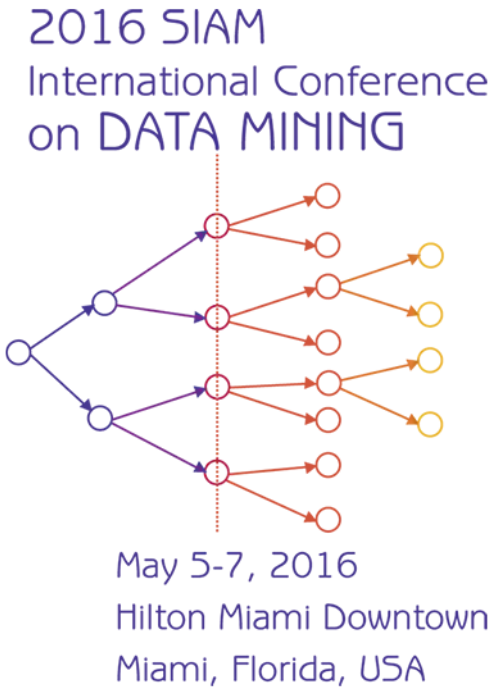
MLRec 2016

2nd International Workshop on Machine Learning Methods for Recommender Systems

May 2016, Miami, Florida, USA

In Conjunction with
16th SIAM International Conference on Data Mining (SDM 2016)



Workshop Chairs

Jiayu Zhou, Michigan State University
Deguang Kong, Samsung Research America
George Karypis, University of Minnesota

Publicity Chair

Shiyu Chang, University of Illinois at Urbana-Champaign

Website Chair

Zhangyang Wang, University of Illinois at Urbana-Champaign

Program Committee

Shiva Kasiviswanathan, Samsung Research America
Shengbo Guo, Facebook
Yao Wu, Simon Fraser University
Lei Cen, Purdue University
Jianpeng Xu, Michigan State University

Invited Speakers

Tina Eliassi-Rad, Rutgers University
Julian McAuley, University of California, San Diego
Hanghang Tong, Arizona State University
Hongxia Jin, Samsung Research America

Overview

The MLRec 2016 workshop aims to bring the attention of researchers to the various data mining and machine learning methods for recommender systems.

Since the introduction of recommender system, there are a lot of machine learning and data mining algorithms designed for effective and efficient recommendation. To name a few, the matrix factorization techniques are widely used to model the latent space in which users and items interact with each other. The factorization machine uses bilinear regression models to capture the non-linear interactions among the user features and item features. In the past years, researchers have utilized many machine learning techniques such as online learning, metric learning, sparse learning, multi-task learning also to foster the development of recommender systems.

This workshop focuses on applying novel as well as existing machine learning and data mining methodologies for improving recommender systems. Indeed there are many established conferences such as NIPS and ICML that focus on the study of theoretical properties of machine learning algorithms. On the other hand, the recent developed conference ACM RecSys focuses on different aspects of designing and implementing recommender systems. We believe that there is a gap between these two ends, and this workshop aims at bridging the recent advances of machine learning and data mining algorithms to improving recommender systems. Since many recommendation approaches are built upon data mining and machine learning algorithms, these approaches are deeply rooted in their foundations. As such, there is an urgent need for researchers from the two communities to jointly work on 1) what are the recent developed machine learning and data mining techniques that can be leveraged to address challenges in recommender systems, and 2) from challenges in recommender systems, what are the practical research directions in the machine learning and data mining community.

We encourage submissions on a variety of topics, including but not limited to:

1. Novel machine learning algorithms for recommender systems, e.g., new content aware recommendation algorithms, new algorithms for matrix factorization handling cold-start items.

2. Novel approaches for applying existing machine learning algorithms, e.g., applying bilinear models, sparse learning, metric learning, neural networks and deep learning, for recommender systems.

3. Novel optimization algorithms and analysis for improving recommender systems, e.g., parallel/distributed optimization techniques and efficient stochastic gradient descent.

4. Industrial practices and implementations of recommendation systems, e.g., feature engineering, model ensemble, and lessons from large-scale implementations of recommender systems.

We believe that advancements on these topics will benefit a variety of algorithm and application domains.

Acknowledgments

We appreciate the efforts of the Program Committee for their comments and feedback on the submissions.

Table of Contents

Semi-supervised Collaborative Ranking with Push at Top

Iman Barjasteh[†*], Rana Forsati[‡*], Abdol-Hossein Esfahanian[‡], Hayder Radha[†]
[†]Department of Electrical and Computer Engineering, Michigan State University
[‡]Department of Computer Science and Engineering, Michigan State University
{forsati,esfahanian}@cse.msu.edu, {barjaste,radha}@msu.edu

Abstract

Existing collaborative ranking based recommender systems tend to perform best when there is enough observed ratings for each user and the observation is made completely at random. Under this setting recommender systems can properly suggest a list of recommendations according to the user interests. However, when the observed ratings are extremely sparse (e.g. in the case of cold-start users where no rating data is available), and are not sampled uniformly at random, existing ranking methods fail to effectively leverage side information to transduct the knowledge from existing ratings to unobserved ones. We propose a **semi-supervised collaborative ranking** model, dubbed S^{2}COR, to improve the quality of cold-start recommendation. S^{2}COR mitigates the sparsity issue by leveraging side information about *both observed and missing* ratings by collaboratively learning the ranking model. This enables it to deal with the case of missing data not at random, but to also effectively incorporate the available side information in transduction. We experimentally evaluated our proposed algorithm on a number of challenging real-world datasets and compared against state-of-the-art models for cold-start recommendation. We report significantly higher quality recommendations with our algorithm compared to the state-of-the-art.

1 Introduction

Due to the popularity and exponential growth of e-commerce and online streaming websites, a compelling demand has been created for efficient recommender systems to guide users toward items of their interests (e.g. products, books, movies) [1]. In collaborative filtering (CF) methods such as matrix factorization [12], where the aim is to accurately predict the ratings, the latent features are extracted in a way to minimize the prediction error measured in terms of popular performance measures such as root mean square error (RMSE). In spark contrast to CF, in collaborating ranking (CR) models [12, 7, 29, 8], where the goal is to rank the unrated items in the order of relevance to the user, the popular ranking measures such as as discounted cumulative gain (DCG), normalized discounted cumulative gain (NDCG), and average precision (AP) [11] are often employed to collaboratively learn a ranking model for the latent features.

Recent studies have demonstrated that CR models lead to significantly higher ranking accuracy over their traditional CF counterparts that optimize rating prediction. This is important considering the fact that what we really care in recommendation is not the actual values of ratings, but the order of items to be recommended to a specific user. Therefore, the error measures such as RMSE are often hopelessly insufficient, as they place equal emphasis on all the ratings. Among ranking models, the methods that mainly concentrate on the *top of the list* have received a considerable amount of attention, due to the higher probability of examining the top portion of the list of recommendations by users. Therefore, the introduction of ranking metrics such as push norm or infinite norm [19, 3, 7, 13], sparked a widespread interest in CR models and has been proven to be more effective in practice [28, 7].

Although CR models for recommender systems has been studied extensively and some progress has been made, however, the state of affairs remains unsettled: the issue of handling *cold-start* items in ranking models and coping with *not missing at random* assumption of ratings are elusive open issues. First, in many real world applications, the rating data are very sparse (e.g., the density of the data is around 1% for many publicly available datasets) or for a subset of users or items the rating data is entirely missing

[*]These authors contributed equally to this work.

(knows as cold-start user and cold-start item problem, respectively) [21]. Second, collaborative filtering and ranking models rely on the critical assumption that the missing ratings are sampled uniformly at random. However, in many real applications of recommender systems, this assumption is not believed to hold, as invariably some users are more active than others and some items are rated by many people while others are rarely rated [27]. These issues have been investigated in factorization based methods, nonetheless, it is not straightforward to adapt them to CR models and are left open [7].

In this paper, we introduce a *semi-supervised collaborative ranking* model, dubbed $\texttt{S}^2\texttt{COR}$, by leveraging side information about *both observed and missing ratings* in collaboratively learning the ranking model. In the learned model, unrated items are conservatively pushed after the relevant and before the irrelevant items in the ranked list of items for each individual user. This crucial difference greatly boosts the performance and limits the bias caused by learning only from sparse non-random observed ratings.

To build the intuition on how incorporating missing ratings in $\texttt{S}^2\texttt{COR}$ is beneficial in handling cold-start problem and mitigating data sparsity issue, we note that in many real world applications the available feedback on items is extremely sparse, and therefore the ranking models fail to effectively leverage the available side information in transdcuting the knowledge from existing ratings to unobserved ones. This problem becomes especially eminent in cases where surrogate ranking models such as pairwise models are used due to their computational virtues, where the unobserved ratings do not play any role in learning the model. As a result, by leveraging rich sources of information about all items, one can potentially bridge the gap between existing items and new items to overcome the cold-start problem.

Turning to the non-random sampling issue of observed ratings, we note that the non-randomness is observing the ratings creates a bias in learning the model that negatively impacts the future predictions and may degrade the resulting recommendation accuracy if ignored. Therefore, the nature of missing ratings has to be modeled precisely as to obtain correct results. To reduce the effect of bias, the proposed ranking model takes a conservative approach and pushes the items with unknown ratings to the middle of ranked list, i.e., after the relevant and before the irrelevant items. This is equivalent to assuming a prior about the unknown ratings which is believed to perform well as investigated in [9].

We conduct thorough experiments on real datasets and compare our results with the state-of-the-art models for cold-start recommendation to demonstrate the effectiveness of our proposed algorithm in recommendation at the top of the list and mitigating the data sparsity issue.

Organization. This paper is organized as follows. We briefly review related work in Section 2. We establish the notation and formally define the problem in Section 3. In Section 4, we propose the semi-supervised collaborative ranking model with a push at the top of the list. We empirically evaluate the proposed method in Section 5, and conclude in Section 6.

2 Related Work

Collaborative ranking for recommendation. The last few years have seen a resurgence in collaborative ranking centered around the technique of exploiting low-rank structures, an approach we take as well. Several approaches to CR have recently been proposed that are mainly inspired by the analogy between query-document relations in IR and user-item relations in recommender systems. The PMF-based approach [4] uses the latent representations produced by matrix factorization as user-item features and learns a ranking model on these features. CofiRank [30] learns latent representations that minimize a ranking-based loss instead of the squared error. ListRankMF [23] aims at minimizing the cross entropy between the predict item permutation probability and true item permutation probability. In [13] a method for Local Collaborative Ranking (LCR) where ideas of local low-rank matrix approximation were applied to the pairwise ranking loss minimization framework is introduced.

Cold-start recommendation with side information. Due in part to its importance, there has been an active line of work to address difficulties associated with cold-start users and items, where a common theme among them is to exploit auxiliary information about users or items besides the rating data that are usually available [24]. A feature based regression ranking model for predicting the values (rates) of user-item matrix in cold-start scenarios by leveraging all information available for users and items is proposed in [17]. The kernelized matrix factorization approach studied in [31], which incorporates the auxiliary information into the MF. In [20] joint factorization of the user-item and item-feature matrices by using the same item latent feature matrix in both decompositions is utilized.

Recommendation with not missing at random ratings. Substantial evidence for violations of the missing at random condition in recommender systems is reported in [16] and it has been showed that incorporating an explicit model of the missing data mechanism can lead to significant improvements in prediction performance.The first study of the effect of non-random missing data on collaborative ranking is presented in [15]. In [25] an EM algorithm to optimize in turn the factorization and the estimation of missing values.

3 Preliminaries

In this section we establish the notation used throughout the paper and formally describe our problem setting.

Scalars are denoted by lower case letters and vectors by bold face lower case letters such as $\mathbf{u}$. We use bold face upper case letters such as $\mathbf{M}$ to denote matrices. The Frobenius norm of a matrix $\mathbf{M} \in \mathbb{R}^{n\times m}$ is denoted by $\|\mathbf{M}\|_{\mathrm{F}}$, i.e, $\|\mathbf{M}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}|M_{ij}|^2}$ and its (i,j)th entry is denoted by $A_{i,j}$. The trace norm of a matrix is denoted by $\|\mathbf{M}\|_*$ which is defined as the sum of its singular values. The transpose of a vector and a matrix denoted by $\mathbf{u}^\top$ and $\mathbf{U}^\top$, respectively. We use $[n]$ to denote the set on integers $\{1, 2, \cdots, n\}$. The set of non-negative real numbers is denoted by $\mathbb{R}_+$. The indicator function is denoted by $\mathbb{I}[\cdot]$. For a vector $\mathbf{u} \in \mathbb{R}^p$ we use $\|\mathbf{u}\|_1 = \sum_{i=1}^{p}|\mathbf{u}_i|$, $\|\mathbf{u}\|_2 = \left(\sum_{i=1}^{p}|\mathbf{u}_i|^2\right)^{1/2}$, and $\|\mathbf{u}\|_\infty = \max_{1\le i\le p}\mathbf{u}_i$ to denote its ℓ_1, ℓ_2, and ℓ_∞ norms, respectively. The dot product between two vectors $\mathbf{u}$ and $\mathbf{u}'$ is denoted by either $\langle \mathbf{u}, \mathbf{u}'\rangle$ or $\mathbf{u}^\top\mathbf{u}'$.

In collaborative filtering we assume that there is a set of n users $\mathcal{U} = \{u_1, \cdots, u_n\}$ and a set of m items $\mathcal{I} = \{i_1, \cdots, i_m\}$ where each user u_i expresses opinions about a set of items. The rating information is summarized in an $n \times m$ matrix $\mathbf{R} \in \{-1, +1, ?\}^{n\times m}, 1 \le i \le n, 1 \le j \le m$ where the rows correspond to the users and the columns correspond to the items and (p, q)th entry is the rate given by user u_p to the item i_q. We note that the rating matrix is partially observed and it is sparse in most cases. We are mainly interested in recommending a set of items for an active user such that the user has not rated these items before.

4 Transductive Collaborating Ranking

We now turn our attention to the main thrust of the paper where we present our transductive collaborative ranking algorithm with accuracy at top by exploiting the features of unrated data. We begin with the basic formulation and then extend it to incorporate the unrated items. The pseudo-code of the resulting learning algorithm is provided in Algorithm 1.

4.1 A basic formulation

We consider a ranking problem, where, given a set of users $\mathcal{U}$ and known user feedback on a set of items $\mathcal{I}$, the goal is to generate rankings of unobserved items, adapted to each of the users' preferences. Here we consider the bipartite setting in which items are either relevant (positive) or irrelevant (negative). Many ranking methods have been developed for bipartite ranking, and most of them are essentially based on pairwise ranking. These algorithms reduce the ranking problem into a binary classification problem by treating each relevant/irrelevant instance pair as a single object to be classified [14].

As mentioned above, most research has concentrated on the rating prediction problem in CF where the aim is to accurately predict the ratings for the unrated items for each user. However, most applications that use CF typically aim to recommend only a small ranked set of items to each user. Thus rather than concentrating on rating prediction we instead approach this problem from the ranking viewpoint where the goal is to rank the unrated items in the order of relevance to the user. Moreover, it is desirable to concentrate aggressively on top portion of the ranked list to include mostly relevant items and push irrelevant items down from the top. Specifically, we propose an algorithm that maximizes the number of relevant items which are pushed to the absolute top of the list by utilizing the P-Norm Push ranking measure which is specially designed for this purpose [19] .

For simplicity of exposition, let us first consider the ranking model for a single user u. Let $\mathcal{X}^+ = \{\mathbf{x}_1^+, \cdots, \mathbf{x}_{n_+}^+\}$ and $\mathcal{X}^- = \{\mathbf{x}_1^-, \cdots, \mathbf{x}_{n_-}^-\}$ be the set of feature vectors of n_+ relevant and n_- irrelevant items to user u, respectively. We consider linear ranking functions where each item features vector $\mathbf{x} \in \mathbb{R}^d$ is mapped to a score $\mathbf{w}^\top\mathbf{x}$. The goal is to find parameters $\mathbf{w}$ for each user such that the ranking function best captures past feedback from the user. The goal of ranking is to maximize the number of relevant items ranked above the highest-ranking irrelevant item. We cast this idea for each user u individ-

Algorithm 1 S^{2}COR

input: $\lambda \in \mathbb{R}_+$: the regularization parameter, and $\{\eta_t\}_{t\geq 1}$: the sequence of scalar step sizes
Initialize $\mathbf{W}_0 \in \mathbb{R}^{n\times d}$
Choose an appropriate step size
for $t = 1, \ldots, T$ **do**
  Compute the sub-gradient of $\mathbf{G}_t \in \partial\mathcal{L}(\mathbf{W}_t)$ using Eq. (**??**)
  $[\mathbf{U}_t, \mathbf{\Sigma}_t, \mathbf{V}_t] \leftarrow \texttt{SVD}(\mathbf{W}_{t-1} - \frac{1}{\eta_{t-1}}\mathbf{G}_t))$
  $\mathbf{W}_t \leftarrow \mathbf{U}_t \left[\mathbf{\Sigma} - \frac{\lambda}{\eta_{t-1}}\mathbf{I}\right]_+ \mathbf{V}_t^\top$
end for
output:

ually into the following optimization problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{n^+}\sum_{i=1}^{n^+} \mathbb{I}\left[\langle \mathbf{w}, \mathbf{x}_i^+\rangle \leq \max_{1\leq j\leq n^-} \langle \mathbf{w}, \mathbf{x}_j^-\rangle\right] \quad (1)$$

where $\mathbb{I}[\cdot]$ is the indicator function which returns 1 when the input is true and 0 otherwise, n^+ and n^- are the the number of relevant and irrelevant items to user u, respectively.

Let us now derive the general form of our objective. We hypothesize that most users base their decisions about items based on a number of latent features about the items. In order to uncover these latent feature dimensions, we impose a low-rank constraint on the set of parameters for all users. To this end, let $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n]^\top \in \mathbb{R}^{n\times d}$ denote the matrix of all parameter vectors for n users. Let $\mathcal{I}_i^+ \subseteq \{1, 2, \ldots, m\}$ and $\mathcal{I}_i^- \subseteq \{1, 2, \ldots, m\}$ be the set of relevant and irrelevant items of ith user, respectively. The overall objective for all users is formulated as follows:

$$\mathcal{F}(\mathbf{W}) = \lambda\|\mathbf{W}\|_* + \sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^+|}\sum_{j\in\mathcal{I}_i^+} \mathbb{I}\left[\langle \mathbf{w}_i, \mathbf{x}_j\rangle \leq \max_{k\in\mathcal{I}_i^-}\langle \mathbf{w}_i, \mathbf{x}_k\rangle\right]\right) \quad (2)$$

where $\|\cdot\|_*$ is the trace norm (also known as nuclear norm) which is the sum of the singular values of the input matrix.

The objective in Eq. (2) is composed of two terms. The first term is the regularization term and is introduced to capture the factor model intuition discussed above. The premise behind a factor model is that there is only a small number of factors influencing the preferences, and that a user's preference vector is determined by how each factor applies to that user. Therefore, the parameter vectors of all users must lie in a low-dimensional subspace. Trace-norm regularization is a widely-used and successful approach for collaborative filtering and matrix completion. The trace-norm regularization is well-known to be a convex surrogate to the matrix rank, and has repeatedly shown good performance in practice [26, 6]. The second term is introduced to push the relevant items of each user to the top of the list when ranked based on the user parameter vector and item features.

The above optimization problem is intractable due to the non-convex indicator function. To design practical learning algorithms, we replace the indicator function in (2) with its convex surrogate. To this end, define the convex loss function $\ell : \mathbb{R} \mapsto \mathbb{R}_+$ as $\ell(x) = [1 - x]_+$. This is the widely used hinge loss in SVM classification (see e.g., [5]) [1]. This loss function reflects the amount by which the constraints are not satisfied. By replacing the non-convex indicator function with this convex surrogate leads to the following tractable convex optimization problem:

$$\mathcal{F}(\mathbf{W}) = \lambda\|\mathbf{W}\|_* + \sum_{i=1}^{n}\left(\frac{1}{|\mathcal{I}_i^+|}\sum_{j\in\mathcal{I}_i^+} \ell\left(\langle \mathbf{w}_i, \mathbf{x}_j\rangle - \|\mathbf{X}_i^-\mathbf{w}_i\|_\infty\right)\right) \quad (3)$$

where $\mathbf{X}_i^- = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_i^-}]^\top$ is the matrix of features of n_i^- irrelevant items in $\mathcal{I}_i^-$ and $\|\cdot\|_\infty$ is the max norm of a vector.

4.2 Semi-supervised collaborative ranking

In this part, we extend the proposed ranking idea to learn both from rated as well as unrated items. The motivation of incorporating unrated items comes from the following key observations. First, we note that commonly there is a small set of rated (either relevant or irrelevant) items for each user and a large number of unrated items. As it can be seen from Eq. (2), the unrated items do not play any role in learning the model for each user as the learning is only based on the pair of rated items. When the feature information for items is available, it would be very helpful if one can leverage such unrated items in the learning-to-rank process to effectively leverage the available side information. By leveraging both

[1] We note that other convex loss functions such as exponential loss $\ell(x) = \exp(-x)$, and logistic loss $\ell(x) = \log(1 + \exp(-x))$ also can be used as the surrogates of indicator function, but for the simplicity of derivation we only consider the hinge loss here.

types of rated and unrated items, we can compensate for the lack of rating data. Second, the non-randomness in observing the observed ratings creates a bias in learning the model that may degrade the resulting recommendation accuracy. Therefore, finding a precise model to reduce the effect of bias introduced by non-random missing ratings seems essential.

To address these two issues, we extend the basic formulation in Eq. (2) to incorporate items with missing ratings in ranking of items for individual users. A conservative solution is to push the items with unknown ratings to the middle of ranked list, i.e., after the relevant and before the irrelevant items. To do so, let $\mathcal{I}_i^\circ = \mathcal{I} \setminus \left(\mathcal{I}_i^+ \cup \mathcal{I}_i^-\right)$ denote the set of items unrated for user $i \in \mathcal{U}$. We introduce two extra terms in the objective in Eq. (2) to push the unrated items $\mathcal{I}_\circ^i$ below the relevant items and above the irrelevant items, which yilelds the following objective:

$$\begin{aligned}\mathcal{L}(\mathbf{w}) = \quad & \frac{1}{|\mathcal{I}_i^+|} \sum_{i \in \mathcal{I}_i^+} \ell\left(\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \max_{j \in \mathcal{I}_i^-} \langle \mathbf{w}, \mathbf{x}_j \rangle\right) \\ + \quad & \frac{1}{|\mathcal{I}_i^+|} \sum_{i \in \mathcal{I}_i^+} \ell\left(\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \max_{j \in \mathcal{I}_i^\circ} \langle \mathbf{w}, \mathbf{x}_j \rangle\right) \\ + \quad & \frac{1}{|\mathcal{I}_i^\circ|} \sum_{i \in \mathcal{I}_i^\circ} \ell\left(\langle \mathbf{w}, \mathbf{x}_i \rangle \leq \max_{j \in \mathcal{I}_i^-} \langle \mathbf{w}, \mathbf{x}_j \rangle\right)\end{aligned} \tag{4}$$

Equipped with the objective of individual users, we now turn to the final collaborating ranking objective as:

$$\begin{aligned}\mathcal{F}(\mathbf{W}) &= \lambda \|\mathbf{W}\|_* \\ &+ \sum_{i=1}^n \left(\frac{1}{|\mathcal{I}_i^+|} \sum_{j \in \mathcal{I}_i^+} \ell\left(\langle \mathbf{w}_i, \mathbf{x}_j \rangle - \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty\right) \right) \\ &+ \sum_{i=1}^n \left(\frac{1}{|\mathcal{I}_i^+|} \sum_{j \in \mathcal{I}_i^+} \ell\left(\langle \mathbf{w}_i, \mathbf{x}_j \rangle - \|\mathbf{X}_i^\circ \mathbf{w}_i\|_\infty\right) \right) \\ &+ \sum_{i=1}^n \left(\frac{1}{|\mathcal{I}_i^\circ|} \sum_{j \in \mathcal{I}_i^\circ} \ell\left(\langle \mathbf{w}_i, \mathbf{x}_j \rangle - \|\mathbf{X}_i^- \mathbf{w}_i\|_\infty\right) \right),\end{aligned} \tag{5}$$

where $\mathbf{X}_i^\circ = [\mathbf{x}_1, \ldots, \mathbf{x}_{n_i^\circ}]^\top$ is the matrix of n_i° unrated items in $\mathcal{I}_i^\circ$.

5 Experiments

In this section, we conduct exhaustive experiments to demonstrate the merits and advantages of the proposed algorithm. We conduct our experiments on three well-known datasets MovieLens, Amazon and CiteULike.

Table 1: Statistics of real datasets used in our experiments.

Statistics	ML-IMDB	Amazon	CiteULike
# users	2,113	13,097	3,272
# items	8,645	11,077	21,508
# ratings	739,973	175,612	180,622
# features	8,744	5,766	6,359
Density	4.05%	0.12%	0.13%

5.1 Datasets

- **ML-IMDB.** We used ML-IMDB which is a dataset extracted from the IMDB and the MovieLens 1M datasets by mapping the MovieLens and IMDB and collecting the movies that have plots and keywords.
- **Amazon.** We used the dataset of best-selling books and their ratings in Amazon. Each book has a one or two paragraphs of textual description, which has been used to have a set of features of the books.
- **CiteULike.** It is an online free service for managing and discovering scholarly references. Users can add those articles that they are interested in to their libraries. Collected articles in a user's library will be considered as relevant items for that user. This dataset does not have explicit irrelevant items and was chosen to illustrate the effect of considering missing data while only having relevant itmes.

For all above datasets, the description about the items were tokenized and after removing the stop words, the rest of the words were stemmed. Then those words that have been appeared in less than 20 items and more that 20% of the items were also removed [22]. At the end, the TF-IDF was applied on the remaining words and the TF-IDF scores represented the features of the items. The statistics of the datasets are given in Table 1. As it is shown in Table 1, all these three datasets have high dimensional feature space.

5.2 Metrics

We adopt the widely used metrics, Discounted Cumulative Gain at n and Recall at n, for assessing the performance of our and baseline algorithms. For each user u, given an item i, let s_k be the relevance score of the item ranked at position k, where $s_k = 1/n$ if the item is relevant to the user u and $s_k = 0$ otherwise.

Discounted Cumulative Gain at n, is defined as:

$$\mathrm{DCG_u@}n = s_1 + \sum_{k=2}^{n} \frac{s_k}{\log_2(k)}$$

If we divide the $\mathrm{DCG_u@}n$ by its maximum value, we get the $\mathrm{NDCG_u@}n$ value. Given the list of top-n item recommendations for each user u, Recall at n will count the number of relevant items appeared in that list. Recall at n is defined as:

$$\mathrm{REC_u@}n = \frac{|\{\text{relevant items to } u\} \cap \{\text{top-}n \text{ items}\}|}{|\{\text{top-}n \text{ items}\}|}$$

DCG@n, $\mathrm{NDCG_u@}n$ and REC@n will be computed for each user and then will be averaged over all users.

5.3 Methodology

Given the partially observed rating matrix, we transformed the observed ratings of all datasets from a multi-level relevance scale to a two-level scale $(+1, -1)$ while 0 is considered for unobserved ratings. We randomly selected 60% of the observed ratings for training and 20% for validation set and consider the remaining 20% of the ratings as our test set. To better evaluate the results, we performed a 3-fold-cross validation and reported the average value for our results.

5.4 Baseline Algorithms

The proposed $\mathtt{S^2COR}$ algorithm is compared to the following algorithms:

- **Feature Based Factorized Bilinear Similarity Model (FBS) [22]:** This algorithm uses bilinear model to capture pairwise dependencies between the features.
- **Collaborative User-specific Feature-based Similarity Models (CUFSM):** By using the history of ratings for users, it learns personalized user model across the dataset [10].
- **Regression based Latent Factor Model (RLF):**[2] This method incorporates the features of items in factorization process by transforming the features to the latent space using linear regression [2]. If the learning method is Markov Chain Monte Carlo, we name it RLF-MCMC.
- **Cosine Similarity Based Recommender (CSR):** Using the similarity between features of items, the preference score of a user on an item will be estimated.

[2]The implementation of this method is available in LibFM library [18].

5.5 Robustness to not missing at random ratings

In this section we compare the effect of incorporating the unobserved ratings in our learning in comparison with excluding them from our learning. Most of the methods in the literature ignore the unobserved ratings and train their model only base on observed ratings. By incorporating the unrated items in ranking, our method can limit the bias caused by learning solely based on the observed ratings and consequently deals with the not missing at random issue of ratings. Table 2 shows results of comparing these two scenarios for $\mathtt{S^2COR}$ on ML-IMDB. In order to see the difference between these two scenarios, we considered 70% of the ratings for training and 30% for test to have more ground truth for our testing. Table 2 shows the NDCG@5, 10,15 and 20 for both scenarios and it shows that incorporating the unobserved ratings causes to improve the accuracy of recommendation list. Hence, the NDCG values for top 5, 10, 15 and 20 items improved when unrated items were included as part of the training process.

5.6 Dealing with cold-start items

We now turn to evaluating the effectiveness of $\mathtt{S^2COR}$ for cold-start recommendation. To do so, we randomly selected 60% of the items as our training items and 20% for validation set and considered the remaining 20% of the items as our test set. In this scenario, baseline algorithms that are used for comparison are CSR, FBS, CUFSM and RLF. For the experiments, we used ML-IMDB, Amazon and CiteULike datasets. Table 3 shows the measurement results of applying mentioned algorithms on these datasets. For each test, the parameters' values producing the best ranking on the validation set were selected to be used and reported. As it can be seen from the results in Table 3, the proposed $\mathtt{S^2COR}$ algorithm outperformed all other baseline algorithms and provided a recommendations with higher quality in comparison to other methods. We can also see from the results of Table 3 that for the ML-IMDB dataset, the improvement in terms of REC@10 is significant compared to other datasets. Since the density of this dataset is much higher than other two datasets, this observation indicates that our method is more effective in utilizing side information compared to other methods. These results demonstrate the effectiveness of $\mathtt{S^2COR}$ in comparison with other state-of-the-art algorithms. $\mathtt{S^2COR}$ was able to outperform other state-of-the-art algorithms by considering the missing data and focusing on top of the recommendation list for

Table 2: Results of employing missing ratings versus ignoring them on ML-IMDB. $\lambda = 0.6$ is regularization parameter, $h = 10$ is dimension of latent features, $T = 100$ is the number of iterations.

Algorithm: S^{2}COR	**NDCG@5**	**NDCG@10**	**NDCG@15**	**NDCG@20**
Observed ratings	1.1690	2.2218	2.8362	3.2849
Observed + missing ratings	**1.1794**	**2.2405**	**2.8585**	**3.3096**

cold-start items.

6 Conclusions

In this paper we introduced a semi-supervised collaborative ranking model by leveraging side information about both observed and missing ratings in collaboratively learning the ranking model. In the learned model, unrated items are conservatively pushed after the relevant and before the irrelevant items in the ranked list of items for each individual user. This crucial difference greatly boosts the performance and limits the bias caused by learning only from sparse non-random observed ratings. The proposed algorithm is compared with seven baseline algorithms on three real world datasets that demonstrated the effectiveness of proposed algorithm in addressing cold-start problem and mitigating the data sparsity problem, while being robust to sampling of missing ratings.

References

[1] G. Adomavicius and A. Tuzhilin. 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(6):734–749, 2005.

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. In *SIGKDD*, pages 19–28. ACM, 2009.

[3] S. Agarwal. The infinite push: A new support vector ranking algorithm that directly optimizes accuracy at the absolute top of the list. In *SDM*, pages 839–850. SIAM, 2011.

[4] S. Balakrishnan and S. Chopra. Collaborative ranking. In *ACM WSDM*, pages 143–152. ACM, 2012.

[5] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[6] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, 2010.

[7] K. Christakopoulou and A. Banerjee. Collaborative ranking with a push at the top. In *WWW*, pages 205–215. International World Wide Web Conferences Steering Committee, 2015.

[8] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *ACM RecSys*, pages 39–46. ACM, 2010.

[9] R. Devooght, N. Kourtellis, and A. Mantrach. Dynamic matrix factorization with priors on unknown values. In *ACM SIGKDD*, pages 189–198. ACM, 2015.

[10] A. Elbadrawy and G. Karypis. User-specific feature-based similarity models for top-n recommendation of new items. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):33, 2015.

[11] K. Järvelin and J. Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR*, pages 41–48. ACM, 2000.

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[13] J. Lee, S. Bengio, S. Kim, G. Lebanon, and Y. Singer. Local collaborative ranking. In *Proceedings of the 23rd international conference on World wide web*, pages 85–96. ACM, 2014.

[14] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[15] B. M. Marlin and R. S. Zemel. Collaborative prediction and ranking with non-random missing data. In *RecSys*, pages 5–12. ACM, 2009.

[16] B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney. Collaborative filtering and the missing at random assumption. In *UAI*, pages 267–275, 2007.

Table 3: Results on cold-start items. λ, μ_1 and β are regularization parameters, h is dimension of latent features, l is the number of similarity functions and T is the number of iterations.

	Algorithms	Hyperparameters	DCG@10	REC@10
ML-IMDB	CSR	—	0.1282	0.0525
	RLF	$h = 15$	0.0455	0.0155
	CUFSM	$l = 1, \mu_1 = 0.005$	0.2160	0.0937
	FBS	$\lambda = 0.01, \beta = 0.1, h = 5$	0.2270	0.0964
	S^{2}COR	$\lambda = 0.6, h = 10, T = 200$	**0.2731**	**0.2127**
Amazon	CSR	—	0.0228	0.1205
	RLF	$h = 30$	0.0076	0.0394
	CUFSM	$l = 1, \mu_1 = 0.25$	0.0282	0.1376
	FBS	$\lambda = 0.1, \beta = 1, h = 1$	0.0284	0.1392
	S^{2}COR	$\lambda = 0.6, h = 10, T = 200$	**0.1195**	**0.1683**
CiteULike	CSR	—	0.0684	0.1791
	RLF	$h = 75$	0.0424	0.0874
	CUFSM	$l = 1, \mu_1 = 0.25$	0.0791	0.2017
	FBS	$\lambda = 0.25, \beta = 10, h = 5$	0.0792	0.2026
	S^{2}COR	$\lambda = 0.6, h = 10, T = 200$	**0.0920**	**0.2243**

[17] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys*, pages 21–28. ACM, 2009.

[18] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[19] C. Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *The Journal of Machine Learning Research*, 10:2233–2271, 2009.

[20] M. Saveski and A. Mantrach. Item cold-start recommendations: learning local collective embeddings. In *RecSys*, pages 89–96. ACM, 2014.

[21] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260. ACM, 2002.

[22] M. Sharma, J. Zhou, J. Hu, and G. Karypis. Feature-based factorized bilinear similarity model for cold-start top-n item recommendation. In *SDM*, 2015.

[23] Y. Shi, M. Larson, and A. Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *ACM RecSys*, pages 269–272. ACM, 2010.

[24] Y. Shi, M. Larson, and A. Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):3, 2014.

[25] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *ICDM*, pages 1055–1060. IEEE, 2010.

[26] N. Srebro, J. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2004.

[27] H. Steck. Training and testing of recommender systems on data missing not at random. In *KDD*, pages 713–722. ACM, 2010.

[28] H. Steck. Gaussian ranking by matrix factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 115–122. ACM, 2015.

[29] M. Volkovs and R. S. Zemel. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2012.

[30] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. *NIPS*, 2007.

[31] T. Zhou, H. Shan, A. Banerjee, and G. Sapiro. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *SDM*, volume 12, pages 403–414. SIAM, 2012.

A Scalable People-to-People Hybrid Reciprocal Recommender Using Hidden Markov Models

Ammar Alanazi* Michael Bain†

Abstract

Recommender systems are methods of personalisation that provide users of online services with suggestions for further interaction with those services. Most recommender systems are for product-to-consumer recommendation, suggesting items or products to users, but there is a growing need for reciprocal recommender systems, where the goal is to suggest users to other users of the system with whom they might like to interact. Unlike product-to-consumer recommendation, for reciprocal recommendation to be successful both parties must agree to the interaction. Reciprocal recommendation is needed, for example, in matching users in online social networks for friendship or dating, or matching users to employers in online recruitment. Since online social network websites typically have many millions of users, the problem of finding successful interactions is not a trivial task, and automating the process becomes essential.

Most existing reciprocal recommender systems use either profile similarity or interaction similarity to recommend new matches, assuming that user preferences are static and ignoring temporal aspects of user behaviour. This paper takes a different approach, and addresses the issue of representing user preferences as dynamic. We introduce a new representation for changes in user preferences and use that representation in creating a reciprocal recommender system applied to online dating.

In this paper, we develop a general framework for combining a Hidden Markov model (HMM) content-based reciprocal recommender system with collaborative filtering techniques to create a unified hybrid recommender. Additionally, a new similarity measure is introduced to rank the recommendations generated by this hybrid recommender. Moreover, we propose, design and implement a reciprocal recommender system using the suggested framework and the new similarity measure. Evaluation of this system shows that it generates better recommendations than existing systems in a time-efficient manner.

1 Introduction

Most of the existing work on recommender systems is built on the assumption that users' behaviours are static and do not change over time. More recently, this assumption has begun to be relaxed in work on temporal recommendation [18, 20, 11]. However, to the best of our knowledge no work has addressed the problem of temporal *reciprocal* recommendation, such as occurs in the context of online dating, employment websites, and other people-to-people interactions. Reciprocal recommender systems are necessary when the entity recommended to the active user (the one receiving the recommendation) must consent or reciprocate in some way to being the object of the recommendation; typically this "entity" is another user.

Analysis of real-world data from a dating website on people-to-people interactions [2] shows that people's behaviour and activity levels do change over time, which leads to the conclusion that we need a dynamic model to generate better recommendations.

To capture these temporal changes, in this paper we describe a scalable hybrid Hidden Markov Model [17] reciprocal recommender system that captures how each user's behaviour evolves over time and generates recommendations accordingly. The proposed hybrid recommender combines collaborative filtering techniques with the HMM recommender proposed in [3] to generate recommendations.

In this research, the main concern is the changes in individuals' behaviours rather than changes in the whole population's behaviour (i.e. trends). Moreover, social networks (including dating websites) are different from typical item-user networks because there is no obvious categorisation of the users in these networks. For example, in article recommender systems, the problem can be abstracted to recommending an article category to minimise the number of classes that the model has to predict. While people can be categorised by age, gender, interests ...etc., reducing the recommendations problem in social networks to recommending a category instead of recommending a specific user does not lead to accurate results. Unlike the standard item-to-user recommenders, where the user can just purchase the recommended item, in this type of recommenders the recommendations are people, and these people also have to accept the other party for the interaction to be considered successful.

In this paper we introduce a novel hybrid recommender system that combines the ability of collaborative filtering methods to generate recommendations with the high performance of the predictions of the Hidden Markov model recommender that was proposed in [3], to generate recommendations in reciprocal do-

*King Abdulaziz City for Science and Technology, Computer Research Institute, Riyadh, Saudi Arabia

†University of New South Wales, Faculty of Engineering, Sydney, Australia

mains with high success rates. The Collaborative Filtering Hidden Markov Models Hybrid Recommender ***CFHMM-HR*** was tested on an industrial-scale data that was acquired from a real dating website and the results show that CFHMM-HR outperforms its counterparts.

Moreover, CFHMM-HR is a scalable hybrid recommender that can run in real-world applications. Although CFHMM-HR was only tested on an online-dating dataset, we believe that the same model can be applied in other reciprocal domains after a few changes in the preprocessing level.

The key contribution we present in this paper is a novel hybrid recommender that combines techniques from content-based recommender systems and collaborative filtering recommender systems to generate recommendations in a people-to-people domain and which outperforms the previous recommender systems that were reported on a similar domain. In particular, the proposed hybrid recommender offers the following novel features:

- It presents a novel hybrid recommender that generates recommendations with high success rate in comparison to its counterparts.
- It presents a recommender that can work on industrial sized datasets in a timely manner and that can be deployed online.
- It introduces a new similarity measure that minimises the number of false positives and maximises the number of true positives predicted by the recommender. Consequently, the new similarity measure maximises the success rate of the recommendations generated by the model.

2 Related Work

Although recommender systems have been investigated thoroughly in the literature, research on temporal aspects of the recommendation problem has only attracted attention in the past few years. Most of the time-aware recommenders proposed in the literature [11, 18, 20, 9, 24, 8, 10] are variations of the Matrix Factorisation (MF) model. However, initial experimentations on MF in people-to-people reciprocal recommendation did not work well [21].

The area of people-to-people reciprocal recommenders has attracted little research attention in comparison to traditional item-to-user recommenders. Although some interesting models were proposed in the literature such as [1, 12, 6, 21, 13, 5, 22, 15], to the best of our knowledge no one has addressed the temporal aspect of the problem explicitly.

3 Dataset

The dataset used to test the model is a real-world commercial dataset from a dating website. In the dating domain, there are users who initiate interactions, we call them *senders*, and people who receive interactions, and we call them *recipients*. Senders and recipients can overlap which means a user can be a recipient and a sender at the same time. There are different forms of interactions that can be exchanged such as predefined messages, emails and chats. In this research, we use the predefined messages, we call them *messages*, to train and test our model because this is the first method of communication between users in most cases and depending on the success of these messages, users can further their communications and exchange other forms of interactions.

When a predefined message is sent, the recipient can ignore this message and not reply to it, reply with a positive predefined message or reply with a negative one. We have only considered messages that have replies to them and classify them as positive or negative interactions based on the reply message.

The dataset has over 3 million users and over 80 million interactions exchanged between these users. Therefore, using the whole dataset is not feasible and representative subsets have to be used instead. To generate training and testing data for our model, a time period was randomly selected (e.g. from March 1^{st} to March 15^{th}, 2009) and all active users during this time period were used as the experiment population. Then, for users in the selected population, all their interactions, even interactions outside the selected time period, were obtained and used to build the model. Several populations were generated and average results across these populations will be presented later in this paper.

The average population size is over 195,000 users of which a little over 16,000 were recipients and about 190,000 were senders. These users exchanged over two million messages amongst themselves. Each population was divided into 70% training data and 30% test data.

This dataset was chosen because it is a real-world commercial reciprocal dataset that has temporal dynamics. Although users' life cycles are mostly short in a dating website [2], there are several life-cycle phases to capture and these changes between phases have their effects on the decision of initiating an interaction and the decision of accepting one.

4 Evaluation Metrics

Several measures have been used in literature to measure the performance of recommender systems. Measures such as Root Mean Squared Error (RMSE) and

Mean Absolute Error (MAE) indicate the accuracy of the system's predictions. However, in applications that require generating actual recommendations to users we are rather interested in measuring how many of these predictions will be used, i.e. the usage of the recommendations [19, 4, 7]. Dating is one of these applications.

In this paper we use the following two metrics that measure the usage of the recommendations. The first one is the ***Success Rate***, also referred to as Precision or Confidence, which is defined as the proportion of generated recommendations that are correct. The other metric used to evaluate the model is ***Recall***, or Sensitivity, which is the proportion of successful interactions in the test set that was predicted successfully by the model [16].

More formally, let R be the set of recommendations generated by the model, R_+ be the subset of R that is correct, I_+ be the set of successful interactions in the test data and $Size(S)$ is the size of a set S.

$$SuccessRate = \frac{Size(R_+)}{Size(R)} = \frac{TP}{TP + FP}$$

$$Recall = \frac{Size(R_+)}{Size(I_+)} = \frac{TP}{TP + FN}$$

Generally, recommender systems in online dating applications are required to generate a pre-determined number of recommendations and in this case the most important evaluation metric is Success Rate [19]. Additionally, opening communication channels in online dating domain consumes time and money in most cases. Therefore, the best recommender is the one that does not encourage the user to start an interaction unless it is very likely to succeed and the measure that captures this likelihood is also the *Success Rate*. However, since experiments in this paper are all offline experiments that are performed on historical data, we will present Recall values as well, but the focus of this research is to improve the Success Rate.

Additionally, we will use the ***F-measure***, also called F-factor, which is a weighted average of success rate and recall [16]. The general formula to calculate F-measure is:

$$F_\beta = (1+\beta^2).\frac{SuccessRate.Recall}{(\beta^2.SuccessRate) + Recall} \text{where} \quad \beta > 0$$

When $\beta = 1$, similar weights are given to both success rate and recall. We decided to use $\beta = 0.25$ to put more emphasis on success rate since it is the focus of this research.

5 A Hybrid Recommender Combining Collaborative Filtering and Hidden Markov Models

The recommender that will be described in this section is an extension to the content-based HMM recommender that was presented in [3]. We will first start by outlining the limitations of the HMM recommender followed by the design, implementation and experimental results of CFHMM-HR.

5.1 Limitations of the Content-Based HMM Recommender Although the experimental results of the content-based HMM recommender [3] are promising, there is one main limitation to that model. The only way of actually generating recommendations is to use brute-force to generate all the possible interactions that could occur and then pass them through the model to predict whether they will succeed or fail. However, this is not feasible due to the size of the dataset. Even with a small sample of 1,000 recipients, there is an average of 12,000 senders interacting with these recipients. Consequently, brute-force will produce 12,000,000 interactions to be validated and this will not work in a timely manner in a real-world recommender. Moreover, that small sample is too small to be representative of the whole dataset.

5.2 Design One of content-based recommenders' strengths is that they can be used as filters on recommendations generated by other methods [14]. On the other hand, collaborative filtering recommenders have the ability to generate a list of recommendations by utilising the similarities between the users.

Therefore, to overcome the limitations of the HMM recommender and be able to generate recommendations, we decided to use a collaborative filtering recommender first to generate the recommendations. Then, we test the top N recommendations using the HMM model and filter out the unsuccessful predictions.

The CFHMM-HR model works in the following order (Figure 1):

- Training data is used to train the HMM recommender and used by the collaborative filtering recommender to generate the initial list of recommendations.
- The initial list of recommendations gets validated by the HMM recommender. The output of this step is another list, the second list, of recommendations which is a smaller subset of the initial list.
- The second list of recommendations gets ranked using a combination measure of likelihood and

collaborative filtering similarity and the final list of recommendations is generated.

To represent the messages of the initial list of recommendations as interactions (described in [3]), we assume that each one of these messages are received immediately after the last message of the training data for each user. Formally:

$$I_r = (O_{tr_{k-n+1}}, O_{tr_{k-n+2}}, \ldots, O_{tr_k}, O_r) \quad \text{if} \quad k \geq n-1$$
$$\text{or} \quad I_r = (\phi, \phi, \ldots, O_0, \ldots, O_{tr_{k-1}}, O_{tr_k}, O_r) \quad \text{if} \quad k < n-1$$

Where tr_k is the last message in the training data for each user, r is the recommended message, O_x is the observation vector for the message x and I_x is the message x represented as an interaction.

The model is built so that it can work with any collaborative filtering recommender as long as it generates the initial list of recommendations in a compatible syntax. The HMM recommender that was used in this model is the one described in [3]. As for the collaborative filtering part of the recommender, we experimented with three different models: Basic CF+ [12], SIM-CF [21] and ProCF [6]. These models were selected because SIM-CF [21] is the model that was chosen to be the recommender for the dating website that we obtained its dataset for this research and SIM-CF's main strength is its high recall. ProCF [6] is one of the best performing models, success rate wise, reported on the same dataset. Finally, Basic CF+ [12] is a simple model that implements the basic idea of collaborative filtering in an easy to understand way.

In each experiment, the HMM part of CFHMM-HR receives a list of the top 200 candidates for each user from the CF recommender. The HMM recommender then filters and re-ranks that list to get the top 50.

Since CFHMM-HR receives the initial list of recommendations with their similarity scores from a CF recommender and we have no control over the similarity measure used in that CF recommender, it is not possible to derive a combined similarity measure from first principles. Instead, we derive a new heuristic similarity measure that combines the similarity score received from the CF recommender with the likelihood values generated by the HMM recommender. The new similarity measure was engineered to assure that each element of the measure has the required effect.

The final list of recommendations generated by CFHMM-HR is ranked based on this new heuristic similarity measure we call $HMMSIM$. The formula to calculate $HMMSIM$ is as follows:

$$HMMSIM(m_i) = \Delta_{likelihood}(m_i) + sim(m_i) + \alpha$$

where:

$$\alpha = \begin{cases} \text{large positive constant}, & \text{if} \quad \Delta_{likelihood}(m_i) \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

and:

$$\Delta_{likelihood}(m_i) = likelihood_{success}(m_i) - likelihood_{failure}(m_i)$$

Also, $sim(m_i)$ is the similarity, between m_i's sender and m_i's recipient, generated by the CF recommender for the message m_i

One of the findings we discovered during the experimentations on HMM-CBR is that it is a conservative recommender that tends to under-predict in most cases. However, its predictions are highly reliable. On the other hand, we noticed that the CF recommenders tend to over-predict both true positives and false positives. Therefore, the main effect of adding $\Delta_{likelihood}(m_i)$ in $HMMSIM$ is to minimise the number of false positives predicted by $sim(m_i)$ by demoting their score.

Further, since the predictions of the HMM recommender are highly reliable, we want the messages that are predicted to be successful by it to be on the top of the list. Therefore, we add the large constant α to promote these messages to the top of the list ($\Delta_{likelihood}(m_i) \geq 0$ means m_i is predicted to be successful).

Additionally, we find that $\Delta_{likelihood}(m_i)$ ranges between -15 and +15, while $sim(m_i)$ value can be over 180 for popular users. Which means that in such cases of popular users with high $sim(m_i)$ values, the addition of $\Delta_{likelihood}(m_i)$ will not be sufficient to have the required effect of demoting the overall score $HMMSIM$. Therefore, the addition of α balances the weights of $sim(m_i)$ and $\Delta_{likelihood}(m_i)$ as well.

To further simplify the intuition behind $HMMSIM$, we present this example. Assume we have a user u_a who received two recommendations m_i (user u_x is recommended to u_a) and m_j (user u_y is recommended to u_a). Also assume that u_x is a popular user and u_y is not which means that $sim(m_i)$ is likely to be larger than $sim(m_j)$. Normally, a popular user receives overwhelming messages and his/her recent history would have many unsuccessful messages. In such cases, we believe that the HMM recommender will identify that and predicts m_i to be unsuccessful. Let's also assume that m_j is predicted to be successful and we have the following values:
$sim(m_i) = 90$, $sim(m_j) = 5$
$\Delta_{likelihood}(m_i) = -15$, $\Delta_{likelihood}(m_j) = 15$

From these values, if we calculate the scores before the addition of α we get the following results:

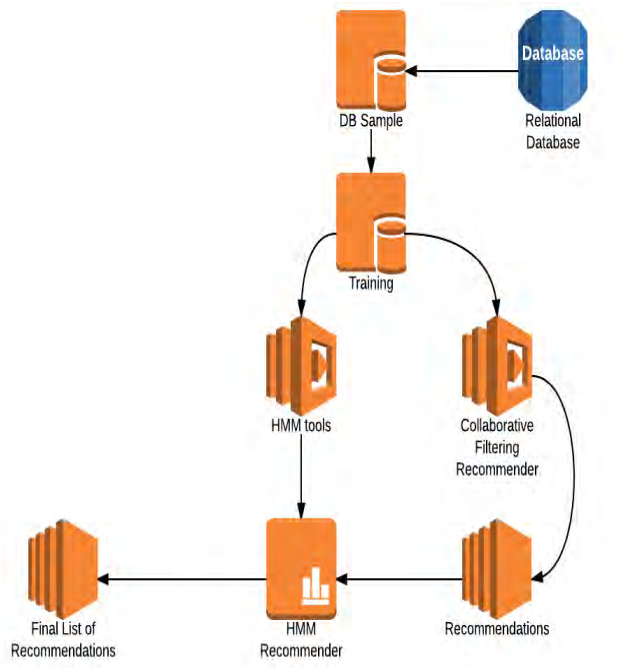


Figure 1: The framework for CFHMM-HR.

$sim(m_i) + \Delta_{likelihood}(m_i) = 75$
$sim(m_j) + \Delta_{likelihood}(m_j) = 20$

Which means that m_i will be ranked higher than m_j even after the demoting. However, if we add $\alpha = 100$ we get:
$HMMSIM(m_i) = 75$, $HMMSIM(m_j) = 120$

This ranks m_j higher than m_i which is the required outcome.

5.3 Implementation The Hidden Markov models Toolkit (HTK) [23] was used to implement the content-based Hidden Markov models part of the recommender and Java was used to implement all the preprocessing and post-processing tools.

6 Experimental Results

This proposed model was also evaluated using the same dataset described in Section 3. Users of the website did not have access to recommendations generated by our model and therefore the model was evaluated using historical data.

6.1 CFHMM-HR with Basic CF+ In this experiment we used the Basic CF+ [12] as the CF recommender. Comparing the accuracy of the recommendations between Basic CF+ alone and CFHMM-HR with Basic CF+ as the CF recommender shows a noticeable overall improvement. The success rate of CFHMM-HR for top 50 recommendations is 15.5% more than the success rate of Basic CF+ and for the top 10 recommendations it is 18.2% more. The recall of the top 50 recommendations is 0.9% more in CFHMM-HR while the recall of the top 10 recommendations is the same in both cases. Please refer to Table 1 for detailed results.

Table 1: A comparison between the results of Basic CF+ and the result of CFHMM-HR ($n = 5$).

Basic CF+			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.252	0.034	0.183
top 20	0.235	0.057	0.199
top 30	0.226	0.076	0.202
top 40	0.219	0.092	0.203
top 50	0.213	0.105	0.201

CFHMM-HR			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.434	0.034	0.256
top 20	0.406	0.060	0.303
top 30	0.388	0.082	0.318
top 40	0.376	0.099	0.323
top 50	0.368	0.114	0.325

6.2 CFHMM-HR with SIM-CF This section presents the experimental evaluation of CFHMM-HR using SIM-CF [21] as the CF recommender. There is an improvement in the success rate and a slight loss of recall. In the top 50 recommendations, CFHMM-HR improves the success rate by 5.7% while losing 0.6% recall. Similarly, in the top 10 recommendations there is an improvement of 5.1% in success rate and a setback of 2.9% in recall when using the CFHMM-HR instead of using the SIM-CF alone. Using the $F_{0.25}$ score to evaluate the overall performance, CFHMM-HR outperforms SIM-CF in all top ranks. More detailed results are presented in Table 2.

6.3 CFHMM-HR with ProCF Here we present the results of CFHMM-HR with ProCF [6] as the CF part of it. We notice an improvement in success rate and a small drop in recall for the top 30 ranks and below. For the top 50 recommendations, CFHMM-HR has 18.9% more success rate than ProCF alone and 0.5% more recall. For the top 10 recommendations, CFHMM-HR shows a 20.2% increase in success rate and a 0.3% decrease in recall. Using the $F_{0.25}$ score to evaluate the overall performance, CFHMM-HR outperforms ProCF in all top ranks. Please refer to Table 3 for more results.

Table 3: A comparison between the results of ProCF and the result of CFHMM-HR ($n = 5$).

ProCF				CFHMM-HR			
Rank	Success Rate	Recall	$F_{0.25}$ score	Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.485	0.032	0.266	top 10	0.687	0.029	0.294
top 20	0.471	0.048	0.310	top 20	0.667	0.042	0.354
top 30	0.455	0.059	0.326	top 30	0.653	0.058	0.407
top 40	0.448	0.068	0.337	top 40	0.642	0.070	0.434
top 50	0.440	0.074	0.341	top 50	0.629	0.079	0.447

Table 2: A comparison between the results of SIM-CF and the result of CFHMM-HR ($n = 5$).

SIM-CF			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.296	0.107	0.268
top 20	0.279	0.163	0.268
top 30	0.265	0.202	0.261
top 40	0.256	0.234	0.255
top 50	0.250	0.260	0.250

CFHMM-HR			
Rank	Success Rate	Recall	$F_{0.25}$ score
top 10	0.347	0.078	0.289
top 20	0.341	0.146	0.316
top 30	0.327	0.191	0.314
top 40	0.314	0.223	0.307
top 50	0.307	0.254	0.303

7 Analysis of Run-Time

Although the main focus of this research is to find a recommendation model that identifies temporal changes and utilises them to better understands user behaviour, there is another aim that is of similar importance. The other aim is that the final model is scalable, time-efficient and can be deployed in a real-world application.

To demonstrate the efficiency of CFHMM-HR, we have calculated the run-time for every step the recommender performs over a large number of experiments. These experiments were run on a Windows Server 2008 machine with the following specifications: Intel Xeon CPU, 2.80 GHz, 4 processors, 32 GB RAM.

The typical scenario for running CFHMM-HR online is as follows:

1. Train the HMM part of CFHMM-HR on a large training data sample. This step can be performed offline and there is no need for frequent re-training. We believe that repeating this step once every month would suffice.

2. Train the CF part of CFHMM-HR overnight. This step needs to be repeated more frequently.

3. Generate the initial list of recommendations from the CF recommender.

4. Test, filter and re-rank the initial list of recommendations using the latest trained HMM recommender. This step needs to be performed overnight and repeated more frequently as well.

5. Send the final list of recommendations to users.

The run-time for training the HMM recommender is shown in Table 4. The average time it takes per message is 106 milliseconds and we suggest using a sufficiently large representative training sample (1-2 million messages). 94% of the time taken in training is spent by the HTK tools that were used to implement the HMM recommender and the remaining 6% is spent by the Java software tool. Moreover, of the time spent by the Java software tool, 86% was spent in input/output operations to process the HTK formatted files. We cannot estimate the proportion of time spent in input/output operation by the HTK tools but we believe it is similar to the Java software tool. The HTK toolkit requires the data to be stored in specially formatted files on the hard disk drive (HDD) which adds a time overhead for opening these files and reading the data and in this case the overhead consumes about 80% of the total time. To deploy CFHMM-HR on an online application, we recommend developing alternative HMM software tools that read the data directly from the DBMS or from memory to eliminate the time overhead.

The run-time of the CF part of CFHMM-HR varies depending on the algorithm used. The three algorithms that were used in this research have been tested in an online trial and they run efficiently [21].

Testing the initial recommendation list by the HMM recommender run-time is presented in Table 5. It takes an average of 8.8 milliseconds to test one message of which 20% is spent by the HTK tools and the remaining 80% is spent by the Java software tool. About 88% of the Java software tool time is spent on input/output operations to generate the HTK formatted files. This

Table 4: Average run-time for training the HMM recommender (per message).

	Total	HTK Tools	Java Software Tool		
			DB Operations	I/O Operations	Other
Time (msec)	106.131	99.259	0.969	5.894	0.009
Percentage		93.52%	0.91%	5.55%	0.01%

leads us to the same conclusion that developing alternative HMM software tools that read the data directly from the DBMS or from memory would minimise the run-time significantly.

8 Discussion

CFHMM-HR is a general framework for combining CF techniques with HMM-CBR in one unified hybrid recommender. Such a combination overcomes one of the main shortcomings of the HMM recommender [3] which is its inability to generate actual recommendations. It also improves the success rate of the CF recommenders considerably. However, the improvement of CFHMM-HR depends on the initial list of recommendations generated by the CF recommender. For example, the best success rate for CFHMM-HR is when it is combined with ProCF and that is because of the three CF methods we combined with HMM-CBR, ProCF is the one with the highest success rate. On the other hand, the best recall for CFHMM-HR is when it is combined with SIM-CF because it is the model with the highest recall. Nevertheless, in all experimentations performed in this research, CFHMM-HR outperforms the use of the CF recommender alone in success rate.

However, while it is possible to deploy CFHMM-HR in an online application in its current setup, we believe that this is not the ideal setup and more optimisation is needed. Mainly, we suggest developing alternative HMM software tools instead of using the HTK toolkit. Because HTK is designed to read data from files stored on HDD, a large proportion of the run-time of CFHMM-HR is spent on writing and reading these files. We estimate that the time overhead caused by the I/O operations is about 80% of the reported run-time and we recommend developing HMM tools that read the data directly from the DBMS or from memory.

In the domain of online dating the interaction vectors of users tend to be sparse and short. Initial experimentations of collaborative filtering methods that are popular with researchers, such as matrix factorization MF, did not work well [21]. A possible reason for that is that in the domain of online dating the interaction matrix is Boolean (positive or negative) and in such cases applying matrix factorisation is harder than applying it on a numerical matrix (ratings). Therefore, we compared our model to other models that are known to work on such data.

9 Conclusion and Future Work

In this paper we presented a hybrid model for people-to-people recommendations using HMM that can capture the temporal changes of users' behaviours and generates better personalised recommendations based on this. Evaluating this model using a commercial dataset for a dating website shows a significant improvement in the success rate of recommendations.

The model combines a HMM content-based recommender and a collaborative filtering algorithm to generate recommendations. In future, we plan on using other dynamic models to represent the recommender such as coupled Hidden Markov models or the Collaborative Kalman Filtering model [20]. Moreover, in CFHMM-HR, the recommendations are generated by a reciprocal CF method and then re-ranked using the temporal HMM-CBR. Incorporating the temporal dynamics in the CF part of the recommender as well would be an interesting extension to our model.

References

[1] J. Akehurst, I. Koprinska, K. Yacef, L. Pizzato, J. Kay, and T. Rej. CCR: A content-collaborative reciprocal recommender for online dating. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence - Volume 3*, IJCAI'11, pages 2199–2204. AAAI Press, 2011.

[2] A. Alanazi and M. Bain. Ranking interaction-based collaborative filtering recommendations using temporal features in online dating. In K. Soliman, editor, *Innovation and Sustainable Competitive Advantage: From Regional Development to World Economies*, volume 1-5, pages 450–457. Int Business Information Management Assoc-IBIMA, 2012. 18th IBIMA Conference, Istanbul, TURKEY, MAY 09-10, 2012.

[3] A. Alanazi and M. Bain. A people-to-people content-based reciprocal recommender using Hidden Markov models. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 303–306, New York, NY, USA, 2013. ACM.

[4] A. Bellogin, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: An algorithmic comparison. In *Proceedings of the 5th ACM*

Table 5: Average run-time for HMM recommender testing (per message).

	Total	HTK Tools	Java Software Tool		
			DB Operations	I/O Operations	Other
Time (msec)	8.787	1.800	0.761	6.213	0.013
Percentage		20.49%	8.66%	70.71%	0.14%

Conference on Recommender Systems, RecSys '11, pages 333–336, New York, NY, USA, 2011. ACM.

[5] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Kim, P. Compton, and A. Mahidadia. Collaborative filtering for people to people recommendation in social networks. In J. Li, editor, *AI 2010: Advances in Artificial Intelligence*, volume 6464 of *Lecture Notes in Computer Science*, pages 476–485. Springer Berlin Heidelberg, 2011.

[6] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Kim, P. Compton, and A. Mahidadia. ProCF: Probabilistic collaborative filtering for reciprocal recommendation. In J. Pei, V. Tseng, L. Cao, H. Motoda, and G. Xu, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7819 of *Lecture Notes in Computer Science*, pages 1–12. Springer Berlin Heidelberg, 2013.

[7] P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1-2):67–119, Feb. 2014.

[8] F. C. T. Chua, R. J. Oentaryo, and E. Lim. Modeling temporal adoptions using dynamic matrix factorization. In *IEEE 13th International Conference on Data Mining*, number Dallas,, pages 91–100, TX, USA, December 2013.

[9] S. Gultekin and J. Paisley. A collaborative kalman filter for time-evolving dyadic processes. In *Proceedings of the IEEE International Conference on Data Mining*, ICDM '14, pages 140–149, Washington, DC, USA, 2014. IEEE Computer Society.

[10] B. Ju, Y. Qian, M. Ye, R. Ni, and C. Zhu. Using dynamic multi-task non-negative matrix factorization to detect the evolution of user preferences in collaborative filtering. *PLoS ONE*, 10(8):e0135090, 08 2015.

[11] Y. Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53(4):89–97, April 2010.

[12] A. Krzywicki, W. Wobcke, X. Cai, A. Mahidadia, M. Bain, P. Compton, and Y. Kim. Interaction-based collaborative filtering methods for recommendation in online dating. In L. Chen, P. Triantafillou, and T. Suel, editors, *Web Information Systems Engineering, AI WISE 2010*, volume 6488 of *Lecture Notes in Computer Science*, pages 342–356. Springer Berlin / Heidelberg, 2010.

[13] L. Li and T. Li. MEET: A generalized framework for reciprocal recommender systems. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 35–44, New York, NY, USA, 2012. ACM.

[14] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer Berlin Heidelberg, 2007.

[15] L. Pizzato, T. Rej, T. Chung, I. Koprinska, and J. Kay. RECON: a reciprocal recommender for online dating. In *Proceedings of the 4th ACM conference on Recommender systems*, pages 207–214. ACM, 2010.

[16] D. M. W. Powers. Evaluation: from precision, recall and f-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, 2(1):37–63, 2011.

[17] L. Rabiner. A tutorial on Hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

[18] N. Sahoo, P. V. Singh, and T. Mukhopadhyay. A Hidden Markov model for collaborative filtering. *MIS Q.*, 36(4):1329–1356, Dec. 2012.

[19] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer US, 2011.

[20] J. Sun, D. Parthasarathy, and K. Varshney. Collaborative kalman filtering for dynamic matrix factorization. *IEEE Transactions on Signal Processing*, 62(14):3499–3509, July 2014.

[21] W. Wobcke, A. Krzywicki, Y. S. Kim, X. Cai, M. Bain, P. Compton, and A. Mahidadia. A deployed people-to-people recommender system in online dating. *AI MAGAZINE*, 36(3):5–18, January 2015.

[22] P. Xia, B. Liu, Y. Sun, and C. Chen. Reciprocal recommendation system for online dating. In *Proceedings of 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Paris, France, 2015.

[23] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. A. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The HTK Book*. Cambridge University Engineering Department, March 2009.

[24] C. Zhang, K. Wang, H. Yu, J. Sun, and E.-P. Lim. Latent factor transition for dynamic collaborative filtering. In *Proceedings of the SIAM International Conference on Data Mining*, pages 452–460. Citeseer, 2014.

Modeling Trust for Rating Prediction in Recommender Systems

Anahita Davoudi* Mainak Chatterjee†

Abstract

Traditional recommender systems usually ignore the social interactions between users in a social network and assume that users are independent and identically distributed. This assumption hinders the users to have access to personalized recommendations based on their circle of trusted friends. To model the recommender systems more accurately and realistically, we propose a social trust model and use the probabilistic matrix factorization method to predict user rating for products based on user-item rating matrix. The effect of users friends tastes is modeled using a real-valued trust which is defined based on importance and similarity between users. Similarity is modeled using a rating-based (Vector Space Similarity algorithm) and connection-based methods; centrality is quantified using degree and eigen-vector centralities. To validate the proposed method, rating estimation is performed on the Epinions dataset. Experiments show that our method provides better prediction when using trust relationship based on centrality and similarity rather than using the binary values. Also, degree centrality is shown to be more effective compared to the eigen-vector centrality. In addition, trust model using connection-based similarity is observed to have better performance compared to the ones that use rating-based similarity.

1 Introduction

Recommender systems help users with item selection and purchasing decisions based on users' tastes and preferences using a variety of information gathering techniques. Such information is gathered either explicitly by mining user's ratings, or implicitly by monitoring user's behavior. These systems offer a personalized experience based on social interactions or user preferences are considered as a fantastic opportunity for retailers in e-commerce businesses. Many recommendation techniques have been studied [12, 21, 22] and have been well adapted to commercial websites such as Amazon, Netflix, etc. Such commercial websites offer a vast number of products for users with different tastes.

An approach which has not received much attention is how to apply the social relationship of users as valuable source of information. Traditional recommender systems assume that users are independent and identically distributed which results in ignoring the social interactions and trust relationships between users. However, user's social relationships play an important role in the behavior of users regarding future ratings. Since most of the similarities within a network are caused by the influence and interactions of its users, it is reasonable to develop a social recommender system based on the user connections and interactions. Social recommender systems focus on easing information and interaction burden by applying different methods that present the most relevant information to the users. But retailing platforms usually do not consider social factors such as relationships and trust among the users and the power of social influence is not exploited. On the other hand, social networking platforms generally do not consider online shopping related factors such as purchase history and product rating.

In addition to social connections, trust relationships also influence one's decisions and ought to be considered for recommendations [3]. In a social network, trust relationships and social relationships are two different concepts. Two socially connected users would not necessary trust each other. Also, multiple connections of a user would not have equal impact on user's opinions and decisions. In addition to trust relationships, users with similar taste in purchasing would show similar behavior when rating a product as well.

In this paper, we combine the features of social networks and e-commerce platforms to design a social recommender mechanism to increase the prediction accuracy of product recommendations in e-commerce by considering the factors of similarity, user importance in the network, and social trust relationships. The proposed model could be practically applied to new emerging social commerce platforms. We argue that users are influenced by social interactions, in particular, by the set of trusted friends and their respective importance. To that end, we combine social trust connections and user-item matrix to predict the rating that a user would assign to a product. We use matrix factorization to factor user-item rating matrix into two low-dimensional matrices consisting of user latent matrix and item latent matrix. For the social connections, we consider both user importance and user similarity to build the social trust model between users. We use vector space similarity to obtain the similarity between users. Using degree and eigen-vector centrality, we quantify the importance of users in the network. We use a linear combination of similarity and centrality to model the trust parameter between users. The proposed method captures the balance between user taste and her friends' taste and adjusts the share of centrality and similarity in the trust values using two parameters. The low-dimensional latent user-specific and item-specific matrices are estimated by performing gradient descent on the objective function. We use a dataset from Epinions to validate the proposed model. We estimate the accuracy of the proposed method in terms of the mean absolute error by comparing the predicted and the actual user

*Department of Computer Science, University of Central Florida
†Department of Computer Science, University of Central Florida

ratings of products. Results reveal that there is a high correlation between the predicted and the actual ratings. The proposed method is also compared using binary trust values as well as considering the eigen-vector and degree centrality. In summary, our experiment results show that the proposed model could enhance recommendation accuracy.

2 Related Work

Let us discuss the various aspects such as trust, similarity, and user preference that that are relevant for this paper.

2.1 Recommender Systems Different types of recommender techniques has been developed: collaborative filtering, content-based or hybrid [9]. Content-based systems use items' characteristics and the ratings that users have given to generate recommendations. However, this approach has a critical problem: when collecting or providing insufficient information, recommender systems tend to fail [1]. Collaborative systems identify similar users and analyze their preferences to generate recommendations. In [4] the users' purchase patterns are derived by sequential pattern analysis to collaboratively recommend items to the users. There are many studies of the combination of content-based and collaborative-based systems [14, 26]. In [26], a hybrid news recommender system in which the recommendation is made based on analyzed users' preferences and computed news similarities is developed. User-based and item-based approaches are combined in [14] to build a hybrid recommendation of movies in P2P networks.

Collaborative filtering methods are divided into three further categories of memory-based, model-based and hybrid of both. Memory-based methods utilize users' past behavior and recommend products that other users with similar interests have selected in the past [22]. They have been widely used in commercial recommender systems [13, 20]. Memory-based algorithms are either user-based [2, 8] or item-based [13, 22]. User-based algorithms predict rating given by a user to an item based on the ratings by similar users, whereas, item-based algorithms estimate the rating based on the ratings of similar items previously chosen by the user. These methods find similar users [2, 8] or similar items [5, 13, 22] for providing accurate predictions. These methods use Pearson Correlation Coefficient (PCC) algorithm [20] or Vector Space Similarity (VSS) algorithm [2] to compute the similarity for finding the similar users or items. Methods used in traditional recommender systems are mostly based on user-item rating matrix. These algorithms usually fail to find similar users since density of ratings in user-item rating matrix is often less than 1 percent [13]. Model-based methods utilize available data to train a predefined model for rating prediction. Some of these methods are: clustering model [11] and the Matrix Factorization model [15]. Model-based approaches can handle problems with limited data using hierarchical clustering to enhance the accuracy of the prediction [11]. Matrix factorization is another model-based method which factorizes the user-item rating matrix using low-rank representation [21]. Although model-based methods mitigate the sparsity problem, handling users who have never rated any item is a challenging problem in both memory-based and model-based approaches [21].

2.2 Trust Models Online communities allow users to easily express their personal preferences, such as the users they trust and the products/services they are interested in. Trust means a subjective expectation that an agent has about another's future behavior based on the interaction history of their encounters [18]. Trust is a construct [25] that has a significant impact on users' online purchasing behavior. Doing business with people we have never met before requires a great deal of trust, especially when the transaction is executed online without any physical interaction. Therefore, trust plays a critical role in e-commerce behavior. Many trust-based models have been introduced such as TrustWalker [10] which is a combination of both trust-based and item-based recommendations, TidalTrust [6] which finds all the raters with shortest distance from the source user and aggregates their ratings. Also, in [19] trust-aware recommendation is used to increase recommendation accuracy.

2.3 Similarity The similarity between two users has been modeled by similarity measures such as Vector Space Similarity (VSS) and Pearson Correlation Coefficient (PCC). Both have been incorporated in social recommender systems [2]. Based on the similarity concept, the trust relations are bidirectional and equal in both directions. However, this is not true in real world relationships where trust relationships are non-transitive. Also, a user trust relationship value is affected by the importance of that user. Users usually tend to follow an important friend regardless of the similarity between them. The trust relationship enforced by the importance of user is asymmetric since every user have their unique importance.

2.4 User Preference Model To provide personalized recommendation, there are two ways to capture users' preferences [7]: implicit and explicit. The implicit method gathers users' behavior to obtain their preferences [4]. Matrix factorization models built in [12] use implicit feedback from system. The explicit method filters and analyzes interactions and feedback to obtain users' specification [23, 26]. In [15] a user-item matrix is considered with users' social trust graph to build a latent low-dimensional matrix for providing a better recommendation. Users opinion is modeled based on her own and her friends' opinions which reflect real life social interactions [16]. In addition, using trust values in recom-

mender systems would help to predict the behavior of those users who have rated fewer products.

Here, we capture the effect of users' similarity in trust values. We also argue that the importance of a user must be taken into consideration for finding the trust values for predicting the rating of products.

3 System Model

We consider a social recommender system for a social network that is represented as a weighted directed graph of users where edges represent the social trust relationship between users. Users rate items (products) on a scale 1 to 5. The adjacency matrix $A_{N\times N}$ represents the social connections between users. Also user-item rating matrix shows the rating given by each user to each item. The user-item rating matrix $R_{M\times N}$ represents the ratings that each user assigns to each item. The existence of a social connection between two users would not necessarily reflect their level of trust in each other. The method presented here is based on the assumption that the trust between users is impacted by similarity and importance of users.

Problem Statement: In a given recommender system, how can we predict the rating that user i would assign to product j, when the social relationship graph and the user-item rating matrix are given.

3.1 Similarity Enforced Trust

3.1.1 Rating Similarity There are several users' characteristics that affect the value of trust between users. Similarity between users is one of the most important ones since two users with the same taste are more likely to trust each other. The effect of similarity has been incorporated in social recommender systems for predicting user rating. Vector Space Similarity (VSS) and Pearson Correlation Coefficient (PCC) [2] are the two most popular methods used for similarity estimation. Here we apply the VSS algorithm to identify the similarity between users utilizes the common items that have been rated by both users i and f to compute similarity which is given by:

$$Sim(i,f) = \frac{\sum_{j\in I(i)\cap I(f)} R_{i,j}\cdot R_{f,j}}{\sqrt{\sum_{j\in I(i)\cap I(f)} R_{i,j}^2}\cdot\sqrt{\sum_{j\in I(i)\cap I(f)} R_{f,j}^2}} \tag{3.1}$$

where j is an item that both users i and f have rated and $R_{i,j}$ is the rating that user i assigned to item j. $I(i)$ represents the set of items rated by user i. VSS is defined in $[0,1]$; larger value implies more similarity between user i and user f. The trust values enforced by similarity can be modeled by weighted average rating of the users using the similarity scores as the weights. Consequently, a connection with high similarity will have more impact on the user's rating.

3.1.2 Connection Similarity The similarity between two users can also be determined by the connection between these two users. The similarity between two users can be measured by the mutual connection they have in common. This can be done using the each user list of connections. For each edge we get the list of connections for both users and then list of mutual connection on both sides. The larger the value would be, it could be an indication of users having more similarity which shows that their connection is more valid in shaping the trust. The list of friends for each user i is defined $F(i)$. The proportion of mutual friends to list of friends for the starting node of relationship is defined as follows:

$$Sim(i,f) = \frac{F(i)\cap F(f)}{F(i)} \tag{3.2}$$

3.2 Centrality Enforced Trust

Although similarity is a major driving force for trust between users, there are other aspects as well. A user with high importance (i.e., high impact) is more likely to be followed by her friends regardless of their similarities. This aspect of trust relationship is modeled by considering the importance of users. The importance of the users in a social network can be quantified using centrality measures such as degree centrality, betweenness, closeness, eigen-vector centrality and pagerank [17]. To obtain the importance of users, we use degree centrality and eigen-vector centrality.

Degree centrality is the simplest centrality measure. It shows the degree of a node, representing how many nodes are connected to it. Eigen-vector centrality gives each node a value which is proportional to the sum of values of its neighbors. Eigen-vector centrality has a property: it can be large either because a node has many neighbors or because it has important neighbors (or both). Pagerank and Katz are similar to eigen-vector centrality except that they add a small free centrality value to each node. Closeness measures the mean distance from a node to other node. Betweenness centrality measures the extent to which a node lies on paths between other nodes. We choose eigen-vector and degree centrality since they consider the connections and also the importance of each connection. Other measure either gives free initial centrality or capturing the path and distances between nodes which we are not interested in.

Degree centrality is used as the basic indication of a user's importance which can be defined as the *number of connections*. In our case, it is the number of incoming edges (in-degree) in the social graph. We define the degree centrality C_l of a user l as:

$$C_l = \sum_{\forall m, l\neq m} A_{l,m} \tag{3.3}$$

where $A_{l,m}$ is the element of the adjacency matrix which represents the connection between user l and user m. Thus, with all connections treated equally, a user with more incoming edges has higher importance in the network.

Eigen-vector centrality of user l at time t is the defined as *sum of the centrality* of all connections of l which is given as:

$$C_l(t) = \sum_{\forall m} A_{l,m}(t) \times C_l(t-1) \tag{3.4}$$

where $C_l(t-1)$ is the centrality of user l at time $t-1$. In contrast to the degree centrality, the eigen-vector centrality considers both the number of incoming edges and also the centrality of the neighboring users. The eigen-vector centrality is computed iteratively by setting all initial values to 1 i.e., $C_l(0) = 1$ for all user l.

3.3 Combined Similarity and Centrality Trust We use a linear combination of similarity and centrality to model the trust of user i in user k as:

$$\Gamma_{i,k} = \beta \frac{Sim(i,k)}{\sum_{l \in \mathcal{T}(i)} Sim(i,l)} + (1-\beta) \frac{C_k}{\sum_{l \in \mathcal{T}(i)} C_l} \tag{3.5}$$

Here, β is the parameter that defines the contribution of similarity and centrality to the overall trust. $\beta = 0$ implies purely centrality enforced trust while $\beta = 1$ refers pure similarity-based trust values. $\mathcal{T}(i)$ refers to the set of trusted friends of user i. C_k refers to the centrality (i.e., measured using either degree or eigen-vector centrality) of user k.

4 Trust Model for Matrix Factorization

Matrix factorization has been widely used to develop social recommender systems [12, 15, 16, 21]. Generally, matrix factorization helps to estimate either the user-item rating or user-trust matrix [16] using low-dimensional representative latent matrices. Here matrix factorization for social recommendation proposed by [16] is employed to examine the performance of the proposed trust relationship. The user-item rating matrix is factorized to learn two $l-$dimensional feature representation of users U and items V.

The user-item rating matrix R consists of m users and n items with rating values in range $[0, 1]$. U_i and V_j represent the $l-$dimensional user-specific and item-specific feature vectors of user i and item j. The conditional distribution for R, given Γ, U, V and σ_R^2 is defined as [15, 21]:

$$p(R|\Gamma, U, V, \sigma_R^2) = \prod_{i=1}^{m} \prod_{j=1}^{n} [\mathcal{N}(R_{ij} | g(\sum_{k \in \mathcal{T}(i)} \Gamma_{ik} U_k^T V_j), \sigma_\Gamma^2)]^{I_{ij}^R} \tag{4.6}$$

where $\mathcal{N}(R_{i,j}|\mu, \sigma_\Gamma^2)$ is probability density function of the Gaussian distribution with mean μ and variance σ_Γ^2. Here, Γ is the proposed trust parameter given by Eq. (3.5), $\Gamma_{i,k}$ is the trust value between users i and k. $R_{i,j}$ is the rating given to item j by user i, and σ_R^2 is the rating variance. I_{ij}^R is an indicator function representing whether user i rated item j. Based on the Bayesian inference and assuming Γ is independent of U and V, the conditional probability of U and V, given R, Γ, σ_R^2 , σ_U^2, and σ_V^2, is defined as:

$$p(U, V|R, \Gamma, \sigma_\Gamma^2, \sigma_U^2, \sigma_V^2) = \prod_{i=1}^{m} \prod_{j=1}^{n} [\mathcal{N}(R_{i,j} | g(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T V_j), \sigma_\Gamma^2)]^{I_{i,j}^R} \times \prod_{i=1}^{m} \mathcal{N}(U_i | 0, \sigma_U^2 \mathbf{I}) \times \prod_{i=1}^{m} \mathcal{N}(V_j | 0, \sigma_V^2 \mathbf{I}) \tag{4.7}$$

where σ_U^2 and σ_V^2 are the variance of user and item feature matrices. $\mathbf{I}$ is the identity matrix. The function $g(x) = 1/(1 + exp(-x))$ is a mapping function whose range is within $[0, 1]$. The set $\mathcal{T}(i)$ contains user i's trusted friends. The proposed social recommender system is based on the idea that user's ratings are impacted by her own taste and her immediate friends' tastes. The parameter α is used to balance between these two factors. The term $U_i^T V_j$ represents the estimated taste of user i of item j, while $\sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T V_j$ term reflects her immediate friends' taste, given as the weighted average of their taste using the trust value as weights.

4.1 User-Specific and Item-Specific Matrices In order to find the optimal values of U and V, the log of the posterior distribution given in Eq. (4.7) should be maximized. Equivalently, U and V can be derived by minimizing the sum-of-squared-errors given in the following equation:

$$\mathcal{L}(R, \Gamma, U, V) = \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{i,j}^R (R_{i,j} - g(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T V_j))^2 + \frac{\lambda_U}{2} ||U||_F^2 + \frac{\lambda_V}{2} ||V||_F^2 \tag{4.8}$$

where $||.||_F^2$ is the Frobenius norm. λ_U and λ_V are user and item latent variance ratios.

The gradient decent approach can be used to solve the minimization problem given in Eq. (4.8) for finding U and V. Gradient decent is a local optimization method based on the partial derivative of the objective function with respect to the decision variables (i.e., U and V). The partial derivatives of $\mathcal{L}$ with respect to U and V are given in Eqs. (4.9) and (4.10).

(4.9)

$$\frac{\partial \mathcal{L}}{\partial U_i} = \alpha \sum_{j=1}^{n} I_{i,j}^R g'(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T V_j) V_j \times (g(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T V_j - R_{i,j}) + (1-\alpha) \sum_{p \in \phi(i)} \sum_{j=1}^{n} I_{p,j}^R g'(\alpha U_p^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(p)} \Gamma_{p,k} U_k^T V_j) \times (g(\alpha U_p^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(p)} \Gamma_{p,k} U_k^T V_j) - R_{p,j}) \Gamma_{p,i} V_j + \lambda_U U_i$$

(4.10)

$$\frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^{m} I_{i,j}^R g'(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T V_j) \times (g(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T V_j - R_{i,j}) \times (\alpha U_i + (1-\alpha) \sum_{k \in \mathcal{T}(i)} \Gamma_{i,k} U_k^T) + \lambda_V V_j$$

Here $g'(x)$ is the derivative of logistic function where $g'(x) = exp(x)/(1+exp(x))^2$. $\phi(i)$ is the set of the users who trust user i [16].

5 Simulation Model and Results

In order to test the validity and accuracy of the proposed rate prediction framework, we conduct extensive simulation experiments with data from Epinions [24].

5.1 Dataset Description Epinions is a review and rating website which allows users to rate items by giving an integer number between 1 and 5. The users can also form social connections by adding other users as their trusted friends. The social connections in this dataset are binary values and do not represent the actual trust values. The dataset includes 22166 users and 355754 social connections, leading to 0.0724 percent density in the user social relationship matrix. The total number of items is 296277, with a total of 922267 ratings, which results in a very sparse item-rating matrix with 0.0140 percent density.

As a result, the user-item rating matrix is also relatively sparse. On average, users have 16.05 trusted friends. The maximum number of friends for a user is 1551 and the most trusted user has 2023 other users trusting her.

5.2 Evaluation Metrics Evaluation measures for recommender systems are usually divided into three categories: 1) Predictive Accuracy Measures (such as MAE, RMSE) which evaluate how close the recommender system is in predicting actual rating values, 2) Classification Accuracy Measures (such as Precision, Recall, F1) which measure the frequency with which a recommender system makes correct/incorrect decisions regarding items based on the relevancy of the recommended items, and 3) Rank Accuracy Measures (such as Discounted cumulative gain(DCG) and Mean Average Precision (MAP)) which evaluate the correctness of the ordering of items performed by the recommendation system.

Precision is a measure of exactness and determines the fraction of relevant items retrieved out of all items (e.g., the proportion of recommended movies that are actually good). Recall is a measure of completeness and determines the fraction of relevant items retrieved out of all relevant items (e.g. the proportion of all good movies recommended). The F1 Metric attempts to combine Precision and Recall into a single value for comparison purposes so it may be used to gain a more balanced view of performance. The precision is the fraction of all recommended items that are relevant and recall is the fraction of all relevant items that were recommended. F-measure is a single value combining different facets of accuracy (precision and recall). Ranking accuracy measure ranks all items for user such that higher-ranked recommendations are more likely to be relevant to users. Since our proposed model focus on the error in the rating prediction, we use the metrics in the first category which evaluate the prediction accuracy of the recommender system. The other two categories are used for classification and ranking.

5.3 Predictive Accuracy Measures Let us formally define that error matrix that we would use.

Mean Absolute Error (MAE): This metric measures the average variation in the predicted rating vs. the actual rating. Let $R_{i,j}^{pre}$ be the predicted rating and $R_{i,j}^{act}$ be the actual rating given by the user i to the product j. The MAE is defined as follows:

$$MAE = \frac{\sum_{i,j} |R_{i,j}^{pre} - R_{i,j}^{act}|}{N} \tag{5.11}$$

Mean Squared Error (MSE): This metric punishes big errors more severely and is defined as follows:

$$MSE = \frac{\sum_{i,j} |R_{i,j}^{pre} - R_{i,j}^{act}|^2}{N} \tag{5.12}$$

Root Mean Squared Error (RMSE): This metric is a variant of mean square error and is defined as follows:

$$RMSE = \sqrt{\frac{\sum_{i,j} |R_{i,j}^{pre} - R_{i,j}^{act}|^2}{N}} \tag{5.13}$$

All these metrics measure the accuracy of the actual predictions and are easy to compute efficiently. Moreover, MAE and MAE-based error estimates have well known statistical properties. These characteristics MAE and RMSE good representative of error metrics to analyze the accuracy of the proposed model.

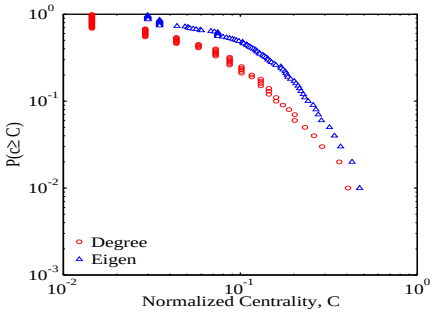


Figure 1: Distribution of centrality

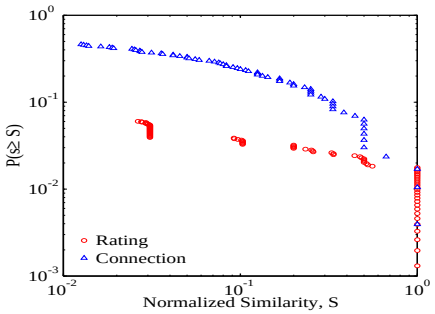


Figure 2: Distribution of similarity

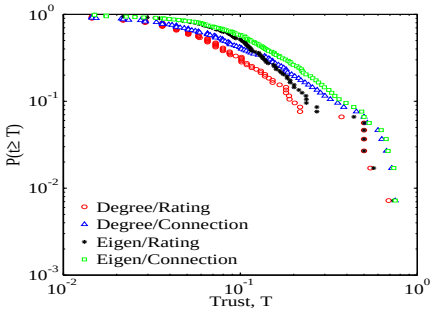


Figure 3: Distribution of trust values

5.4 Simulation Results As mentioned before, the trust relationships between users are defined based on centrality and similarity measures. Fig. 1 shows the distribution of degree and eigen-vector centrality. In Fig. 2, the distribution of rating-based and connection-based similarity are shown. The rating-based similarity has a relatively sparse distribution due to the lack of mutual rated products for two friends in many cases. The trust values are calculated as the weighted summation of centrality of similarity using the weight constant β. Fig. 3 shows the distribution of trust values using $\beta = 0.5$. Based on the different centrality and similarity measures, there are four types of trust values as illustrated in Fig. 3.

The proposed trust model is used to predict users rating based on the discussed matrix factorization technique using 75 percent of the data as the training set. According to Eq. (4.7), a user's opinion about a particular product would be a linear function of her connections' taste and her own taste using a weighting factor α. Smaller values of α is an indication of less impact from neighbors. As previously defined in Eq. (3.5), the trust model is presented as the linear combination of centrality and similarity using the weighting factor β. Higher values of β indicate higher impact of similarity rather than centrality on the trust values. Here, user and item latent variance ratio (λ_U and λ_V) are set to 0.001. The latent size is $L = 4$, $\alpha = 0.4$, and the number of iterations is 300. The performance of the proposed trust model for different values of β in terms of MAE and RMSE is shown in Figs. 4 and 5. Compared to the binary trust model (dashed black lines), the proposed trust model has better performance. Comparing different definitions of trust reveals that degree centrality is the better measure to model trust using eigen-vector centrality. The same is true for connection-based similarity compared to rating-based. An interesting point is that, although including centrality in trust model enhances the recommendation performance compared to the binary trust model, the trust models solely based on similarity (i.e.,$\beta = 1$) show the best performance for the studied network.

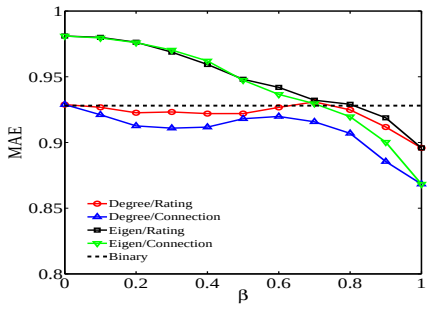


Figure 4: MAE using binary trust and the proposed trust model

The performance of the trust model (the definition which had the best performance in Figs. 4) and 5) for different latent sizes and training percentages are shown in Fig. 6 and Fig. 7. Generally, increasing the latent size as well as using more training data enhance the performance of the recommender system.

The probability distribution of rating estimation error (i.e., estimated rating minus actual rating) for the binary trust and proposed trust model is shown in Fig. 8. Both probability distributions are a little right skewed, implying

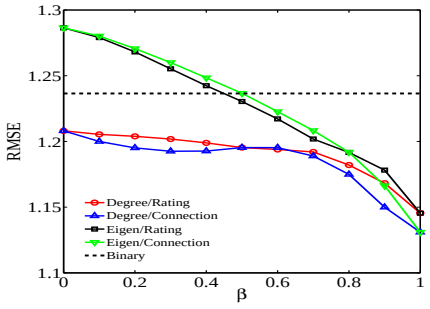


Figure 5: RMSE using binary trust and the proposed trust model

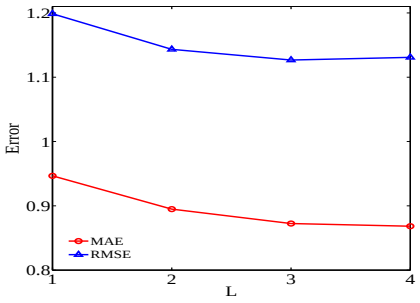


Figure 6: Errors for different latent sizes using degree centrality and connection-based similarity to define trust

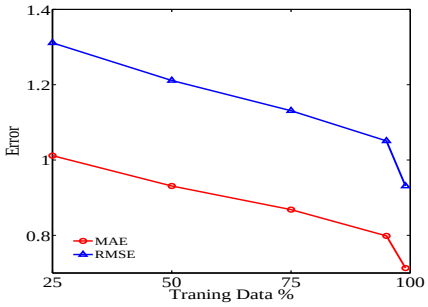


Figure 7: Errors for various training set sizes using degree centrality and connection-based similarity to define

over-estimation. However, the proposed trust model seems to have relatively better performance especially for errors between 0.5 and 2, since it estimates more between 0.5 and 1 and less between 1 and 2 compared to the binary model. The probability distribution of absolute error ratio (i.e., absolute error divided by the actual rating) is shown in Fig. 9. The proposed trust model leads to less error ratio between 1 and 2 and more between 0 and 1 which implies relatively better performance.

In Figs. 10 and 11 the estimated versus actual ratings are shown for the proposed and the binary trust models. The boxes illustrate the lower, upper and inter quartiles, while the redline is the medium. The height of the boxes represents the variation of the estimated rating. Comparing Figs. 10 and 11,

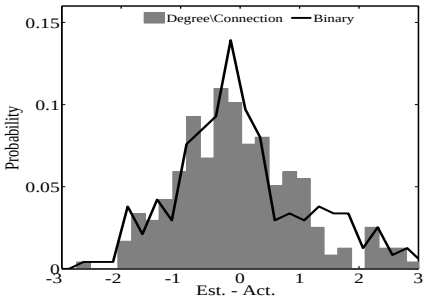


Figure 8: The probability distribution of error for rating estimation using binary trust and the proposed trust model

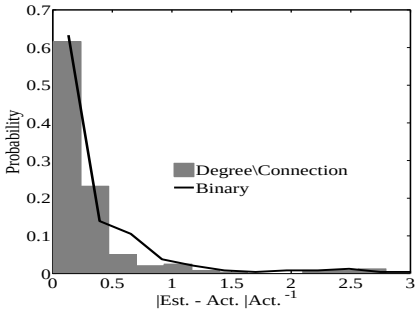


Figure 9: Absolute error ratio for rating estimation using binary trust and the proposed trust model

it is observed that the proposed trust model produces better estimations for low ratings (1 and 2) by slightly undermining the estimation. In addition, for high ratings, the proposed trust model reduces the variation of estimations, i.e., the height of the quartile boxes.

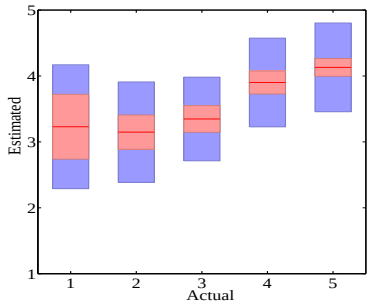


Figure 10: The quartile plot of actual versus estimated rating for the proposed trust model

6 Conclusions

With emerging applications of social networks and considering the role of social interactions in our daily life decisions, extracting information from user's social relationships is becoming a popular method for predicting user's behavior. We capture the trust relationships between users considering users with similar profile and their importance. The main as-

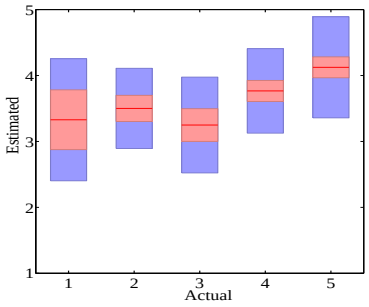


Figure 11: The quartile plot of actual versus estimated rating for the binary model.

sumption is that the users with more similarity would trust each other more; also users with higher importance would be trusted more. Similarity is quantified by a rating-based approach and a connection-based method. The importance is modeled by degree centrality and eigen-vector centrality. We define trust as a linear combination of similarity and centrality using a weighting parameter. The proposed framework is validated using real data from Epinions. Our result indicates that the proposed trust model produces better rating estimation in terms of the mean absolute error (MAE), the root mean squared error (RMSE) and error distribution, compared to the traditional binary trust model which is widely used in recommender systems. Trust enforced by degree centrality shows better performance compared to eigen-vector centrality. The same conclusion is valid for connection-based similarity compared to rating-based. The trust relationships are also observed to be more dependent on the similarity rather than centrality.

References

[1] H.J. Ahn, "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem", Information Sciences, vol. 178, pp. 37-51, 2008.

[2] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering", Uncertainty in Artificial Intelligence, pp. 43–52, 1998.

[3] M. Chau and J. Xu, "Mining communities and their relationships in blogs: a study of online hate groups", Intl. J. of Human Computer Studies, vol. 65, pp. 57-70, 2005.

[4] K. Choi, D. Yoo, G. Kim, Y. Suh, "A hybrid online-product recommendation system: combining implicit rating-based collaborative filtering and sequential pattern analysis", Electronic Commerce Research and Applications, vol. 11, pp. 309-317, 2012.

[5] M. Deshpande and G. Karypis, "Item-Based Top N-Recommendation, ACM Transaction on Information Systems, Vol. 22, pp. 143–177, 2004.

[6] J. Golbeck, "Computing and Applying Trust in Web-based Social Networks", PhD thesis, University of Maryland College Park, 2005.

[7] U. Hanani, B. Shapira, P. Shoval, "Information filtering: overview of issues, research and systems", User Modeling and User-Adapted Interaction, Vol. 11, pp. 203-259, 2001.

[8] J. Herlocker, J. Konstan J., A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering", ACM SIGIR Conference, pp. 230–237 1999.

[9] Z. Huang, W. Chung, H. Chen, "A graph model for E commerce recommender systems", Journal of the American Society for Information Science and Technology, vol. 55, 259-274, 2004.

[10] M. Jamali and M. Ester, "Trustwalker: a random walk model for combining trust-based and item-based recommendation", ACM SIGKDD, pp. 397–406, 2009.

[11] A. Kohrs and B. Merialdo, "Clustering for Collaborative Filtering Applications", In proc. of the International conference on Computational Intelligence for Modeling Control and Automation, 1999.

[12] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques For Recommender Systems", Computer vol. 8, pp. 30–37, 2009.

[13] G. Linden, B. Smith, and J. York,"Amazon.com recommendations: Item-to-item collaborative filtering", IEEE Internet Computing, pp.76–80, 2003.

[14] Z.B. Liu, W.Y. Qu, H.T. Li, and C.S. Xie, "A hybrid collaborative filtering recommendation mechanism for P2P networks", Future Generation Computer Systems, vol. 26, pp. 1409-1417, 2010.

[15] H. Ma, H. Yang, M. R. Lyu and I. King, "SoRec: Social Recommendation Using Probabilistic Matrix Factorization", In proc. of ACM CIKM, pp. 931–940, 2008.

[16] H. Ma, I. King and M. R. Lyu, "Learning to Recommend with Explicit and Implicit Social Relations", ACM Transaction Intelligent Systems Technology, Vol. 2, 2011.

[17] M. Newman, "Networks:an introduction", Oxford University Press, 2010.

[18] L. Mui, M. Mohtashemi and A. Halberstadt, "A computational model of trust and reputation", Proc. of the 35th International Conference on System Science, pp. 280-287, 2002.

[19] J. O'Donovan and B. Smyth, "Trust in Recommender Systems", In Proc. of IUI, pp. 167–174, 2005.

[20] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture collaborative filtering of netnews", ACM CSCW, pp. 175–186, 1994.

[21] R. Salakhutdinov and A. Mnih,"Probabilistic matrix factorization", Advances in Neural Information Processing Systems, Vol. 20, 2008.

[22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms", World Wide Web conference, pp. 285–295, 2001.

[23] J.B. Schafer, J.A. Konstan, J. Riedl, "E-commerce recommendation applications", Data Mining and Knowledge Discovery, vol. 5, pp. 115–153, 2001.

[24] J. Tang. [online]. Available: www.public.asu.edu/ jtang20/datasetcode/truststudy.htm

[25] P. Van Baalen, J. Bloemhof-Ruwaard, and E. van Heck, "Knowledge sharing in an emerging network of practice", European Management Journal, vol. 23, pp. 300-314, 2005.

[26] H. Wen, L. Fang, L. Guan, "A hybrid approach for personalized recommendation of news on the Web", Expert Systems with Applications, vol. 39, pp. 5806–5814, 2012.

Towards Automatic Ranking App Risks via Heterogenous Privacy Indicators

Deguang Kong* Lei Cen† Hongxia Jin‡

Abstract

To inform the users of dangerous levels of mobile apps, assessing privacy risks of mobile apps becomes an urgent task. This paper presents the *first* systematic study on privacy risk ranking of mobile apps via incorporating the heterogeneous privacy indicators (*i.e.*, permission access, user review, developers' description and ads library). We formalize the risk ranking problem as an optimization problem, which uses *"risk propagation"* technique to automatically rank the risks of mobile apps by considering the privacy indicators from different aspects, such that the ranking order can be automatically learned by considering data *manifold* information. Our method can automatically rank the risks of mobile apps given a few number of labeled mobile apps, The exploration on the impacts of different privacy indicators will give insight on which privacy indicators are more closely related to the privacy risks of mobile apps.

1 Introduction

Nowadays, people spend more time on using mobile apps on smart phones and tablets because of the convenience they bring to people's daily life. Personalized service (*such as* targeted advertising, personal recommendation) is possible on mobile devices when users' personal information *such as* contact and location is accessible by mobile apps. However, disclosing personal information to mobile apps could lead to serious privacy issues. Mobile app risk assessment is an effective way to display the risks of mobile apps by summarizing the information that related to the unauthorized access to users' personal information. It makes the risk transparent and warns the users of potential personal information leakage. A risk score strategy has been shown to have a "significant positive effects" [2] for users, which allows the users to better perceive the levels of security risks.

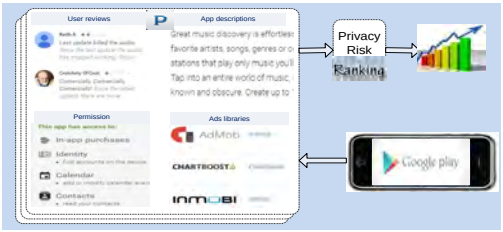


Figure 1: Motivation: Ranking the risks of mobile apps using app meta data such as *description*, *user review* and *permission access*, *ads library*. To automatically ranking *more* different mobile apps, a ranking model is proposed to capture the relations between the ranking score and privacy indicators from different aspects.

In android systems, permissions indicate the resources that the apps can access, and thus can be viewed as a privacy indicator [3]). From users' perspective, the meta data such as users' reviews and developers' descriptions reflect users' perceptions and developers' expectations for the apps, and thus are also correlated [2] with risks of apps.

Given the heterogeneous privacy indicators of mobile apps, a question that naturally follows is: *can we design an automated approach to analyze the risks of mobile apps by utilizing the heterogeneous indicators*? On one hand, there are millions of mobile apps on Google play and labeling the risk score for each mobile app is time consuming and tedious. The proposed method is required to label the risks of mobile apps efficiently and effectively when *only* a very small number of mobile app risk scores are available. On the other hand, the proposed

*Samsung Research America, San Jose, CA, US 95134, doogkong@gmail.com

†Purdue University, West Lafayette, IN 47907, lcen@purdue.edu

‡Samsung Research America, San Jose, CA, US 95134, hongxia@acm.org

approach should utilize the privacy indicators from different aspects, and make a comprehensive assessment. How to combine all the heterogeneous privacy indicators to accurately estimate the risks of apps is under-explored but highly desirable. Recent works, including permission usage pattern mining [1], app permission prediction from meta-data, mobile app recommendation, however, do not essentially solve this problem.

We propose a new approach to rank the privacy risks of mobile apps via *heterogeneous* privacy indicators. The proposed approach only requires a small number of risk score of apps labeled by experts and can automatically predict risk scores of other apps. The predicted scores can be used to improve the credibility of apps in app play store, and make the users be aware of the security risks of mobile apps.

2 Methodology

Assume we have n mobile apps, and each mobile app is abstracted as a data point $\mathbf{x}_i$ denoting the privacy indicators the app carries. We name the features extracted from v-th ($1 \leq v \leq V$) privacy indicator as the v-th aspect feature. In particular, let $\mathbf{x}_i^v \in \Re^{p_v}$ be the v-th view feature (*i.e.*, features extracted from permission, user review, *etc.*) of a mobile app i, p_v be the dimension of feature extracted from the v-th view. Consider all the mobile apps, $\mathbf{X}^v = [\mathbf{x}_1^v, \mathbf{x}_2^v, \cdots, \mathbf{x}_n^v]$, where each data column vector is $\mathbf{x}_i^v \in \Re^{p_v}$.

For the mobile app risk ranking problem, each mobile app is given a scalar value $y_i \in \Re^+$ as the risk score. Without loss of generality, we assume the risk scores for the first $\ell \ll n$ apps are already labeled by security experts, which are denoted as $\{\mathbf{x}_i, y_i\}_{i=1}^{\ell}$. The mobile app risk ranking task is to learn a function f: such that $y_i = f(\mathbf{x}_i)$, which can predict the risk scores y_i for *unlabeled*[1] mobile app $\mathbf{x}_i$ ($\ell + 1 \leq i \leq \ell + u$). The ranking/order of y_i reflects the severity of security levels for different apps. As a number of notations will be used in next sections, we summarize them in Table 1 for clarity.

Let $\mathbf{f} = [f_1, f_2, \cdots, f_n]$ be the desired risk scores[2] corresponding to apps $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n]$, where $f_1 = y_1, f_2 = y_2, \cdots, f_\ell = y_\ell$ for the labeled apps. Taking all the above considerations, we propose to optimize the following objective function with respect to $\mathbf{f}$, *i.e.,*

$$\begin{aligned} \min_{\mathbf{f},\alpha} \quad & \sum_{v=1}^{V} \alpha_v \mathbf{f}^T \tilde{\mathbf{L}}^v \mathbf{f} + \lambda \|\alpha\|_2^2 + \mathbf{f}^T \tilde{\mathbf{L}}^{\mathcal{W}} \mathbf{f} - \mathbf{f}^T \tilde{\mathbf{L}}^{\mathcal{S}} \mathbf{f} \\ (1) s.t. \quad & \alpha^T \mathbf{e} = 1; \;\; \alpha \geq 0; \;\; f_i = y_i \; (1 \leq i \leq \ell); \end{aligned}$$

where V denotes the number of types of privacy indicators extracted from mobile apps. Eq.(1) consists of three parts:

(1) *risk propagation*: term $\sum_{v=1}^{V} \alpha_v \mathbf{f}^T \tilde{\mathbf{L}}^v \mathbf{f}$;

(2) *multi-view privacy indicator weight* α: term $\|\alpha\|_2^2, \alpha^T \mathbf{e} = 1, \alpha \geq 0$;

(3) *constraint* $\mathbf{f}$ *by incorporating prior knowledge*: term $f_i = y_i, \mathbf{f}^T \tilde{\mathbf{L}}^{\mathcal{W}} \mathbf{f} - \mathbf{f}^T \tilde{\mathbf{L}}^{\mathcal{S}} \mathbf{f}$, *etc.*

Table 1: Notations used in the paper

Notation	Description
$\mathbf{x}_i^v$	$\in \Re^{p_v}$, v-th view of feature
$y = [y_1, y_2, \cdots, y_i]$	$y_i \in \Re^+$, risk score for app i
$\ell; u$	# of labeled apps, # of unlabeled apps; $n = \ell + u$
α	$\in \Re^V$, contribution weight for each feature type
$\mathbf{f} = [f_1, f_2, \cdots, f_n]$	$\in \Re^n$, the desired app risk ranking score
W_{ij}^v	the similarity of app i, j in terms of v-th view indicator
$\mathbf{f}^T$	inverse of the vector $\mathbf{f}$

References

[1] M. Frank, B. Dong, A. P. Felt, and D. Song. Mining permission request patterns from android and facebook applications. pages 870–875, 12 2012.

[2] C. S. Gates, J. Chen, N. Li, and R. W. Proctor. Effective risk communication for android apps. *IEEE Trans. Dependable Sec. Comput.*, 11(3):252–265, 2014.

[3] C. S. Gates, N. Li, H. Peng, B. P. Sarma, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy. Generating summary risk scores for mobile applications. *IEEE Trans. Dependable Sec. Comput.*, 11(3):238–251, 2014.

[1] In the paper next, "*unlabeled*" refers to the apps whose risk scores are required to be labeled.

[2] For clarity purpose, we make a distinction between $\mathbf{y}$ and $\mathbf{f}$. Let $\mathbf{f}$ be the desired risk score for mobile app, but $\mathbf{y}$ only has the risk scores for the labeled apps, *i.e.,* $y_i = 0$ if $(\ell + 1) \leq i \leq n$.