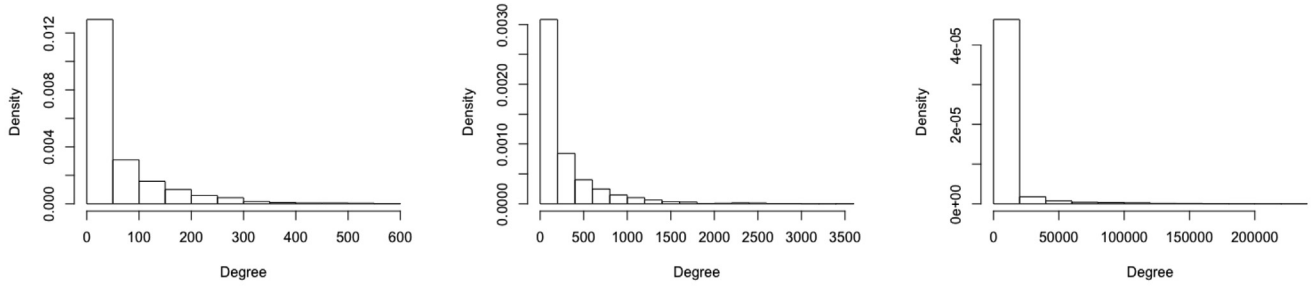
4.3. Hubness evaluation

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

$$p_i^K(x)\begin{cases} 1, & \text{if item } \times \text{ is among the } K \\ & \text{nearest neighbors of item } i \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

Table 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 (Harper and Konstan, 2016). MovieLens 100 K dataset contains data collected through MovieLens web site (movielens.umn.edu) from 1997/09/19 through 1998/04/22. MovieLens 1 M 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 100 K (Harper and Konstan, 2016)	943	1682	100,000	6%
MovieLens 1 M (Harper and Konstan, 2016)	6040	3952	1,000,209	4%
Netflix Prize (Bennett et al., 2007)	480,189	17,770	100,480,507	1%



(a) ML-100K (b) ML-1M (c) Netflix

Fig. 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.

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

$$O^K(x) = \sum_{i=1}^{n} p_i^K(x) \tag{14}$$

Then, hubness is defined as the skewness of the distribution of O^K (Radovanović et al., 2010):

$$H^K = \frac{E[(O^K - \mu_{O^K})^3]}{\sigma_{O^K}^3}. \tag{15}$$

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 (Knees et al., 2014). 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 Eq. (13) is redefined as:

$$p_i^K(x) \begin{cases} \bar{\mathbf{R}}_{i_x}, & \text{if item } \times \text{ is among the } K \\ & \text{nearest neighbors of item } i \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

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

5. Experimental results

Experiments over MovieLens dataset (ml-100 K and ml-1 M) were conducted using a 4 GHz Intel i7 CPU, 16 GB 1600 MHz 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 (Gantner et al., 2011) 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

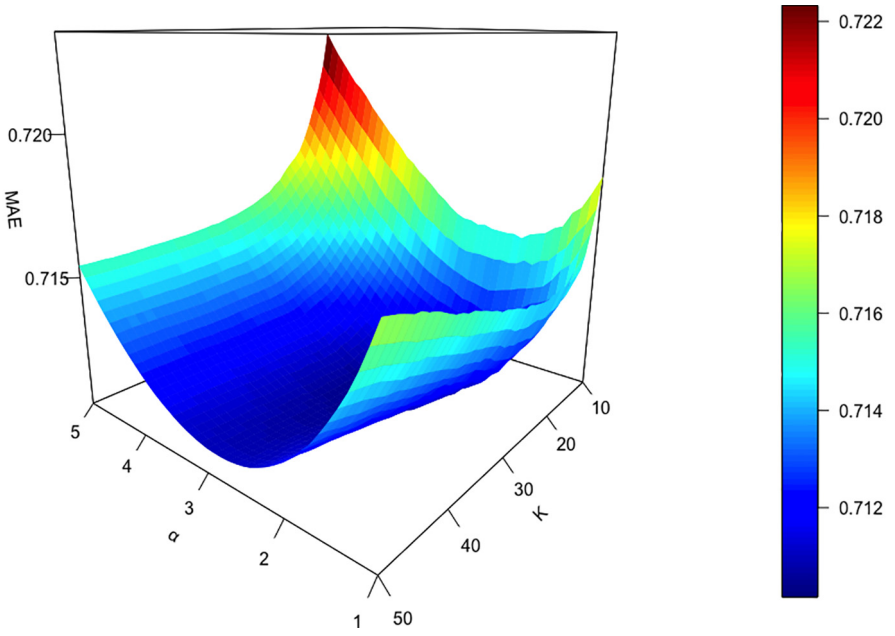
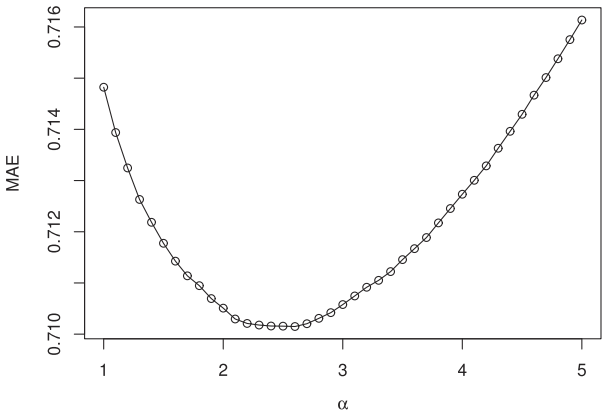
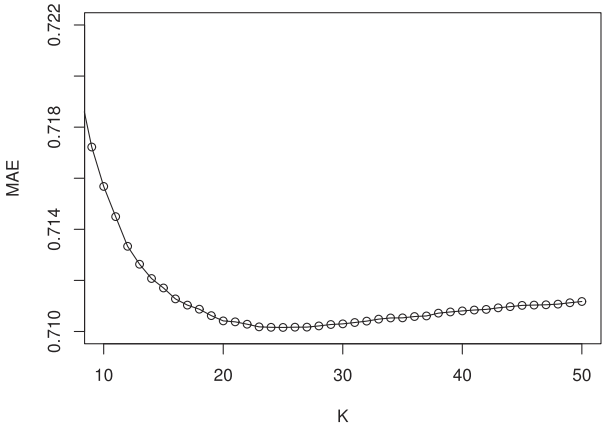


Fig. 4. MAE performance for different size of nearest neighbor list and case amplification parameter, where x-axis is the power (case amplification parameter) raised on structural similarity measurement, y-axis is the size of the nearest neighbor list, z-axis is the MAE(Mean Absolute Error). We can see 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.

Fig. 5. Accuracy performance for different nearest neighbor list size and amplification parameter. Those two figures are transections of Fig. 4.

Nelder-Mead algorithm (Piotte and Chabbert, 2009). 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 5.1. 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 5.2, we compared time and memory resources required by memory-based and model-based methods. In Section 5.3, we compared the prediction accuracy and coverage of memory-based methods by cutting off cold start users.

5.1. Prediction accuracy

5.1.1. MovieLens datasets

5-fold test is performed on the MovieLens (ml-100 K and ml-1 M) 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-100 K dataset as shown in Fig. 4. MAE is the lowest when K = 25. By fixing K = 25, it is observed MAE follows a concave shape in Fig. 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 Fig. 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-1 k and ml-1 M datasets with respect to MAE as shown in Tables 4 and 5.

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 4 and 5. This means that it can predict 3–4 times faster than traditional KNN methods with pre-calculated similarities in the preference prediction phase. Over the ml-100 K dataset, we use K = 25 whereas traditional KNN with adjusted cosine uses K = 60. For the ml-1 M 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 Fig. 6 over ml-100 K dataset.

5.1.2. Netflix dataset and modified dataset

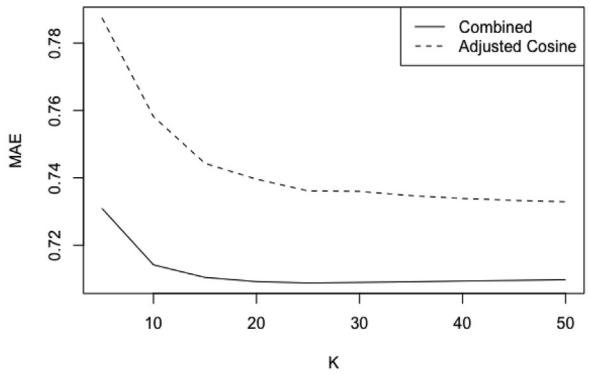
Prediction accuracy in terms of MAE and RMSE is tested on the Netflix dataset using predefined Netflix probe and testing dataset

Table 4
Comparison of prediction accuracy among state-of-the-art CFs on ML-100 K 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 (Gantner et al., 2011).

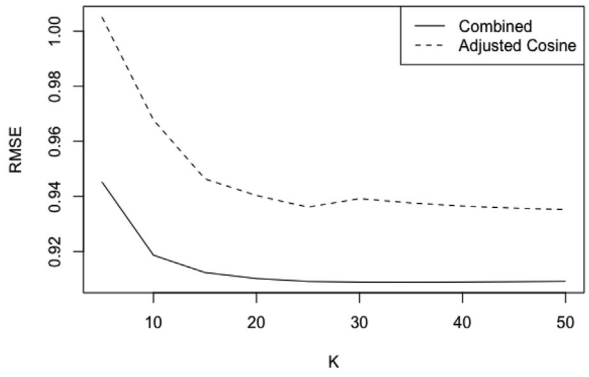
Method	Parameterization	MAE	RMSE
ItemKNNPearson	K = 60	0.736125	0.936507
ItemKNNAdjustedCosine	K = 60	0.733618	0.935528
ItemKNNPearsonReg	reg_I = 1, reg_U = 12, K = 60	0.733399	0.932648
ItemKNNAdjustedCosineReg	reg_I = 1, reg_U = 12, K = 60	0.729538	0.929991
ItemKNNBCosine	K = 25	0.725362	0.924898
BiasedMatrixFactorization	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 5
Comparison of prediction accuracy among state-of-the-art CFs on ML-1 M 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 (Gantner et al., 2011).

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



(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.

Fig. 6. Prediction accuracy with different nearest neighbor size K.

(Bennett et al., 2007). 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 6,7. 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

Table 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 (Gantner et al., 2011).

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 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 (Gantner et al., 2011).

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

training dataset that contains probe dataset to train in Tables 6 and 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) (see 7).

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-1 M dataset.

5.1.3. Netflix dataset and modified dataset without cold start users

Since collaborative filtering methods, especially memory-based CFs, are prone to cold start problem (Rubens et al., 2015; Schein et al., 2002), experiments are performed to see how much prediction accuracy improvement can be achieved on non-cold start users. experiments in Section 5.1.2 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

Table 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-100 K	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-1 M	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**

users who rated at least 20 items (Harper and Konstan, 2016), with the larger Netflix dataset, users who rated at least 50 are selected to be non-cold start users. By comparing Tables 9 and 10, the predication accuracy of all algorithms became worse in terms of higher MAE and RMSE with predefined training-probe datasets in Table 9 compared to the result of using training-probe separated dataset in Table 10. Model-based CF (BiasedMatrixFactorization) performs better than the proposed method with training dataset that contains ratings from testing dataset as shown in Table 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 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 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 (Gantner et al., 2011).

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 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 (Gantner et al., 2011).

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

Table 11
Comparison of time consumption between memory-based and model-based algorithms on ml-1 M 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 Gantner et al. (2011). 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

5.2. 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 min, whereas ItemKNNCombined takes only 2 min as shown in Table 11. Note that in order to make one single prediction, model-based methods take the same amount of time as shown in the Table 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.

5.3. 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 5.1.3. 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 12 and Fig. 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 − 1.13–158.

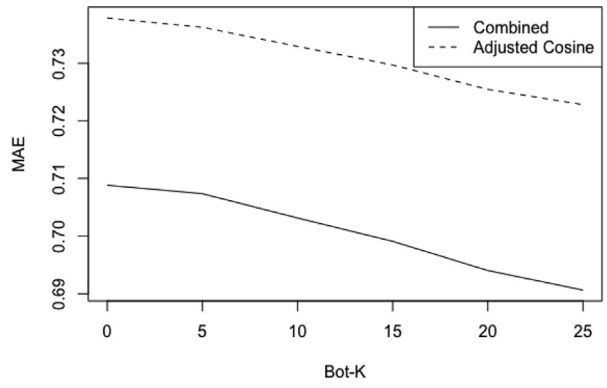
6. Discussion

6.1. Prediction accuracy

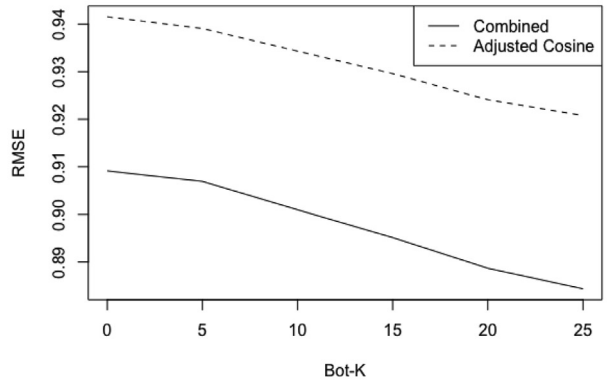
The proposed similarity measurement can improve the prediction accuracy of traditional KNN method for the MovieLens and Netflix datasets in terms of lower MAE as shown in Section 5.1. Since the number of co-rated users between each pair of items $|C_{ic}|$ and the degree distribution among items varies as shown in Fig. 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

Table 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

Fig. 7. Accuracy performance for different bot-K cutoff.

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 Eqs. 8 and 9 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 Fig. 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.

Table 13
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

Table 14
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*	*|Error|*	$Error^2$
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*	*|Error|*	$Error^2$
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

Although model-based CFs are known to be more accurate than memory-based CFs, we can see that KNN memory-based CF with the proposed similarity measurement can perform better than more sophisticated model-based CFs in terms of lower MAE. Although model-based CFs that utilize the entire user-item space to train can be more unbiased, with the nearest neighbor list ranked by the proposed similarity measurement, local KNN CFs can make prediction more preferred by this particular predicting user, thus achieve lower MAE. Therefore, local KNN CFs can perform better in prediction accuracy than model-based CFs.

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 Wang et al. (2019). Moreover, except MCCF, the prediction accuracy of those GCN approaches reported in Wang et al. (2019) 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.

6.2. Computational resources

Model based CFs require the entire user-item matrix **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 5.2. 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.

6.3. 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 5.3. 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 (Rubens et al., 2015) 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.

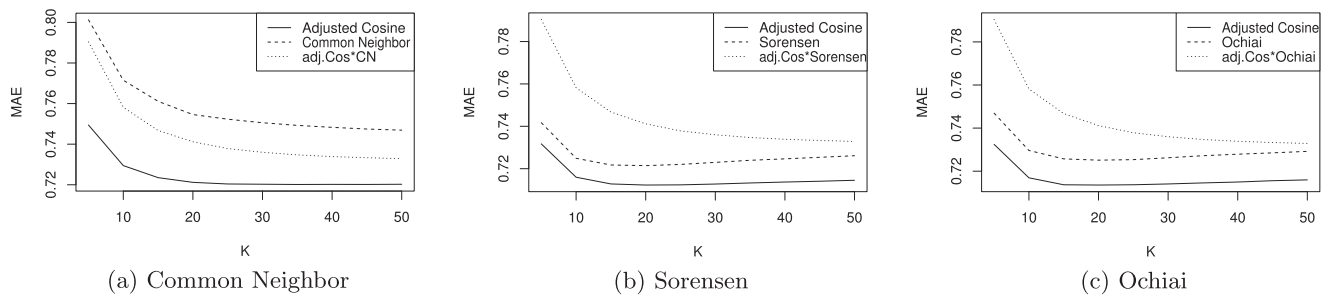


Fig. 8. MAE of adjusted cosine combined with different structural similarity measurements with $\alpha = 1$ over ML-100 K dataset. The performance of the combination of adjusted cosine and structural similarity measurement works better than using adjusted cosine or structural similarity measurement alone.

Table 15
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 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.

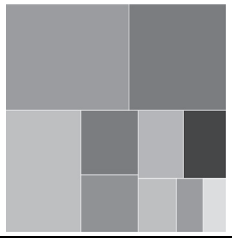
Ranking	Item ID	S_{struct}	S_{rating}	New Sim
1	1	1.0	0.5	$1^{10} \times 0.5 = 5.00e{-1}$
2	2	0.9	0.5	$0.9^{10} \times 0.5 = 1.74e{-1}$
3	3	0.8	0.8	$0.8^{10} \times 0.8 = 8.58e{-2}$
4	4	0.7	0.2	$0.7^{10} \times 0.2 = 5.64e{-3}$
5	5	0.6	0.1	$0.6^{10} \times 0.1 = 6.04e{-4}$
6	6	0.5	0.3	$0.5^{10} \times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

(c) 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	$0.6^{1} \times 0.1 = 0.06$

(d) 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.



(*continued on next page*)

(d) CASE 3: $\alpha = 1.5$ 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	$0.1^{1.5} \times 0.7 = 0.0221$

(e) CASE 3: Percentage of each neighbor items when $\alpha = 1.5$: 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.

7. Conclusion

In this paper, 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. Again, 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 (Koren, 2007).

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.

CRediT authorship contribution statement

Mario Ventresca: Conceptualization, Methodology, Formal analysis, Writing - review & editing, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

Thanks to David Zage for valuable initial discussion.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17*, 734–749.
Aggarwal, C. C. et al. (2016). *Recommender systems*. Springer.
Ahn, H. J. (2008). A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences, 178*, 37–51.
Bennett, J., Lanning, S. & et al. (2007). The netflix prize. In Proceedings of KDD cup and workshop (Vol. 2007, p. 35). New York, NY, USA.
Billsus, D., & Pazzani, M.J. (1998). Learning collaborative information filters. In Icml (Vol. 98, pp. 46–54).
Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research, 3*, 993–1022.
Bobadilla, J., Ortega, F., Hernando, A., & Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems, 26*, 225–238.
Bobadilla, J., Serradilla, F., & Bernal, J. (2010). A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems, 23*, 520–528.
Breese, J.S., Heckerman, D. & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the fourteenth conference on uncertainty in artificial intelligence (pp. 43–52). Morgan Kaufmann Publishers Inc.
Brynjolfsson, E., Hu, Y., & Smith, M. D. (2003). Consumer surplus in the digital economy: Estimating the value of increased product variety at online booksellers. *Management Science, 49*, 1580–1596.
Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*, 331–370.
Casey, M. A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., & Slaney, M. (2008). Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE, 96*, 668–696.
Chen, L., Wu, L., Hong, R., Zhang, K. & Wang, M. (2020). Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. arXiv preprint arXiv:2001.10167.
Desrosiers, C., & Karypis, G. (2011). A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook* (pp. 107–144). Springer.
Dias, M. B., Locher, D., Li, M., El-Deredy, W. & Lisboa, P.J. (2008). The value of personalised recommender systems to e-business: a case study. In Proceedings of the 2008 ACM conference on Recommender systems (pp. 291–294). ACM.
Feng, J., Fengs, X., Zhang, N., & Peng, J. (2018). An improved collaborative filtering method based on similarity. *PloS One, 13* e0204003.
Gantner, Z., Rendle, S., Freudenthaler, C. & Schmidt-Thieme, L. (2011). Mymedialite: A free recommender system library. In Proceedings of the fifth ACM conference on recommender systems (pp. 305–308). ACM.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*, 61–70.
Hamilton, W., Ying, Z. & Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in neural information processing systems (pp. 1024–1034).
Harper, F. M., & Konstan, J. A. (2016). The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TIIS), 5*, 19.
He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web* (pp. 173–182).
Herlocker, J. L., Konstan, J. A., Borchers, A. & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval (pp. 230–237). ACM.
Hofmann, T. (1999). Probabilistic latent semantic indexing. In Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval (pp. 50–57). ACM.
Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS), 22*, 89–115.
Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory, 14*, 55–63.
Kipf, T. & Welling, M. (2017). Semi-supervised classification with graph convolutional networks iclr.
Knees, P., Schnitzer, D. & Flexer, A. (2014). Improving neighborhood-based collaborative filtering by reducing hubness. In Proceedings of international conference on multimedia retrieval (p. 161). ACM.
Koren, Y. (2007). How useful is a lower rmse? URL https://www.webcitation.org/65tSpdL9E?url=http://www.netflixprize.com/community/viewtopic.php?id=828.
Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining (pp. 426–434). ACM.
Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE, 7*, 76–80.
Liu, H., Hu, Z., Mian, A., Tian, H., & Zhu, X. (2014). A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems, 56*, 156–166.
Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., & Zhou, T. (2012). Recommender systems. *Physics Reports, 519*, 1–49.
Maja Kabiljo, A. I. (2015). Recommending items to more than a billion people. https://code.facebook.com/posts/861999383875667/recommending-items-to-more-than-a-billion-people/.
Melville, P., Mooney, R. J. & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In Aaai/iaai (pp. 187–192).
Mu, Y., Xiao, N., Tang, R., Luo, L., & Yin, X. (2019). An efficient similarity measure for collaborative filtering. *Procedia Computer Science, 147*, 416–421.
Nanopoulos, A., Radovanović, M. & Ivanović, M. (2009). How does high dimensionality affect collaborative filtering? In Proceedings of the third ACM conference on recommender systems (pp. 293–296). ACM.
Patra, B. K., Launonen, R., Ollikainen, V., & Nandi, S. (2015). A new similarity measure using bhattacharyya coefficient for collaborative filtering in sparse data. *Knowledge-Based Systems, 82*, 163–177.
Pazzani, M. J. & Billsus, D. (2007). Content-based recommendation systems. In The adaptive web (pp. 325–341). Springer.
Piotte, M., & Chabbert, M. (2009). The pragmatic theory solution to the netflix grand prize. *Netflix Prize*. documentation.
Polatidis, N., & Georgiadis, C. K. (2016). A multi-level collaborative filtering method that improves recommendations. *Expert Systems with Applications, 48*, 100–110.
Radovanović, M., Nanopoulos, A., & Ivanović, M. (2010). Hubs in space: Popular nearest neighbors in high-dimensional data. *Journal of Machine Learning Research, 11*, 2487–2531.
Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. & Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on computer supported cooperative work (pp. 175–186). ACM.
Rubens, N., Elahi, M., Sugiyama, M., & Kaplan, D. (2015). Active learning in recommender systems. In *Recommender systems handbook* (pp. 809–846). Springer.
Sarwar, B., Karypis, G., Konstan, J. & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285–295). ACM.
Schein, A. I., Popescul, A., Ungar, L. H. & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In Proceedings of the 25th annual international ACM SIGIR conference on research and development in information retrieval (pp. 253–260). ACM.
Schnitzer, D., Flexer, A., Schedl, M., & Widmer, G. (2012). Local and global scaling reduce hubs in space. *Journal of Machine Learning Research, 13*, 2871–2902.
Su, X., Greiner, R., Khoshgoftaar, T. M., & Zhu, X. (2007). Hybrid collaborative filtering algorithms using a mixture of experts. In *Proceedings of the IEEE/WIC/ACM international conference on web intelligence* (pp. 645–649). IEEE Computer Society.
Su, X. & Khoshgoftaar, T. M. (2006). Collaborative filtering for multi-class data using belief nets algorithms. In 2006 18th IEEE international conference on tools with artificial intelligence (ICTAI'06) (pp. 497–504). IEEE.
Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, 2009*, 4.
Sun, S.-B., Zhang, Z.-H., Dong, X.-L., Zhang, H.-R., Li, T.-J., Zhang, L., & Min, F. (2017). Integrating triangle and jaccard similarities for recommendation. *PloS One, 12* e0183570.
Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 61*, 611–622.
Wang, X., He, X., Wang, M., Feng, F., & Chua, T.-S. (2019). Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval* (pp. 165–174).
Wang, X., Wang, R., Shi, C., Song, G. & Li, Q. (2019b). Multi-component graph convolutional collaborative filtering. arXiv preprint arXiv:1911.10699.
Wang, X. & Wang, Y. (2014). Improving content-based and hybrid music recommendation using deep learning. In Proceedings of the 22nd ACM international conference on multimedia (pp. 627–636). ACM.
Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate Research, 30*, 79–82.
Willmott, C. J., Matsuura, K., & Robeson, S. M. (2009). Ambiguities inherent in sums-of-squares-based error statistics. *Atmospheric Environment, 43*, 749–752.
Wu, F., Zhang, T., Souza Jr., A.H. d., Fifty, C., Yu, T. & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. arXiv preprint arXiv:1902.07153.
Xue, G. -R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y. & Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval (pp. 114–121). ACM.
Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 974–983).
Zhang, F., Gong, T., Lee, V. E., Zhao, G., Rong, C., & Qu, G. (2016). Fast algorithms to evaluate collaborative filtering recommender systems. *Knowledge-Based Systems, 96*, 96–103.