Doctoral Dissertation

Recommendation Framework via Matrix Factorization and Translation

Chanyoung Park (박 찬 영)

Department of Computer Science and Engineering

Pohang University of Science and Technology

2018


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

행렬 분해 및 트랜슬레이션 기법 기반의 추천 시스템 프레임워크

Recommendation Framework via Matrix Factorization and Translation

Recommendation Framework via Matrix Factorization and Translation

by

Chanyoung Park
Department of Computer Science and Engineering
Pohang University of Science and Technology

A dissertation submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of philosophy in the Computer Science and Engineering

Pohang, Korea
10. 31. 2018
Approved by
Hwanjo Yu
Academic advisor

Recommendation Framework via Matrix Factorization and Translation

Chanyoung Park

The undersigned have examined this dissertation and hereby certify that it is worthy of acceptance for a doctoral degree from POSTECH

10. 31. 2018

Committee Chair	Hwanjo Yu	
Member	Chi-Hyuck Jun	
Member	Young Myoung Ko	
Member	Minsu Cho	(Seal)
Member	Suha Kwak	(Seal)

DCSE 박 찬 영. Chanyoung Park
20142891
Recommendation Framework via Matrix Factorization and Translation,
행렬 분해 및 트랜슬레이션 기법 기반의 추천 시스템 프레임워크
Department of Computer Science and Engineering , 2018, 112p, Advisor : Hwanjo Yu. Text in English.

ABSTRACT

According to the recent technical report from Amazon.com, estimated 30 percent of their page views were from recommendations, and likewise as for Netflix, more than 80 percent of movies watched on Netflix came through recommendations, and the value of Netflix recommendations is estimated at more than US$1 billion per year. Therefore, research on recommender systems is of great value to players in the industry, and they are striving hard to build successful recommender systems.

Among various recommendation techniques, collaborative filtering (CF) has been shown to be the most successful; it assumes that users who have had similar interests in the past will tend to share similar interests in the future. While CF is a promising direction of recommender system, it fails to achieve a high performance without sufficient amount of user-item interaction records or user feedback; this phenomenon is also known as the data sparsity problem. In this dissertation, we focus on data-centric approaches to overcoming the data sparsity problem prevalent in recommender systems. We postulate that to effectively alleviate the data sparsity problem, understanding characteristics of data related to users and items should be preceded, and that the recommendation models should be

built upon the insights obtained from them. Our data-centric approaches can be categorized into two different streams; **auxiliary data-driven approach** and **algorithmic approach**.

The auxiliary data-driven approach refers to methods that leverage side information related to users and items to make up for the lack of user feedback on items; i.e., to solve the data sparsity problem. As the first approach along this line, we propose a social network-based recommendation method that incorporates users' social network into recommendation by modeling two different roles of users as trusters and trustees while considering the structure of social network. Then, we propose an image-based recommendation method that leverages information hidden in the so-called "also-viewed" products, to also account for the non-visual item aspects overlooked by previous visually-aware recommender systems. Lastly, we propose an implicit feedback-based recommendation method that focuses on dealing with the missing user-item interactions of purchase records by leveraging users' past click records.

The algorithmic approach aims at fully exploiting the hidden properties of data under the algorithmic perspective, whose ultimate goal is to solve the data sparsity problem. As the first approach along this line, we first propose a data sampling scheme called decision boundary focused under-sampling (DBFUS), which under-samples majority class regarding the distance to the decision boundary to retain the classification accuracy, to cope with class imbalance problem among positive and negative instances. Then, we propose a metric learning-based method to discover the intensity and the heterogeneity of user-item relationships embodied in implicit user–item interactions.

We validate the benefit of the above data-centric approaches through extensive experiments. To summarize, this dissertation sheds light on the power of data-centric approaches for solving the data sparsity problem.


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교

Contents

I. Introduction

1.1 Research Motivation

Recently, the importance of recommender systems is increasingly emphasized due to the exploding growth of the amount of data available on the web. According to the recent technical report from Amazon.com, estimated 30 percent of their page views were from recommendations, and likewise as for Netflix, more than 80 percent of movies watched on Netflix came through recommendations, and the value of Netflix recommendations is estimated at more than US$1 billion per year. Therefore, research on recommender systems is of great value to players in the industry, and they are striving hard to build successful recommender systems. Among various recommendation techniques, the most successful approach is collaborative filtering (CF) [1]; it recommends items to a user based on previous ratings of other users whose tastes are similar to the target user. However, this in turn implies that the performance of CF will suffer without a sufficient amount of ratings previously given by users, which is common in reality. This problem, which is also widely known as data sparsity problem, is a long-standing challenge for building successful recommender systems.

To alleviate the data sparsity problem, a plethora of data-centric approaches have been proposed. Data-centric approach refers to methods that are built upon the insights obtained from data related to users and items to compensate for the lack of user feedback. Data-centric approaches can be categorized into two different streams; **auxiliary data-driven approach** and **algorithmic approach**. A vast majority of research focused on incorporating auxiliary information related to users and items, such as user social network [2, 3, 4, 5, 6, 7, 8], item description text [9, 10, 11, 12, 13], item images [14, 15, 16, 17, 18], and temporal dynamics [19, 20, 21, 22, 23]. Apart from these auxiliary data-driven methods, algorithmic approaches that try to resolve the data sparsity issue in the perspective of algorithms have been also proposed [24, 25, 26, 27, 28, 29, 30].

1.2 Research Question

The goal of this dissertation is to propose several novel approaches along each of the above approaches. Here, we pose the following research questions (RQs), and tackle each of them throughout this dissertation. **RQ1** is related to

the auxiliary data-driven approach, **RQ3** is related to the algorithmic approach for data sparsity reduction, and **RQ2** is related to both of the approaches.

RQ1. How can we extract useful information from auxiliary information by using network analysis?

Various types of data can be represented as graphs consisting of nodes and edges. For example, the social relationship between users can be represented as graphs; users as nodes and their relationships as edges. In **Chapter II**, we propose a novel Learning-to-rank-based recommendation method that optimizes the top-k ranking prediction accuracy by additionally considering the social network information. Specifically, it models two different roles of users as trusters and trustees while considering the structural information of the network, both of which have been commonly overlooked by past research.

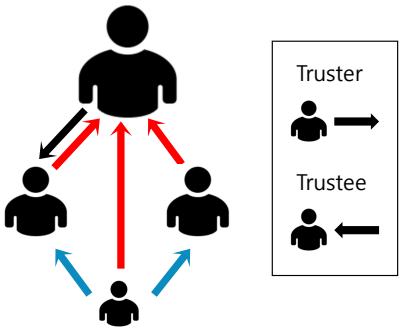


Figure 1.1: Two roles of a user as a truster and a trustee (The larger the user, the more important is the user in the network).

Figure 1.1 shows how the network we aim to model looks like; the arrows denote the direction of the trust and the size of the users represent the their relative importance in the network.

Moreover, in **Chapter III**, we build an item affinity network using so-called "also-viewed" products (Figure 1.2) and extract useful information hidden in the network; "Also-viewed" products reflect various aspects of a given product that have been overlooked by visually-aware recommendation methods proposed in past research; We demonstrate that every product domain has dominant aspects that are more influential to user ratings than others.

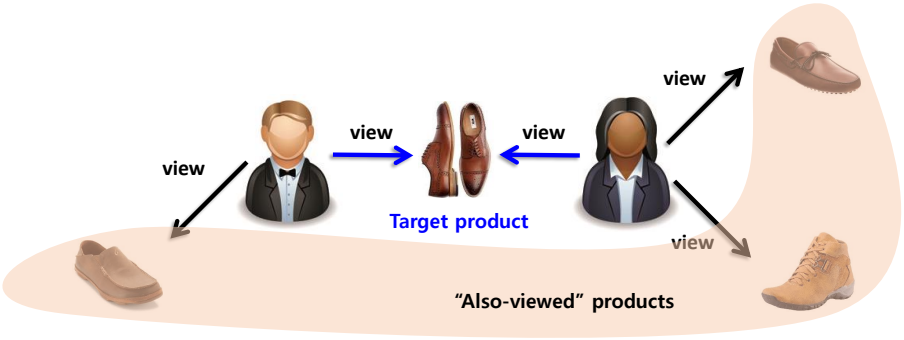


Figure 1.2: Definition of "also-viewed" products.

RQ2. How can we compensate for the lack of negative feedback in implicit feedback data?

While explicit feedback data such as ratings are definite signals for user's preference, the amount of explicit feedback given to the system is not enough to accurately predict user's preference. Therefore, research on leveraging implicit feedback has been widely conducted and is gaining more and more attention. In **Chapter IV**, we focus on dealing with the missing user–item interactions of purchase records, i.e., non-purchased items, and propose to leverage users' past click records, which not only reveal users' broad interests but are also considered as a context in which purchases have been made. Figure 1.3 shows the core idea of our approach; instead of considering non-purchased items equally as negative, we split them by using click records.

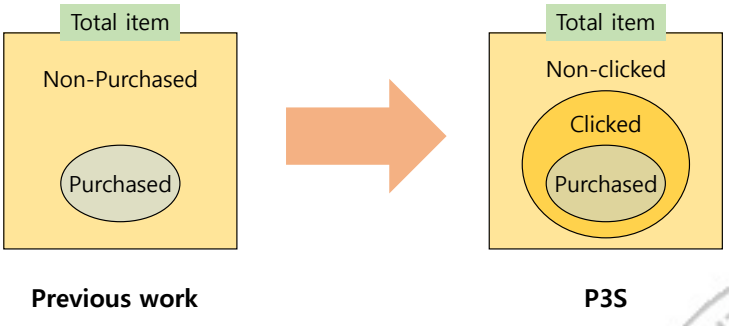


Figure 1.3: Leveraging click records to complement missing feedback.

Moreover, in **Chapter V**, we propose a novel sampling approach to cope with class imbalance problem; imbalance between purchased and clicked-but-not-purchased items. As shown in Figure 1.4, we calculate the decision

boundary θ and use it as a criterion for sampling. Specifically, a half of the data is sampled near the boundary, i.e., $\theta <$instances $\leq +\epsilon$, and the other half is sampled from the other area, i.e., instances $> \theta + \epsilon$.

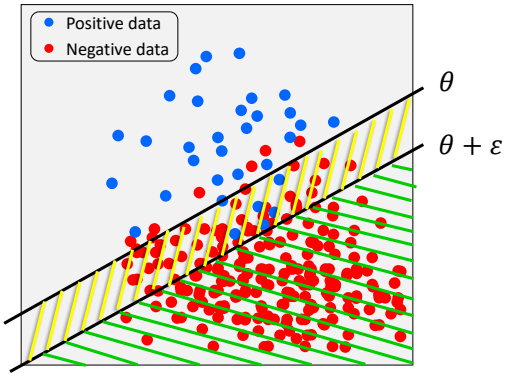


Figure 1.4: Leveraging click records to complement missing feedback.

RQ3. Can we model the intensity and the heterogeneity of user–item relationships in implicit feedback without using external auxiliary data?

A user's implicit feedback, such as click, view or purchase, does not necessary indicate that he holds an equal preference for the interacted items. Therefore, it is critical to extract some fine-grained user preferences towards items from these implicit feedback, and metric learning approaches have been proposed accordingly. However, previous metric learning approaches suffer from an inherent limitation, which is that each user is projected to a single point in the metric space. Such a rigid restriction imposed on user embeddings makes it hard to properly model the intensity and the heterogeneity of user–item relationships in implicit feedback. In **Chapter VI**, we embed users and items as points in a low-dimensional metric space, and additionally introduce translation vectors to translate each user to multiple points. Equipped with the user–item specific translation vectors, a user is translated toward his relevant items by considering his relationships with the items, which facilitates the modeling of the intensity and the heterogeneity of user–item relationships in implicit feedback that were overlooked by the previous metric learning approaches

II. Recommendation based on truster and trustee relationship in user trust network.

2.1 Overview

Due to the data sparsity problem, social network information is often additionally used to improve the performance of recommender systems. While most existing works exploit social information to reduce the *rating prediction error*, e.g., RMSE, a few had aimed to improve the *top-k ranking prediction accuracy*. This paper proposes a novel *top-k ranking* oriented recommendation method, TRecSo, which incorporates social information into recommendation by modeling two different roles of users as trusters and trustees while considering the structural information of the network. Empirical studies on real-world datasets demonstrate that TRecSo leads to a remarkable improvement compared with previous methods in top-k recommendation.

2.2 Introduction

Due to the ever increasing amount of information available to users on the Web, recommender systems have recently gained their popularity in industrial companies such as Amazon[1], Netflix[2], Facebook[3], to name a few. Indeed, it has been reported that Amazon are generating about 35% of their revenue through personalized recommendations provided by their own recommendation algorithms [31]. Moreover, recommender systems are widely used to help academic researchers in their decision making process when searching for relevant scientific articles. [32, 33, 9]. Hence, improving the quality of recommender systems has been an attractive research problem for academic researchers.

In the beginning, a significant amount of research was devoted to accurately predicting the overall ratings to reduce the rating prediction error, e.g., RMSE, MAE [34, 35]. Given a user-item rating matrix, their main goal was to provide the correct rating that a user will give to an item. However, rather than better predicted ratings, users are more interested in seeing a list of top-k items that aligns with their preferences. Moreover, it is well known that minimizing the

[1] http://www.amazon.com/
[2] http://www.netflix.com/
[3] http://www.facebook.com/

rating prediction error does not always result in better top-k list of items [36, 37]. Consequently, the Learning-To-Rank (LTR) [38] method, a supervised machine learning method that directly builds a ranking list from training data without an intermediate step of rating prediction, has gained popularity to provide accurate results at top-k.

Despite the success of recommendation approaches, recommender systems suffer from an inherent limitation called the *data sparsity problem*, that is, the recommendation is hardly accurate due to lack of observations (i.e., ratings) because users typically rate a small number of items. To tackle the data sparsity problem, researchers have tried to incorporate auxiliary information such as social network relationship among users [39, 40, 6, 3, 5], text reviews on items [41, 13, 9], time related information [42, 20, 23] and items' quality information [43, 32, 33]. Specifically, this paper focuses on incorporating the social network information of users in the top-k recommendation.

Recently, two top-k recommendation methods have been developed to incorporate the social network information based on the LTR approach. Specifically, Yao et al. [44] linearly combine a user's taste and her direct friends' tastes in optimizing the top-k recommendation. However, they do not utilize other important information hidden in the social network such as the structural information or truster-trustee relationship [45]. Zhao et al. [7] optimize the top-k recommendation from relative ordering that can be extracted from purchase history or browsing history, but they cannot handle numerical ratings directly. Note that numerical ratings usually contain much richer information on user's preference than relative ordering.

This paper proposes a novel LTR-based top-k recommendation method, TRecSo, which leverages the social network information to optimize top-k recommendation. TRecSo is distinguished from previous methods in that it models two different roles of users as trusters and trustees while considering the structural information of the network. Precisely, we map users into two types of low dimensional spaces according to their roles, that is, *truster* space and *trustee* space under the following assumptions:

- When *user A* is given several choices of items, he turns to people he trusts to ask them their opinions about the items.
- Consequently, the behavior of *user A* will influence the people that trust *user A*.

We summarize our contributions as follows:

- We develop a novel Learning-To-Rank (LTR) based top-k recommendation method that takes into account the social network information among users to alleviate the data sparsity problem.
- We map users into two types of low dimensional spaces according to their roles while considering the structural information of the trust network.
- Our experimental results on three real-world datasets indicate that our method considerably outperforms previous methods in top-k recommendation.

The remainder of this paper is organized as follows. Section 2.3 describes the related work; The problem is formally discussed in Section 2.4; In Section 4.4, we describe our proposed method, TRecSo, and demonstrate the learning algorithm; In Section 4.6, we describe the complexity of our proposed method followed by experimental results in Section 2.7. Finally in Section 2.8, we conclude the paper.

2.3 Related Work

Triggered by the Netflix prize [34] that ended in 2009, the field of recommender system has seen rapid progress since then. In this section, we review the existing methods including traditional collaborative filtering methods, *social* network based recommender systems, top-k ranking oriented recommender systems and top-k ranking oriented *social* recommender systems, the most related work to ours.

2.3.1 Traditional Collaborative Filtering

Typically, traditional collaborative filtering (CF) methods can be divided into two major categories, memory-based CF and model-based CF. Memory-based CF uses similarity measures such as Pearson Correlation or Cosine Similarity between users or items, and recommend items based on the neighbors [46, 47]. While gaining much popularity due to its easy implementation and interpretation, it does not scale well to large datasets and the performance decreases as ratings become sparse. Model-based CF, on the other hand, instead of simply computing similarities, learns a parametric model to fit the user-item ratings, which generally results in a better performance than memory-based CF [48]. Among various model-based methods [49, 50, 51], matrix factorization [35, 52] is the most widely used method upon which our proposed method, TRecSo, is built.

2.3.2 Social recommender system

Due to the inherent nature of recommender system, where users only rate a small number of items, the user-item ratings data is typically very sparse, that is, the entries are mostly unknown. Recently, social network based recommender systems have gained spotlight due to the rapid growth of social network services including Facebook[4], Twitter[5]. In a way akin to the traditional CF, social recommender systems are also categorized [53] into memory-based method [54, 55] and model-based method. Moreover, model-based methods are further divided into three different categories: Matrix co-factorization methods [3, 56], ensemble methods [4, 57] and regularization methods [58, 5, 59].

As the most representative memory-based social recommender approach, Jamali *et al.* [54] combine the memory-based CF based approach and the social network based approach under the random walk framework. They show that the prediction accuracy is improved by preferring raters with a shorter distance. Moreover, Ma *et al.* [3] were first to incorporate users' social network information into a matrix co-factorization based recommender system. They propose a probabilistic model that fuses the user-item rating data with the users' social network data by sharing a common user-specific latent factor and formulate the objective function as follows:

$$\min \sum_{i=1} \sum_{j=1} \left(r_{ij} - p_i^T q_j\right)^2 + \sum_{i=1} \sum_{k=1} \left(s_{ij} - p_i^T z_k\right)^2 \tag{2.1}$$

where $p_i \in \mathbb{R}^K$, $q_j \in \mathbb{R}^K$ and $z_k \in \mathbb{R}^K$ denote user-specific, item-specific latent vectors and factor-specific latent vectors, respectively; r_{ij} denotes the rating of user i on item j, and s_{jk} is equal to 1 if user j and user k are socially related, and 0 otherwise. In addition, Ma *et al.* [4] proposed an extended method that linearly combines a user's taste and her friends' tastes for modeling ratings as:

$$\hat{r}_{ij} = \alpha p_i^T q_j + (1 - \alpha) \sum_{k \in T_i} \tilde{s}_{ik} p_k^T q_j \tag{2.2}$$

where $\tilde{s}_{ik}$ represents the normalized trust strength between a user i and user k; α balances between user's taste and his friends' tastes and T_i denotes the social friends of user i. Finally, Ma *et al.* [5] propose a regularization method under the assumption that a user's preference should be similar to that of his social friends, so they formulate the objective function by forcing the user latent feature vectors

[4] http://www.facebook.com/
[5] http://www.twitter.com/

to be close to those of his friends as follows:

$$\min \sum_i \sum_j \left(r_{ij} - p_i^T q_j\right)^2 + \alpha \sum_i \left(p_i - \sum_{k \in T_i} s_{ik} p_k\right)^2 \tag{2.3}$$

where α controls the importance of the social regularization.

However, while the vast majority of commercial recommender systems provide recommended item lists to users because most users are only interested in seeing top-k items, the aforementioned social recommender systems mainly focus on minimizing the rating prediction error., e.g., RMSE, MAE. As a matter of fact, improving such traditional error criteria does not necessarily lead to improving top-k performance [36, 60, 37]. Therefore, in this paper we focus on finding a better top-k list of items rather than correctly predicting the ratings.

2.3.3 Top-k ranking oriented recommender system

As mentioned in the previous section, the eventual goal of a recommender system is to provide a top-k list of items as a recommendation. Consequently, several approaches have been proposed with the objective of top-k recommendation, which cast this problem as Learning-to-Rank (LTR) [38] where a personalized ranking list is directly generated from the training data without an intermediate step of the rating prediction. LTR based top-k recommendation approaches are categorized into pair-wise and list-wise models. Pair-wise models learn users' relative preferences of each item pair [61, 26]. Although pair-wise models have shown substantial improvements in terms of top-k recommendation, they have issues with high computational complexity. Most recently, list-wise models have gained much attentions due to its scalability compared with pair-wise models [62, 63]. Given a list of items as a training sample, list-wise models directly predict a ranking list of items for each user based on the distance between the ground truth ranking list and the predicted list. In this paper, we adopt the list-wise approach and demonstrate the superiority of our proposed model, TRecSo, over the relevant pair-wise and list-wise competitors.

2.3.4 Top-k ranking oriented social recommender system

Approaches that belong to this section are considered as direct baselines to our model, TRecSo. In this section, we introduce two state-of-the-art methods [44, 7] and address their limitations.

Zhao *et al.* [7] propose a pair-wise LTR approach based on the assumption that users tend to show a higher preference to items that their friends prefer than

items that they are not aware of at all. However, their approach cannot handle numerical ratings directly and they have high computational complexity due to the inherent nature of pair-wise LTR approach. Moreover, Yao *et al.* [44] adopt the rating model of [4], and linearly combine a user's taste and her friends' tastes for modeling ratings. Given the rating model as Equation 2.2, they optimize top-k recommendation list by using a concept called *top-one probability*, which will be explained in Section 2.4 for later use in our model. However, they fail to utilize other important information hidden in social network such as the structural information and truster-trustee relationship. In this paper, we demonstrate that our approach, TRecSo, outperforms these two state-of-the-art approaches over three real-world datasets including datasets used in [44] and [7].

2.4 Problem Description

We first introduce the notations that we use throughout the paper. Let $\mathcal{U} = \{u_1, u_2, ..., u_N\}$ be the set of users and $\mathcal{V} = \{v_1, v_2, ..., v_M\}$ be the set of items, where N and M are number of users and items, respectively. The ratings given by users in $\mathcal{U}$ on items in $\mathcal{V}$ are represented by a rating matrix $\mathcal{R} = [r_{ij}]_{N \times M}$, where r_{ij} denotes the rating of u_i on v_j. Depending on applications, r_{ij} can be either a real number or a binary value. When users explicitly express their opinions on items, the rating value is a real number, often in range [1,5], and when $\mathcal{R}$ reflects users' action, such as click or non-click and bookmark or non-bookmark, r_{ij} is a binary value. Although this paper focuses on the former case, it can be extended to the latter case as well. Without loss of generality, we convert the ratings of 1...5 into the interval [0,1] by normalization. In addition, $\mathcal{S} = [s_{ik}]_{N \times N}$ is the user trust matrix, where $s_{ik} = 1$ denotes that u_i trusts u_k. It is worth noting that the matrix $\mathcal{S}$ is asymmetric, that is, $s_{ik} \neq s_{ki}$, which means that the fact that u_i trusts u_k does not imply that u_k trusts u_i. Epinion[6], Ciao[7] and Twitter[8] are the most representative applications that provide asymmetric relationship between users, and such asymmetric relationship is the generalized version of symmetric relationship. i.e, Friendship. Therefore, our method can be certainly applied to the case of symmetric relationship. With the aforementioned notations, we can formally specify our problem as follows:

Problem Definition

Given: *The observed rating matrix* $\mathcal{R}$ *and the trust matrix* $\mathcal{S}$

[6]http://www.epinions.com/
[7]http://www.ciao.com/
[8]http://www.twitter.com

Goal: *Recommend each user a ranking list of unobserved items considering their personal preferences.*

2.5 Method

In Section 2.5.1, we briefly explain the concept of *Plackett-Luce model* that is required for understanding our proposed model, TRecSo. Then, we explain how the rating and the trust relationship are modeled in Section 2.5.2 and 2.5.3, respectively, and propose a unified model that combines the rating and the trust in Section 2.5.4, where we show how the unified model is learned.

2.5.1 Preliminary: Plackett-Luce model

Plackett-Luce model [64] computes the probability distribution over the permutations of items that each user has rated. The underlying assumption is that different permutation of items have different probability distributions, and a high permutation probability value implies that the permutation is more likely to be preferred by the user.

Probability of Permutation Given a set of M items $\mathcal{V}$, let $\pi = \{\pi_1, \pi_2, ..., \pi_M\}$ be one of the possible permutations of $\mathcal{V}$. If its corresponding ratings are given by $\{r_{\pi_1}, r_{\pi_2}, ..., r_{\pi_M}\}$, the probability of the permutation π is defined as follows:

$$P(\pi) = \prod_{i=1}^{M} \frac{\phi(r_{\pi_i})}{\sum_{k=i}^{M} \phi(r_{\pi_k})}$$

where r_{π_i} is the rating given to item π_i and $\phi(r) = e^r$. However, the time complexity of such computation is too high, since the number of different permutations is $M!$ for a set of M items. Therefore, to alleviate such problem, we employ the probability of only the top-ranked item among a given ranked list as follows:

$$P(\pi_i) = \frac{\phi(r_{\pi_i})}{\sum_{j=1}^{k} \phi(r_{\pi_j})} \tag{2.4}$$

Equation 2.4 models the probability of an item scored r_{π_i} being ranked on the top-one position in the list and we call it *top-one probability*.

2.5.2 Modeling Rating

Due to the asymmetry property of the trust matrix as mentioned in Section 2.4, we map each user into two different latent vectors, each of which rep-

resents truster or trustee. Such mappings are reasonable under the assumption that when a user is given several choices of items, he asks the people he trusts for their opinions about the items, and also the decision made by the user will influence the people that trust the user. Then, the rating of u_i on v_j is predicted as follows:

$$\hat{r}_{ij} = g(\mu + b_{u_i} + b_{v_j} + {q_j}^T(\alpha p_i + (1-\alpha)w_i + |I_i|^{-\frac{1}{2}} \sum_{t \in I_i} y_t + |T_i|^{-\frac{1}{2}} \sum_{v \in T_i} x_v)) \quad (2.5)$$

where $g(\cdot)$ is the logistic function that bounds the range of predicted ratings into [0,1]; μ is the average of all ratings; b_{u_i} and b_{v_j} represent the user and item biases for u_i and v_j, respectively. User biases and item biases should be taken into consideration because some users tend to give higher ratings than others, and some items (e.g., famous items) tend to receive higher ratings than others. Note that it is well known from the result of the Netflix Prize [34, 35] that incorporating the concept of biases significantly boosts the performance of the matrix factorization based recommender system. q_j represents the latent vector of item j; p_i and w_i represent the user latent vectors as a truster and as a trustee role of u_i, respectively, which are also used to model the social information in Equation 2.7. In other words, a user latent vector is modeled by a linear combination of truster specific latent vector and trustee specific latent vector, since a user plays a role of truster and trustee at the same time. α balances between the truster and trustee roles. (the higher the value of α, the more weight is given to the truster role of a user); I_i denotes the set of items rated by u_i; y_t is the latent vector of item t, which models implicit influence of items rated by u_i; T_i is the set of users that u_i trusts (e.g., whom u_i follows in social network); and x_v is the latent vector of whom u_i trusts, which models implicit influence of the users trusted by u_i. It is worth mentioning that although we do not explicitly model the trust propagation in our model, we include the implicit influence of social friends ($|T_i|^{-\frac{1}{2}} \sum_{v \in T_i} x_v$) of user i so that users with only a few ratings are rather modeled by their social friends. Note that this way of modeling the rating value is distinguished from existing methods [6, 35] in that none of them integrated the concept of bias and two roles (Truster and Trustee) of users while at the same time considering the implicit influence of the users trusted by u_i.

By adopting the *top-one probability* 2.4, we are now able to formulate the objective function aiming at minimizing the uncertainty between the training list and the predicted list of ratings by using cross-entropy measure as follows, which

can be interpreted as list-wise ranking prediction:

$$\mathcal{L}(b_U, b_V, p, w, q, y, x) = -\sum_{i=1}^{N}\sum_{j\in I_i} P_{l_i}(r_{ij}) \log P_{l_i}(\hat{r}_{ij}) + \frac{\lambda_p}{2}||p_i||_F^2 + \frac{\lambda_w}{2}||w_i||_F^2 + \frac{\lambda_v}{2}||q_j||_F^2 + \frac{\lambda_c}{2}\left(\sum_{t\in I_i}||y_t||_F^2 + \sum_{v\in T_i}||x_v||_F^2\right) + \frac{\lambda_b}{2}\left(||b_{u_i}||_F^2 + ||b_{v_j}||_F^2\right) \tag{2.6}$$

where $P_{l_i}(r_{ij})$ models the probability of an item scored r_{ij} being ranked on the top-one position in u_i's ranked list l_i; λ_p, λ_w, λ_v, λ_c and λ_b are regularization parameter for truster vector, trustee vector, item vector, implicit vectors and bias vectors, respectively. Note that the last five terms are regularization terms to avoid model over-fitting.

2.5.3 Modeling Trust

Intuitively, the unknown relationship $\hat{s}_{ik}$ between u_i and u_k can be estimated as follows:

$$\hat{s}_{ik} = g(b_{p_i} + b_{w_k} + w_k^T p_i) \tag{2.7}$$

where b_{p_i} and b_{w_k} represent the truster bias and trustee bias, respectively. By sharing the term p_i and w_k in Equation 2.5 and Equation 2.7, and by *simultaneously learning both latent models*, we are able to properly model the different roles of users as trusters and trustees.

To reflect the structural information of the network, s_{ik} is adjusted based on the degree of nodes such that it gives lower weights to those who *trust* many users (large out-degrees) and gives higher weights to those who *are trusted* by many users (large in-degrees):

$$s_{ik}^* = \sqrt{\frac{Indegree(v_k)}{Outdegree(v_i) + Indegree(v_k)}} \times s_{ik} \tag{2.8}$$

where v_i and v_k are nodes for u_i and u_k in the network, respectively [3]; $Indegree(v)$ and $Outdegree(v)$ are links that come into the node v and links that go out from the node v. It is worth noting that the model performance has been in fact improved by this adjustment and it is demonstrated in Section 2.7.2.

Likewise, the objective function for minimizing the cross-entropy between the probability distribution of training list and the predicted list of trust values

can be formulated as follows:

$$\mathcal{L}(b_p, b_w, p, w) = -\sum_{i=1}^{N}\sum_{k\in T_i} P_{l_i}(s^*_{ik}) \log P_{l_i}(\hat{s}_{ik}) + \frac{\lambda_b}{2}\left(||b_{p_i}||_F^2 + ||b_{w_i}||_F^2\right) + \frac{\lambda_p}{2}||p_i||_F^2 + \frac{\lambda_w}{2}||w_i||_F^2 \tag{2.9}$$

where $P_{l_i}(s_{ik})$ models the probability of trust strength s_{ik} being ranked on the top-one position in u_i's ranked list l_i. Note that the last three terms are regularization terms to avoid model over-fitting.

2.5.4 Unified Model

So far, we have described how the ratings and the trust relations are modeled in TRecSo. In this section, we demonstrate the unified model that integrates the two models, and we show how the parameters are learned.

In order to jointly model the ratings and the social relationships among users, we formulate a unified model by combining the equation 2.6 and equation 2.9, which is given as follows:

$$\begin{aligned}\mathcal{L} = &-\sum_{i=1}^{N}\sum_{j\in I_i} P_{l_i}(r_{ij}) \log P_{l_i}(\hat{r}_{ij}) - \lambda_t \sum_{i=1}^{N}\sum_{k\in T_i} P_{l_i}(s^*_{ik}) \log P_{l_i}(\hat{s}_{ik}) \\ &+ \frac{\lambda_p}{2}||p_i||_F^2 + \frac{\lambda_w}{2}||w_i||_F^2 + \frac{\lambda_v}{2}||q_j||_F^2 + \frac{\lambda_c}{2}\left(\sum_{t\in I_i}||y_t||_F^2 + \sum_{v\in T_i}||x_v||_F^2\right) \\ &+ \frac{\lambda_b}{2}(||||b_{u_i}||_F^2 + ||b_{v_j}||_F^2 + |b_{p_i}||_F^2 + ||b_{w_i}||_F^2)\end{aligned} \tag{2.10}$$

where λ_t controls the importance of trust regularization, which means that the higher the λ_t, the more impact a user's trusters have on the user's preference. Note that in addition to the regularization terms in Equation 2.10, we adopt weighted-λ-regularization [65] to further avoid model over-fitting. Moreover, in order to reduce the model complexity, we set $\lambda_p = \lambda_w = \lambda_v = \lambda_c = \lambda$.

Optimization Procedure Having formulated the non-convex objective function as shown Equation 2.10, we compute the gradient of each latent vector, i.e., $p_i, w_k, q_j, b_{u_i}, b_{p_i}$, $b_{w_k}, b_{q_j}, y_t, x_v$, and learn them by stochastic gradient descent [66] from which we obtain the local minimum solution. The gradients are

given as follows:

$$\frac{\partial \mathcal{L}}{\partial p_i} = \alpha \sum_{k \in I_i} e_{ij} \cdot q_j + \lambda_t \sum_{v \in T_i} e_{iv} \cdot w_v + (\lambda + \lambda_t) \cdot p_i \tag{2.11}$$

$$\frac{\partial \mathcal{L}}{\partial w_k} = (1-\alpha) \sum_{j \in I_k} e_{ik} \cdot q_j + \lambda_t \sum_{i \in T_k} e_{ik} \cdot p_i + (\lambda + \lambda_t) \cdot w_k \tag{2.12}$$

$$\frac{\partial \mathcal{L}}{\partial q_j} = \alpha \sum_{i \in U_j} e_{ij} \left(\alpha p_i + (1-\alpha) w_i + |I_i|^{-\frac{1}{2}} \sum_{t \in I_i} y_t + |T_i|^{-\frac{1}{2}} \sum_{v \in T_i} x_v \right) q_j + \lambda q_j \tag{2.13}$$

$$\frac{\partial \mathcal{L}}{\partial b_{u_i}} = \sum_{j \in I_i} e_{ij} + \lambda b_{u_i} \tag{2.14}$$

$$\frac{\partial \mathcal{L}}{\partial b_{p_i}} = \lambda_t \sum_{v \in T_i} e_{iv} + \lambda_t b_{p_i} \tag{2.15}$$

$$\frac{\partial \mathcal{L}}{\partial b_{w_k}} = \lambda_t \sum_{i \in T_k} e_{ik} + \lambda_t b_{w_k} \tag{2.16}$$

$$\frac{\partial \mathcal{L}}{\partial b_{q_j}} = \sum_{i \in U_j} e_{ij} + \lambda b_{q_j} \tag{2.17}$$

$$^{\forall} t \in I_i, \frac{\partial \mathcal{L}}{\partial y_t} = \sum_{j \in I_i} e_{ij} |I_i|^{-\frac{1}{2}} q_j + \lambda y_t \tag{2.18}$$

$$^{\forall} v \in T_i, \frac{\partial \mathcal{L}}{\partial x_v} = \sum_{j \in I_i} e_{ij} |T_i|^{-\frac{1}{2}} q_j + \lambda x_v \tag{2.19}$$

$$\text{where } e_{ij} = \left(\frac{\exp(g(\hat{r}_{ij}))}{\sum_{k \in I_i} \exp(g(\hat{r}_{ik}))} - \frac{\exp(r_{ij})}{\sum_{k \in I_i} \exp(r_{ik})} \right) g'(\hat{r}_{ij}) \tag{2.20}$$

$$e_{ik} = \left(\frac{\exp(g(\hat{s}_{ik}))}{\sum_{k \in I_i} \exp(g(\hat{s}_{ik}))} - \frac{\exp(s^*_{ik})}{\sum_{k \in I_i} \exp(s^*_{ik})} \right) g'(\hat{s}_{ik}). \tag{2.21}$$

Note that $g'(x) = \exp(x)/(1+\exp(x))^2$ is the derivative of logistic function $g(x)$. The detailed steps of TRecSo are shown in Algorithm 1 and the notations are explained in Table 2.1. Note that the algorithm terminates when the difference in loss between two iterations is less than 0.000001 or when reaching the predetermined number of iterations.

Algorithm 1 TRecSo Algorithm

Input: $\mathbf{R}$: User-Item rating matrix, $\mathbf{S}$: User-User trust relation matrix
Output: Learned model parameters $\mathbf{p}, \mathbf{w}, \mathbf{q}, \mathbf{b_u}, \mathbf{b_p}, \mathbf{b_w}, \mathbf{b_q}, \mathbf{y}, \mathbf{x}$

repeat
  for $i = 1$ to N **do**
    for $j = 1$ to M **do**
      Calculate $\frac{\partial \mathcal{L}}{\partial p_i}, \frac{\partial \mathcal{L}}{\partial w_i}, \frac{\partial \mathcal{L}}{\partial q_j}, \frac{\partial \mathcal{L}}{\partial b_{u_i}}, \frac{\partial \mathcal{L}}{\partial b_{q_j}}, \frac{\partial \mathcal{L}}{\partial y_t}, \frac{\partial \mathcal{L}}{\partial x_v}$ for all $t \in I_i$ and $v \in T_i$ using Eq. 2.11, 2.12, 2.13, 2.14, 2.17, 2.18, 2.19
      Update $p_i \leftarrow p_i - \gamma \frac{\partial \mathcal{L}}{\partial p_i}$
      Update $w_i \leftarrow w_i - \gamma \frac{\partial \mathcal{L}}{\partial w_i}$
      Update $q_j \leftarrow q_j - \gamma \frac{\partial \mathcal{L}}{\partial q_j}$
      Update $b_{u_i} \leftarrow b_{u_i} - \gamma \frac{\partial \mathcal{L}}{\partial b_{u_i}}$
      Update $b_{q_j} \leftarrow b_{q_j} - \gamma \frac{\partial v}{\partial b_{q_j}}$
      Update $y_t \leftarrow y_t - \gamma \frac{\partial \mathcal{L}}{\partial y_t}$
      Update $x_v \leftarrow x_v - \gamma \frac{\partial \mathcal{L}}{\partial x_v}$
    end for
  end for
  for $i = 1$ to N **do**
    for $k = 1$ to N **do**
      Calculate $\frac{\partial \mathcal{L}}{\partial p_i}, \frac{\partial \mathcal{L}}{\partial w_k}, \frac{\partial \mathcal{L}}{\partial b_{p_i}}, \frac{\partial \mathcal{L}}{\partial b_{w_k}}$ using Eq. 2.11, 2.12, 2.15, 2.16
      Update $p_i \leftarrow p_i - \gamma \frac{\partial \mathcal{L}}{\partial p_i}$
      Update $w_k \leftarrow w_k - \gamma \frac{\partial \mathcal{L} L}{\partial w_k}$
      Update $b_{p_i} \leftarrow b_{p_i} - \gamma \frac{\partial \mathcal{L}}{\partial q_j}$
      Update $b_{w_k} \leftarrow b_{w_k} - \gamma \frac{\partial \mathcal{L}}{\partial b_{u_i}}$
    end for
  end for
until Convergence

Table 2.1: Notation

Symbol	Description
N, M, K	number of users, items and latent dimensions
$\mathbf{p} \in \mathbb{R}^{N \times K}, \mathbf{w} \in \mathbb{R}^{N \times K}$	truster, trustee specific latent matrix
$\mathbf{q} \in \mathbb{R}^{M \times K}$	item latent matrix
$\mathbf{b_u} \in \mathbb{R}^{N}, \mathbf{b_q} \in \mathbb{R}^{M}$	user-bias, item-bias latent vector
$\mathbf{b_p} \in \mathbb{R}^{N}, \mathbf{b_w} \in \mathbb{R}^{N}$	truster-bias, trustee-bias latent vector
$\mathbf{y} \in \mathbb{R}^{M \times K}$	item latent matrix that represents implicit influence
$\mathbf{x} \in \mathbb{R}^{N \times K}$	user latent matrix that represents implicit influence
γ	learning rate

2.6 Complexity Analysis

The computational time of learning the proposed model TRecSo is dominated by computation of the loss function $\mathcal{L}$ in Equation 2.10 and its gradients with respect to feature vectors given in Equation 2.11 to 2.19. Let $|\mathcal{R}|$ and $|\mathcal{S}|$ be the number of observed ratings and trust relations, respectively. Then, the complexity of evaluating $\mathcal{L}$ is $\mathcal{O}(K|\mathcal{R}|c+d|S|)$, where K is the size of latent dimensionality, $c = max(a, b)$ and $d = max(b, K)$. Note that a and b are the average ratings an item has received and the average ratings an has user rated. Since the matrices $\mathcal{R}$ and $\mathcal{S}$ are extremely sparse as shown in Table 2.2, the values of $|\mathcal{R}|$ and $|S|$ are substantially smaller. Furthermore, the computational complexity of computing gradients $\frac{\partial L}{\partial p_i}, \frac{\partial L}{\partial w_i}, \frac{\partial L}{\partial q_j}, \frac{\partial L}{\partial b_{u_i}}, \frac{\partial L}{\partial b_{q_j}}, \frac{\partial L}{\partial y_t}, \frac{\partial L}{\partial x_v}$ for all $t \in I_i$ and $v \in T_i$, that is, Equations 2.11 to 2.19, are $\mathcal{O}(K|\mathcal{R}|+K|S|), \mathcal{O}(K|\mathcal{R}|+K|S|), \mathcal{O}(a|\mathcal{R}|+K|\mathcal{R}|a+K|\mathcal{R}|b+K|\mathcal{R}|), \mathcal{O}(|\mathcal{R}|a+|\mathcal{R}|), \mathcal{O}(|\mathcal{R}|a+|\mathcal{R}|), \mathcal{O}(|\mathcal{R}|a+K|\mathcal{R}|a), \mathcal{O}(|\mathcal{R}|a+K|\mathcal{R}|b)$. Consequently, the overall complexity per iteration is $\mathcal{O}(K|\mathcal{R}|c + d|S|)$. Since $c \ll |\mathcal{R}|$ or $|S|$, the overall computational complexity is linear with respect to the number of observed ratings and trust relations. Therefore, unlike pair-wise LTR methods as mentioned in Section 2.3.3, our proposed model can be scaled to large datasets.

2.7 Experiments

In this section, we carry out experiments to compare the quality of the top-k recommendation of our method, TRecSo, with several state-of-the-art methods on three real-world datasets. Our experiments are designed to verify the following questions:

1. How does TRecSo perform compared with other related competitors?
2. Does considering the social network structure as in Equation 2.8 enhance the performance of TRecSo?
3. How does the trade-off parameter of TRecSo affect the quality of the top-k recommendation?
4. How does the dimensionality of user and item latent space affect the performance of TRecSo?

2.7.1 Experimental Setting

Datasets We use three public real-world datasets, each of which contains both user-item ratings and trust relationships among users. The trust relations be-

tween users are asymmetric in all three datasets. Note that the ratings in Epinions[9] and Ciao[10] are integers in range [1,5], whereas those in Filmtrust[11] are real values in range [0.5, 4] with step 0.5. The detailed data statistics are described in Table 2.2.

Table 2.2: Data Statistics

	Rating				Trust		
	User	Item	Rating	Density	User	Links	Density
FilmTrust	1,508	2,071	35,497	1.1366%	1,642	1,853	0.0687%
Ciao	7,375	99,746	278,483	0.0379%	7,375	111,781	0.2055%
Epinion	40,163	139,738	664,824	0.0118%	49,289	487,183	0.0201%

Experimental Protocol We adopt an experiment protocol called *Weak generalization* that has been widely used for evaluating the performance of top-k recommender system [67, 44]. Weak generalization is evaluated by predicting the rank of unrated items for users known at training time. For our experiments, we randomly select n=10, 20, 50 observed ratings for each user and used them for training, and the model performance is evaluated on the remaining observed ratings. In order to evaluate the models on at least 10 items per user, we remove users with less than 20, 30, 60 rated items, respectively.

Parameter Setting For all the comparing methods, we set the parameters with optimal values based on the cross-validation. As for our proposed method, TRecSo, we set $\alpha = 0.4, \lambda = 0.1, \lambda_t = 0.8$ and $\gamma = 0.01$ for Filmtrust and Ciao datasets, and $\alpha = 0.4, \lambda = 0.5, \lambda_t = 0.5$ and $\gamma = 0.001$ for Epinion dataset, where γ is the learning rate. As mentioned in Section 2.5.4, we set $\lambda_p = \lambda_w = \lambda_v = \lambda_c = \lambda$ in order to reduce the model complexity. Note that for all the experiments, we set the size of dimensionality to 5, and the results reported in this section are the mean values over 5 runs on 5 different datasets.

Evaluation Metric The standard evaluation metrics for the recommender systems with the task of predicting the correct ratings (not the list of top-k items) are Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). Both metrics measure the distance between the true ratings and the predicted ratings. However, since our paper aims at improving the top-k recommendation quality, we use Normalized Discounted Cumulative Gain (NDCG), which is one of the

[9] http://www.trustlet.org/downloaded_epinions.html
[10] http://www.jiliang.xyz/datasetcode/ciao.zip
[11] http://www.librec.net/datasets/filmtrust.zip

most commonly used metric in the field of information retrieval. Generally, when Web users use search engines, rather than examining all the pages returned by the search engine, they look at only a few pages on the top of the returned list. Indeed, NDCG pays more attention to the top of a ranked list, and gives a high score to the list that contains more relevant items near the top position of the ranked list. In other words, NDCG takes the actual rating values into consideration, and gives a high score to the list with high ratings near the top position. For example, regarding NDCG@1, a list with an item that received a 5-star rating on the top-1 position gets a higher score than a list with an item that received a 4-star rating on the top-1 position. The NDCG value at kth position with respect to u_i is defined as follows:

$$NDCG_i@k = Z \sum_{j=1}^{k} \frac{2^{r_{ij}} - 1}{\log(1+j)}$$

where Z is a normalization constant to make the NDCG of the optimal ranking equal to 1. Finally, by calculating the value of $NDCG_i@k$ of each u_i in $\mathcal{U}$ and taking the mean value of all the users, we obtain $NDCG@k$ as follows:

$$NDCG@k = \frac{1}{|\mathcal{U}|} \sum_{i=1}^{N} NDCG_i@k \tag{2.22}$$

where $|\mathcal{U}|$ is the number of users in $\mathcal{U}$. Moreover, while metrics such as Precision@k (the ratio of relevant items in the predicted top-k ranked list) and Recall@k (the ratio of the relevant items over all the relevant items for each user in the predicted top-k ranked list) are not appropriate for our case because these metrics are better suited for datasets with implicit feedback where only binary relevance information is given, we show the results for Precision@k and Recall@k to verify the superiority of our method in various evaluation metrics for ranking.

Competitors Since our method is a social network based learning-to-rank (LTR) recommendation method whose goal is to optimize the top-k recommendations, we compare with the following state-of-the-art recommendation methods. Our competitors are divided into three different categories: Traditional CF method, ratings-only-based LTR method and social network-based LTR method.

1. Traditional CF method

    - ItemKNN: A traditional recommendation method based on similarity of items.

2. Ratings-only-based LTR methods

   - WRMF [52]: A weighted matrix factorization algorithm with implicit feedback data (One-class collaborative filtering).
   - BPRMF [26]: An item recommendation algorithm based on pair-wise Learning-to-Rank strategy combined with matrix factorization.
   - ListRank [62]: A list-wise Learning-to-Rank method combined with matrix factorization.

3. Social network-based LTR methods

   - SBPR [7]: An extended version of BPRMF by including social network information (Pair-wise Learning-to-Rank).
   - SoRank [44]: A social network based list-wise Learning-to-Rank algorithm that linearly combines a user's taste and her direct friends' tastes in optimizing the top-k recommendation.
   - TRecSo: Our proposed method.

Our competitors are selected for the following reasons. First, by comparing the performance of traditional CF method (memory-based CF) with WRMF (model-based CF), we show that the model-based CF outperforms the memory-based CF as mentioned in Section 2.3.1. Second, we include ratings-only-based LTR methods to verify the benefit of incorporating the social network information in the top-k recommendation task. Finally, most recent social network-based LTR methods are considered as our direct competitors in order to justify the benefit of our proposed method.

Note that we did not consider methods such as [6, 3, 5, 55, 68] because their focus was minimizing the rating prediction error (RMSE and MAE) rather than optimizing top-k recommendation. Our method is implemented on top of LibRec[12], a Java library for recommender system, where most of our competitors are implemented.

2.7.2 Performance Analysis

Comparison with competitors We report the results of all the tested users in Figure 3.4. In order to evaluate our method in Precision and Recall, we consider the 4-5 star ratings as relevant to a user and and 1-3 as irrelevant. As shown in the figure, our proposed method (TRecSo) generally outperforms the competitors

[12]http://www.librec.net/

in all three datasets. Note that the performance benefit of TRecSo is especially significant in NDCG for the following two reasons.

1. The information loss is inevitable during the conversion of the explicit feedback datasets into implicit feedback datasets. Therefore, the performance benefit of TRecSo is not as significant in Precision and Recall.

2. NDCG is originally designed for evaluating the performance of methods based on the explicit feedback datasets.

Moreover, from the comparison with LRMF [62] we observe that incorporating the social information significantly improves the recommendation on top-k items. In addition, we conclude from the comparison with SoRank [44] that modeling the two different roles of users as trusters and trustee and incorporating the implicit influence of the users trusted by each user instead of linear combination of users and their friends boosts the performance of top-k recommendation.

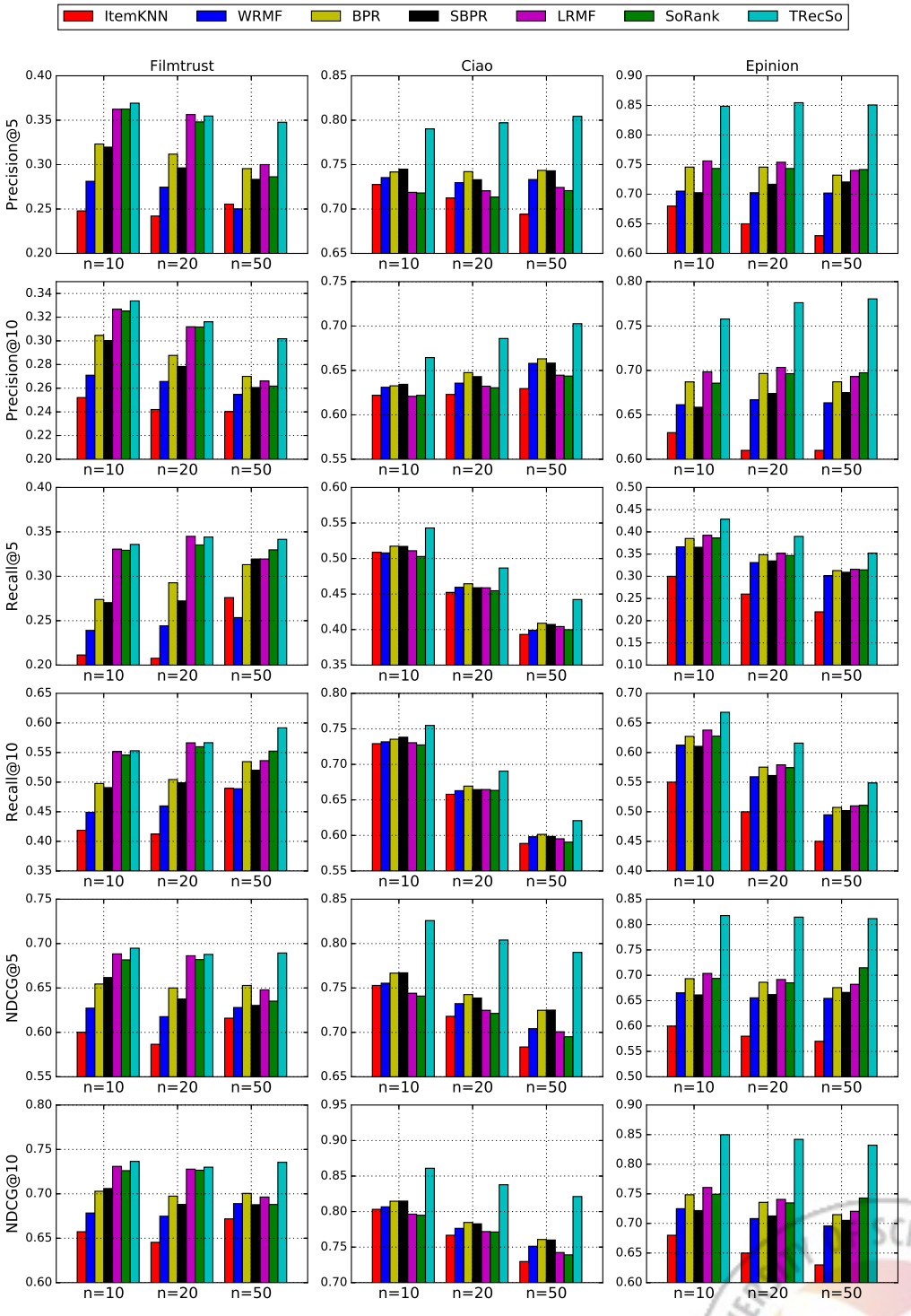


Figure 2.1: Performance comparison on three datasets

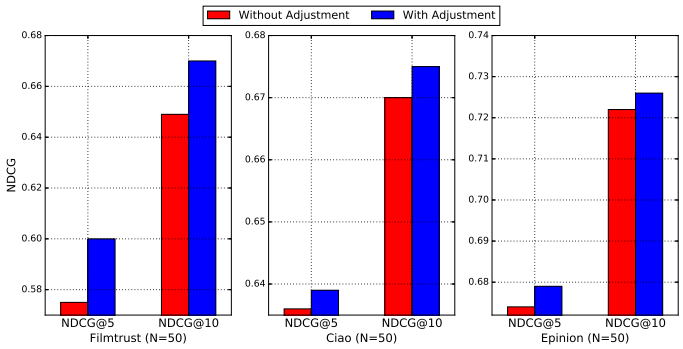


Figure 2.2: Impact of considering the structure of the social network

Consideration of social network structure In order to answer the second question that we mentioned in the beginning of Section 4.6, we conduct experiments to see whether the consideration of the social network structure of users improves the top-k recommendation performance on three datasets. As shown in Figure 2.2, we observe that adjusting s_{ik} by using Equation 2.8 indeed improves the NDCG values. The results clearly indicate that giving lower weights to users who trust many users and giving higher weights to those who are trusted by many users reflects the social phenomena of the real-world.

Impact of Parameters α and λ_t In our proposed model, TRecSo, there are two trade-off parameters that should be tuned, i.e., α and λ_t. In order to determine the best performing parameters on each dataset, we adjust one parameter while fixing the other. We first investigate the impact of α while fixing λ_t to 0.8. As explained in Section 4.4, α is the parameter for balancing the relative importance of influence of truster and trustee. Figure 2.3 illustrates the result of varying the value of α when the size of dimensionality is set to 5. The results show that the proper value of α exists, and it helps improve the recommendation quality.

After discovering the proper values of α, we then further explore the second trade-off parameter λ_t, which controls the importance of trust regularization. Figure 2.4 shows the performance of TRecSo while changing the values of λ_t and fixing $\alpha = 0.5$. It is clear from the results that incorporating the trust network, that is when $\lambda_t > 0$, as auxiliary information to the ratings improves the performance of top-k recommendation. However, we observe that the legitimate balance between the influence from the ratings and the influence from the trust network should be empirically discovered, and the proper value differs among datasets.

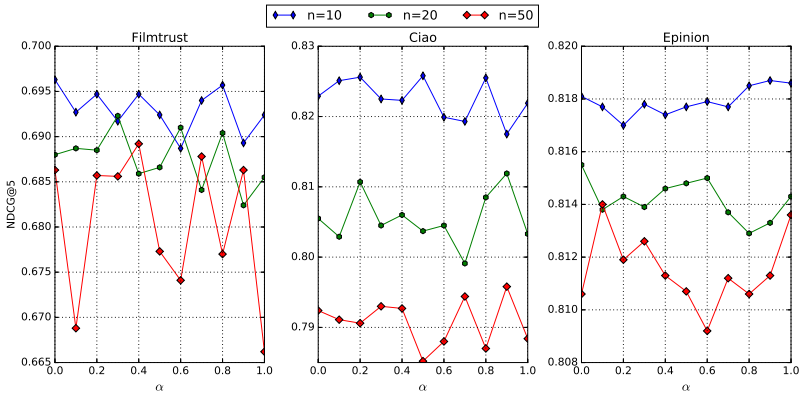


Figure 2.3: Impact of Parameter α on three datasets

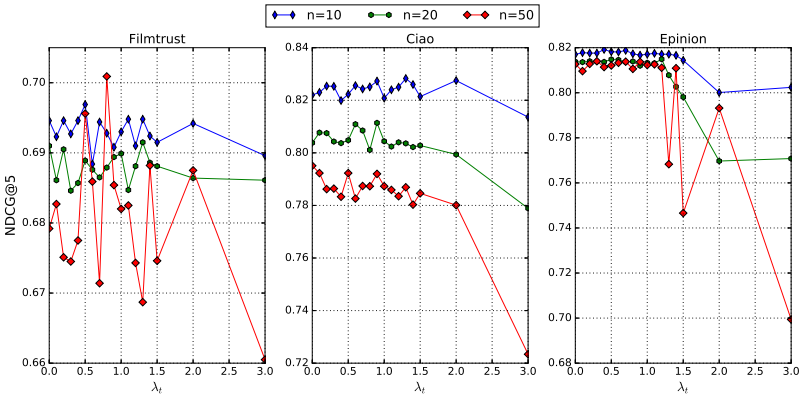
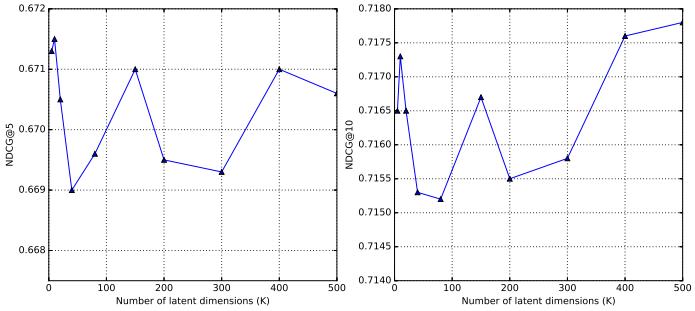


Figure 2.4: Impact of Parameter λ_t on three datasets

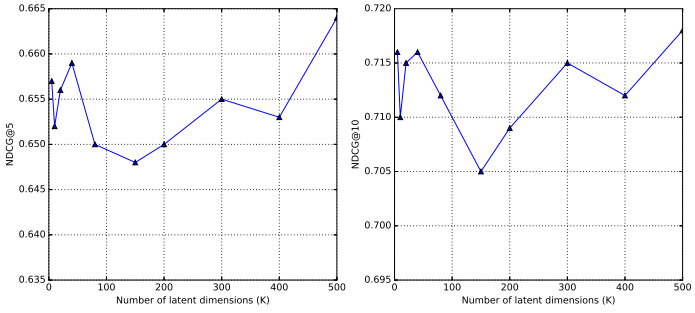
Dimensionality Analysis In this section, we address the fourth question that we introduced in Section 2.7, that is, to analyze the effect of the number of latent dimensions of user and item latent vectors on the performance of TRecSo. Generally, it is known from literatures that the performance of the recommendation improves as the number of latent dimensions increases [69, 70]. Figure 3.7 shows the performance with respect to the number of latent dimensions of our proposed model.

Interestingly enough, while experiments on Epinion dataset (Figure 2.5c) show the trend of performance improvement as the number of latent dimensions increases, we could not discover any particular trends from the experiments on Filmtrust and Ciao datasets (Figure 2.5a and2.5b). Although precisely interpreting the meaning of each latent dimension is infeasible, we assume that it represents the profile of user's interest and item's features. For datasets with relatively small number of users and items, such as Filmtrust and Ciao, a large

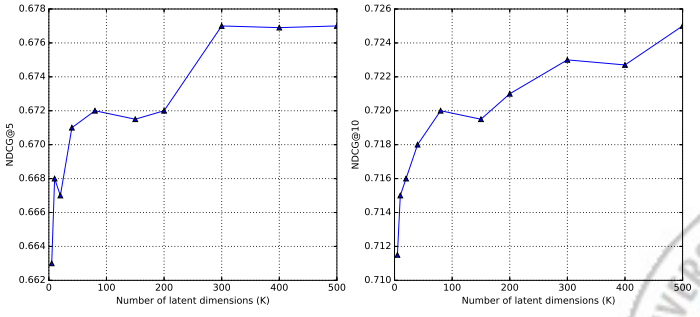
number of latent dimensionality would surpass the inherent number of profiles of users and items. However, for datasets like Epinion, which is composed of a large number of users and items, the performance of recommendation improves as the number of latent dimensionality increases. Nevertheless, if the number of dimensions is too large, the complexity will significantly increase. Therefore, we need to find a proper number of latent dimensions in order to balance the trade-off between the performance and the complexity.



(a) Filmtrust dataset (n=20)



(b) Ciao dataset (n=20)



(c) Epinion dataset (n=20)

Figure 2.5: Impact of latent dimensionality on three datasets

2.8 Conclusion

This paper proposes TRecSo, a novel LTR based recommendation method that optimizes the top-k ranking prediction accuracy by additionally considering the social network information. Specifically, TRecSo integrates the social network information into the Learning-To-Rank (LTR) based objective function for recommendation. Thanks to the flexibility (can be generalized to symmetric social relationship) and the low complexity (compared with pair-wise LTR approaches) of our model, our proposed method can be easily integrated into a real-world applications where user-item interaction history and user social network information are given. Comprehensive experimental results show that TRecSo significantly outperforms the state-of-the-art algorithms in the top-k ranking accuracy of recommendation.

III. Recommendation based on Item-affinity Network generated from "Also-viewed" Products

3.1 Overview

For online product recommendation engines, learning high-quality product embedding that captures various aspects of the product is critical to improving the accuracy of user rating prediction. In recent research, in conjunction with user feedback, the appearance of a product as side information has been shown to be helpful for learning product embedding. However, since a product has a variety of aspects such as functionality and specifications, taking into account only its appearance as side information does not suffice to accurately learn its embedding. In this paper, we propose a matrix co-factorization method that leverages information hidden in the so-called *"also-viewed"* products, i.e., a list of products that has also been viewed by users who have viewed a target product. *"Also-viewed"* products reflect various aspects of a given product that have been overlooked by visually-aware recommendation methods proposed in past research. Experiments on multiple real-world datasets demonstrate that our proposed method outperforms state-of-the-art baselines in terms of user rating prediction. We also perform classification on the product embedding learned by our method, and compare it with a state-of-the-art baseline to demonstrate the superiority of our method in generating high-quality product embedding that better represents the product.

3.2 Introduction

Triggered by the Netflix Prize [34] in 2009 whose goal was to predict user ratings on movies based on previous user feedback, the vast majority of research on recommender systems [72] has focused on accurately predicting user ratings [73]. To this end, it is crucial to learn high-quality product embedding, the dimensions of which align with those of user preference, since high-quality product embedding yields higher accuracy in user rating prediction [74]. Therefore, researchers have striven to learn high-quality product embedding by incorporating various side information related to products such as product reviews [75, 9, 10] and prod-

Figure 3.1: A target product and its *"also-viewed"* products.

uct images [15, 14], which help to learn better product embedding, and eventually lead to a better user rating prediction accuracy.

In this paper, we study the varying importance of product aspects for users in different product domains. Consider a real-world online shopping scenario where users rate products. What makes users assign high ratings to certain products? That is, what aspects of a product influence user ratings? The products' inherent aspects such as *appearance, functionality* or *specifications* certainly have a vital effect on user ratings. *However, the extent to which each aspect influences users varies among product domains.* For example, in clothing, the appearance of clothes is undoubtedly the most influential factor, whereas in "Office Products", such aspects as functionality and conformance to specifications mainly influence user ratings.

Notably, such phenomena are reflected in online shopping browsing histories in the form of *"also-viewed"* products, i.e., a list of products that have also been viewed by users who have viewed a target product. *"Also-viewed"* product information can be obtained from browsing histories of users. As an intuitive example, consider **Example 1**.

Example 1. Figure 3.1 shows examples of *"also-viewed"* products in different product domains in *Amazon*[1]. Interestingly, while the *"also-viewed"* products in clothing domain (Boys' and Girls' Clothing) look similar, those in other domains (Automotive, Pet supplies, Office products) are not similar in appearance, but are functionally related. Consider "Pet supplies" as an example. Given a liquid flea repellent as a target product, the *"also-viewed"* products

[1]http://www.amazon.com

include visually different but functionally related products, e.g., flea killing capsules and flea traps.

Example 1 shows that when users shop online, they pay more attention to different aspects of products in different product domains.

Although recent works [15, 14] have successfully taken into account the appearance of products in visually-aware product domain, they have yielded only a slight improvement in other domains where aspects such as functionality and specifications are significant for user ratings. This is because while every product domain has different aspects that are more influential to user ratings as shown in **Example 1**, existing methods only consider product appearance as side information that plays an important role only in clothing domain. Moreover, even in clothing, the appearance of a product is not the only factor that influences user ratings, and thus there is room for further improvement in modeling user ratings. Furthermore, due to the inherent data sparsity, existing methods have modeled users' visual preferences only based on the images of products rated by them in the past, which are usually very few in number. Consequently, the lack of rated products of users gives rise to the insufficiently modeled visual preferences of the users, which eventually degrades the accuracy of user rating prediction.

To address the aforementioned limitations of the existing works in the area, we propose a matrix co-factorization method called Visual Matrix Co-Factorization (VMCF). Our method leverages *"also-viewed"* products that reflect various aspect of a given product that have been previously overlooked by [15, 14]. Precisely, *"also-viewed"* products help systems to learn more high-quality product embedding for two reasons. First, *"also-viewed"* products encode not only visual similarity, but also functional or specification-related similarity, as shown in **Example 1**. Thus, we can capture aspects overlooked by visual features of a product by leveraging *"also-viewed"* products, even in clothing domain. In other domains, where the appearance of a product is not as significant, the explicit relationships among products expressed through *"also-viewed"* information are even more helpful than visual features in building more high-quality product embedding. Second, since most products have an insufficient number of ratings, the explicit relationships between each rated product and its unrated *"also-viewed"* products help us reflect various aspects of products in the product embedding, and thus eventually compensate for a lack of rated products. Moreover, *"also-viewed"* products are more helpful when only a few rated products are given, i.e., *"Cold-start"* setting.

Our main contributions are summarized as follows:

1. To reflect the relationships among products in terms of various aspects

such as appearances, functionality and specifications, we build a so-called *product-affinity network* using *"also-viewed"* products.

2. We then simultaneously factorize user ratings data and the *product-affinity network* by sharing the product embedding, which results in high-quality product embedding, and eventually, more accurate user rating prediction.

3. Experimental results on multiple real-world datasets demonstrate that our proposed method significantly outperforms state-of-the-art methods, especially in *"Cold-start"* setting where each product has only a few ratings.

4. By additionally performing classification on product embedding, we empirically demonstrate that the product embedding generated by our method represents the latent dimensions of products better than a state-of-the-art method.

The remainder of this paper is organized as follows: We briefly review related work in Section 3.3, and provide the formulation of the problem that we solve in Section 3.4. We then introduce our proposed method in Section 3.5, followed by results of comprehensive experiments on real-world datasets in Section 3.6. Finally, the conclusion is presented in Section 3.7.

3.3 Related Work

While recommender systems have lately generated a vast amount of research literature, we only review the studies closely related to ours, i.e., matrix factorization and visually-aware approaches for recommendation.

3.3.1 Matrix Factorization

The goal of matrix factorization (MF) is, given n users and m products, to decompose rating matrix $\mathcal{R} \in \mathbb{R}^{n \times m}$ into two low rank-L matrices $U \in \mathbb{R}^{L \times n}$ and $V \in \mathbb{R}^{L \times m}$, and minimize the reconstruction error as follows [76]:

$$\min_{U,V} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} \left(r_{ij} - U_i^T V_j\right)^2 + \frac{\lambda_u}{2} ||U||_F^2 + \frac{\lambda_v}{2} ||V||_F^2 \tag{3.1}$$

where r_{ij} is the rating assigned by user i to product j, $U_i \in \mathbb{R}^L$ and $V_j \in \mathbb{R}^L$ represent the embedding for user i and product j, respectively, where L is the dimensionality of the rating embedding space. λ_u and λ_v are regularization parameters for user embedding matrix U and product embedding matrix V, respectively, and

$||\cdot||_F^2$ denotes the Frobenius norm. The gradient descent-based optimization technique is generally applied to find the local minimum solution for Eq. 3.1. Matrix factorization is commonly used as a building block for extending recommender systems to incorporate side information such as social networks [6, 77, 78, 79], textual data [80, 75, 9, 10], or both [81, 82], temporal dynamics [83, 23], and product images [15, 14].

3.3.2 Visually-Aware Approaches for Recommendation

The appearance of a product is one of the most important aspects that impacts user ratings in online shopping. However, it has been commonly ignored because past image feature extraction methods failed to achieve satisfactory performance in visual machine learning tasks such as image classification and object detection. However, recent breakthroughs in deep learning has facilitated such tasks to attain high performance, and thus product image is now recognized as a valuable source of side information for recommender systems. Specifically, image features extracted from a pre-trained Convolutional Neural Network (CNN) effectively represent the latent properties of images. He *et al.* [15, 14] recently introduce visually-aware recommender systems based on the framework of matrix factorization. They embed high-dimensional image features extracted from a pre-trained CNN into low-dimensional features, and use them to model user and product visual embeddings. Their main concern is the appearance of a product, which plays a significant role in clothing domain. However, other product aspects are ignored, such as functionality and specification, which are certainly more significant than appearance in domains such as Automotive, Pet supplies, and Office products.

Meanwhile, McAuley *et al.* [84, 16] use the *"also-viewed"* product information to recommend visually alternative products in clothing domain as a *link prediction* task. Our proposed method is different from this method in that we leverage *"also-viewed"* product information for a *user rating prediction task* to consider *various aspects beyond appearance* in product embedding, which improves user rating prediction *in general domains* not limited to the clothing domain.

3.4 Problem Formulation

Notations We first introduce the notations used in this paper. Let $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ be the set of users and $\mathcal{V} = \{v_1, v_2, ..., v_m\}$ be the set of products, where n and m are the number of users and products, respectively. The

Table 3.1: Notations.

Notation	Explanation
$\mathcal{U}, \mathcal{V}$	User set ($\|\mathcal{U}\|= n$), Product set($\|\mathcal{V}\|= m$)
$\mathcal{R}$	Rating matrix ($n \times m$)
$\mathcal{S}$	*Product-affinity matrix* ($m \times m$)
$\mathcal{F}$	Visual feature matrix ($C \times m$)
f_j	Visual feature of product j ($C \times 1$)
r_{ij}	Rating assigned by user i to product j
s_{jk}	$s_{jk} = 1$ if product j and k are neighboring nodes
L	Num. dimensions of product embedding
D	Num. dimensions of product visual embedding
C	Num. dimensions of CNN features
U_i, V_j	Embedding for user i and product j ($L \times 1$)
P_i, Q_j	Visual embedding for user i and product j ($D \times 1$)
Z_k	*"Also-viewed"* product embedding for product k ($L \times 1$)
$\mathbf{E}$	Embedding kernel matrix ($D \times C$)
$g(\cdot)$	Sigmoid function

ratings assigned by users in $\mathcal{U}$ to products in $\mathcal{V}$ are represented by rating matrix $\mathcal{R} = [r_{ij}]_{n \times m}$, where r_{ij} denotes the rating that user i assigns to product j. Depending on the application, r_{ij} can be either a real number or a binary value. When users explicitly express their opinions on products, r_{ij} is a real number, often in the range [1,5], and when $\mathcal{R}$ reflects users' action such as click or non-click and bookmarked or not bookmarked, r_{ij} is a binary value. Although this paper focuses on the former case, it can readily be applied to the latter as well. Without loss of generality, we convert the ratings of 1...5 into the interval [0,1] through normalization. The notations used in the paper are summarized in Table 3.1.

3.4.1 Extracting Visual Features

As in [15, 14], we use a pre-trained CNN to extract features from product images. Precisely, we use the CNN architecture proposed by [85], which was trained on 1.2 million ImageNet (ILSVRC2010) images [86]. We pass m products through the pre-trained CNN, and extract the outputs of the second fully-connected layer to construct a product feature matrix $\mathcal{F} \in \mathbb{R}^{C \times m}$, the column vector $f_j \in \mathbb{R}^C$ of which denotes the visual feature vector of product $j \in \mathcal{V}$, where $C = 4096$. Moreover, we introduce an embedding kernel matrix $\mathbf{E} \in \mathbb{R}^{D \times C}$ to transform high-dimensional visual features $f_j \in \mathbb{R}^C$ into D-dimensional product visual embedding space by $\mathbf{E}f_j$.

3.4.2 Constructing Product-affinity Network

In order to incorporate *"also-viewed"* product information into our method, we build a so-called *product-affinity network* whose nodes denote products and edges encode the *"also-viewed"* relationships among products. As shown earlier in **Example 1**, neighboring products in the *product-affinity network* share common product aspects such as appearance, functionality and specifications, which implies that various aspects of the product are reflected in the network. Note that the edges are directed from a target product to its *"also-viewed"* products, and therefore the relationships are usually not symmetric. We represent the *product-affinity network* as a *product-affinity matrix* $\mathcal{S} = [s_{jk}]_{m \times m}$ such that each entry s_{jk} is defined as:

$$s_{jk} = \begin{cases} 1 & k \in N_j. \\ 0 & \text{otherwise.} \end{cases} \tag{3.2}$$

where N_j denotes the set of *"also-viewed"* products of product j.

Given the aforementioned notations, visual features and *product-affinity matrix*, our problem is defined as:

Problem Definition
Given: *The observed rating matrix* $\mathcal{R}$*, product visual feature matrix* $\mathcal{F}$ and *product-affinity matrix* $\mathcal{S}$
Goal: *Predict missing ratings* $r_{ij} \in \mathcal{R}$*, where* $i \in \mathcal{U}$ *and* $j \in \mathcal{V}$

3.5 Method

In this section, we describe our Visual Matrix Co-Factorization (VMCF) method that leverages *"also-viewed"* product information projected on *product-affinity matrix* $\mathcal{S}$ in order to take into account various product aspects overlooked by past visually-aware recommendation methods. We first explain how ratings $\mathcal{R}$ and the *product-affinity matrix* $\mathcal{S}$ are modeled independently, and demonstrate how these two models are jointly combined using the graphical model described in Figure 3.2.

3.5.1 Modeling Rating

Given user-product rating matrix $\mathcal{R} = [r_{ij}]_{n \times m}$, with n users and m products, let r_{ij} represent the rating of user i for product j, and $U \in \mathbb{R}^{L \times n}$ and $V \in \mathbb{R}^{L \times m}$ be user and product embedding matrices, with column vectors U_i

and V_j representing user-specific and product-specific embedding vectors, respectively. Moreover, in order to incorporate the visual factors into our model, given user and product visual embedding matrices $P \in \mathcal{R}^{D \times n}$ and $Q \in \mathcal{R}^{D \times m}$ respectively, the term $P_i^T Q_j$ is added [14] where $P_i \in \mathbb{R}^D$ and $Q_j \in \mathbb{R}^D$ denote the user-specific and product-specific visual embedding vectors, respectively, whose inner product models the visual correspondence between user i and product j. Given the embedded visual feature $Q_j = \mathbf{E}f_j$ as explained in Section 3.4.1, we define the conditional distribution over the observed ratings as:

$$\begin{aligned} P(\mathcal{R}|U,V,P,\mathbf{E},\sigma_R^2) &= \prod_{i=1}^{n}\prod_{j=1}^{m}\left[\mathcal{N}(r_{ij}|g(U_i^T V_j + P_i^T Q_j),\sigma_R^2)\right]^{I_{ij}^R} \\ &= \prod_{i=1}^{n}\prod_{j=1}^{m}\left[\mathcal{N}(r_{ij}|g(U_i^T V_j + P_i^T \mathbf{E}f_j),\sigma_R^2)\right]^{I_{ij}^R} \end{aligned} \tag{3.3}$$

where $\mathcal{N}(x|\mu,\sigma^2)$ denotes the probability density function of a Gaussian distribution with mean μ and variance σ^2, and I_{ij}^R is the indicator function that is equal to 1 if user i rated product j, and 0 otherwise. We use the logistic function $g(\cdot)$ to restrict the range of $U_i^T V_j + P_i^T \mathbf{E}f_j$ within [0,1], and convert rating r_{ij} into the range [0,1]. For each hidden variable, we place zero-mean spherical Gaussian priors [76] as follows:

$$\begin{aligned} P(U|\sigma_U^2) &= \prod_{i=1}^{n}\mathcal{N}(U_i|0,\sigma_U^2 I),\ P(V|\sigma_V^2) = \prod_{j=1}^{m}\mathcal{N}(V_j|0,\sigma_V^2 I) \\ P(P|\sigma_P^2) &= \prod_{i=1}^{n}\mathcal{N}(P_i|0,\sigma_P^2 I),\ P(\mathbf{E}|\sigma_{\mathbf{E}}^2) = \prod_{p=1}^{D}\prod_{q=1}^{C}\mathcal{N}(\mathbf{E}_{pq}|0,\sigma_{\mathbf{E}}^2) \end{aligned} \tag{3.4}$$

Given Eqs. 3.3 and 3.4, we can compute the log-posterior distribution over the hidden variables:

$$\begin{aligned} &P(U,V,P,\mathbf{E}|\mathcal{R},\sigma_U^2,\sigma_V^2,\sigma_P^2,\sigma_{\mathbf{E}}^2,\sigma_R^2) \\ &\propto P(\mathcal{R}|U,V,P,\mathbf{E},\sigma_R^2)P(U|\sigma_U^2)P(V|\sigma_V^2)P(P|\sigma_P^2)P(\mathbf{E}|\sigma_{\mathbf{E}}^2) \\ &= \prod_{i=1}^{n}\prod_{j=1}^{m}\left[\mathcal{N}(r_{ij}|g(U_i^T V_j + P_i^T \mathbf{E}f_j),\sigma_R^2)\right]^{I_{ij}^R} \\ &\times \prod_{i=1}^{n}\mathcal{N}(U_i|0,\sigma_U^2 I) \times \prod_{j=1}^{m}\mathcal{N}(V_j|0,\sigma_f^2 I) \times \prod_{i=1}^{n}\mathcal{N}(P_i|0,\sigma_P^2 I) \times \prod_{p=1}^{D}\prod_{q=1}^{C}\mathcal{N}(\mathbf{E}_{pq}|0,\sigma_{\mathbf{E}}^2) \end{aligned} \tag{3.5}$$

3.5.2 Modeling Also-viewed Relationships

With regard to the *product-affinity matrix* $\mathcal{S}$, we define the conditional distribution over the observed *product-affinity matrix* as:

$$P(\mathcal{S}|V, Z, \sigma_S^2, \sigma_Z^2) = \prod_{j=1}^{m} \prod_{k=1}^{m} \left[\mathcal{N}(s_{jk}|g(V_j^T Z_k), \sigma_S^2, \sigma_Z^2)\right]^{I_{jk}^S} \tag{3.6}$$

where $Z \in \mathbb{R}^{L \times m}$ is the *"also-viewed"* product embedding matrix, with column vector Z_k representing *"also-viewed"* product-specific embedding vector for product k. We model *"also-viewed"* relationship between product j and k by $V_j^T Z_k$ rather than $V_j^T V_k$ in order to reflect the asymmetric nature of the *product-affinity network*. I_{jk}^S is the indicator function that is equal to 1 if product k belongs to one of the *"also-viewed"* products of product j , and 0 otherwise. Again, we place zero-mean spherical Gaussian priors on the hidden variables V and Z as:

$$P(V|\sigma_V^2) = \prod_{j=1}^{m} N(V_j|0, \sigma_V^2 I),\ P(Z|\sigma_Z^2) = \prod_{k=1}^{m} N(Z_k|0, \sigma_Z^2 I) \tag{3.7}$$

Hence, given Eqs. 3.6 and 3.7, we can compute the log-posterior distribution over the hidden variables as:

$$\begin{aligned} &P(V, Z|S, \sigma_V^2, \sigma_Z^2, \sigma_S^2) \propto P(S|V, Z, \sigma_S^2) P(V|\sigma_V^2) P(Z|\sigma_Z^2) \\ &= \prod_{j=1}^{m} \prod_{k=1}^{m} \left[N(s_{jk}|g(V_j^T Z_k), \sigma_S^2)\right]^{I_{jk}^S} \times \prod_{j=1}^{m} N(V_j|0, \sigma_V^2 I) \prod_{k=1}^{m} N(Z_k|0, \sigma_Z^2 I) \end{aligned} \tag{3.8}$$

3.5.3 Unified Model

Thus far, we have shown how to independently model the ratings and *"also-viewed"* relationships among products. In this section, we propose a unified model where user ratings with associated product images are combined with *"also-viewed"* product information, which eventually yields high-quality product embeddings. Given an observed product j rated by user i, Figure 3.2 illustrates how we fuse both the rating model introduced in Section 3.5.1 and the *"also-viewed"* relationship model introduced in Section 3.5.2 into a matrix co-factorization framework by sharing the product embedding matrix V.

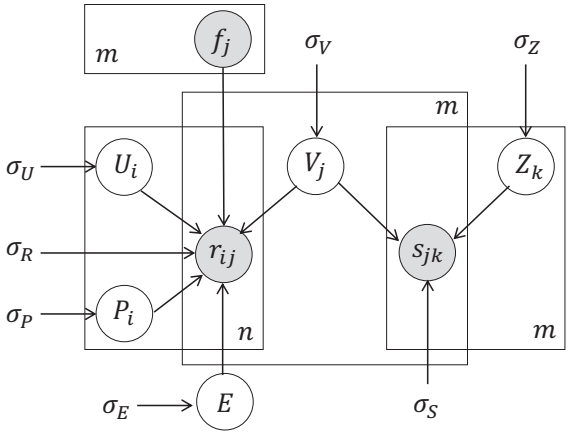


Figure 3.2: Graphical Model for Visual Matrix Co-Factorization (VMCF), where shaded nodes denote observed variables and the rest, hidden variables.

Based on Figure 3.2, the log-posterior distribution of VMCF is given by:

$$
\begin{aligned}
&P(U,V,P,Z,\mathbf{E}|R,S,\sigma_U^2,\sigma_V^2,\sigma_P^2,\sigma_Z^2,\sigma_{\mathbf{E}}^2,\sigma_R^2,\sigma_S^2) \\
&\propto P(\mathcal{R}|U,V,P,\mathbf{E},\sigma_R^2) \times P(S|V,Z,\sigma_S^2)P(U|\sigma_U^2)P(V|\sigma_V^2)P(P|\sigma_P^2)P(Z|\sigma_Z^2)P(\mathbf{E}|\sigma_{\mathbf{E}}^2) \\
&= -\frac{1}{2\sigma_R^2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}^R\big(r_{ij}-g(U_i^TV_j+P_i^T\mathbf{E}f_j)\big)^2 - \frac{1}{2\sigma_S^2}\sum_{j=1}^{m}\sum_{k=1}^{m} I_{jk}^S\big(s_{jk}-g(V_j^TZ_k)\big)^2 \\
&- \frac{1}{2\sigma_U^2}\sum_{i=1}^{n} U_i^TU_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{m} V_i^TV_i - \frac{1}{2\sigma_P^2}\sum_{i=1}^{n} P_i^TP_i \\
&- \frac{1}{2\sigma_Z^2}\sum_{k=1}^{m} Z_k^TZ_k - \frac{1}{2\sigma_{\mathbf{E}}^2}\sum_{p=1}^{D}\sum_{q=1}^{C}\mathbf{E}_{pq}^2 - \frac{1}{2}\left(\left(\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}^R\right)\ln\sigma_R^2 + \left(\sum_{j=1}^{m}\sum_{k=1}^{m} I_{jk}^S\right)\ln\sigma_S^2\right) \\
&- \frac{1}{2}\left(nL\ln\sigma_U^2+mL(\ln\sigma_V^2+\ln\sigma_Z^2)+nD\ln\sigma_P^2+DC\ln\sigma_{\mathbf{E}}^2\right)+\mathcal{C}
\end{aligned}
\tag{3.9}
$$

where $\mathcal{C}$ is a constant that is independent from the variables to be learned. Maximizing the log-posterior over the hidden variables with fixed hyper-parameters (i.e., the observation noise variances and prior variances) is equivalent to minimizing the following objective function:

$$L(R,S,U,V,P,Z,\mathbf{E}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}^{R}\big(r_{ij} - g(U_i^T V_j + P_i^T \mathbf{E} f_j)\big)^2$$
$$+ \frac{\lambda_S}{2}\sum_{j=1}^{m}\sum_{k=1}^{m} I_{jk}^{S}\big(s_{jk} - g(V_j^T Z_k)\big)^2 + \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_V}{2}||V||_F^2 + \frac{\lambda_P}{2}||P||_F^2 + \frac{\lambda_Z}{2}||Z||_F^2 + \frac{\lambda_{\mathbf{E}}}{2}||\mathbf{E}||_F^2 \tag{3.10}$$

where $\lambda_S = \sigma_R^2/\sigma_S^2$, $\lambda_U = \sigma_R^2/\sigma_U^2$, $\lambda_V = \sigma_R^2/\sigma_V^2$, $\lambda_P = \sigma_R^2/\sigma_P^2$, $\lambda_Z = \sigma_R^2/\sigma_Z^2$, $\lambda_{\mathbf{E}} = \sigma_R^2/\sigma_{\mathbf{E}}^2$ and $||\cdot||_F^2$ denotes the Frobenius norm. Note that λ_S is a balancing parameter that regulates the importance of *"also-viewed"* products in the unified model. Having formulated a non-convex objective function as Eq.3.10, we compute the gradient of each embedding variable, i.e., U_i, V_j, P_i, Z_k, $\mathbf{E}$, and learn them by gradient descent to obtain a local minimum solution. Detailed equations of the gradients for Eq.3.10 and the time complexity of VMCF are given as follows:

Gradients of Equation 3.10

$$\begin{aligned}
\frac{\partial L}{\partial U_i} &= \sum_{j=1}^{m} I_{ij}^{R}(g(\hat{r}_{ij}) - r_{ij})g'(\hat{r}_{ij})V_j + \lambda_U U_i \\
\frac{\partial L}{\partial V_j} &= \sum_{i=1}^{n} I_{ij}^{R}(g(\hat{r}_{ij}) - r_{ij})g'(\hat{r}_{ij})U_i \\
&\quad + \lambda_S \sum_{k=1}^{m} I_{jk}^{S}(g(\hat{s}_{jk}) - s_{jk})g'(\hat{s}_{jk})Z_k + \lambda_V V_j \\
\frac{\partial L}{\partial P_i} &= \sum_{j=1}^{m} I_{ij}^{R}(g(\hat{r}_{ij}) - r_{ij})g'(\hat{r}_{ij})\mathbf{E}f_j + \lambda_P P_i \\
\frac{\partial L}{\partial Z_k} &= \lambda_S \sum_{j=1}^{m} I_{jk}^{S}(g(\hat{s}_{jk}) - s_{jk})g'(\hat{s}_{jk})V_j + \lambda_Z Z_k \\
\frac{\partial L}{\partial \mathbf{E}} &= \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}^{R}(g(\hat{r}_{ij}) - r_{ij})g'(\hat{r}_{ij})P_i f_j^T + \lambda_{\mathbf{E}}\mathbf{E}
\end{aligned} \tag{3.11}$$

Complexity Analysis The overall complexity of P3S_1 is composed of the calculation of both Eqs. 3.10 and 3.11. Considering the sparseness of $\mathcal{R}$ and $\mathcal{S}$, the computation Eq. 3.10 has complexity $O(\rho(L+D)+\mu L)$, where ρ denotes the average number of observed ratings in $\mathcal{R}$, and μ denotes the average number of observed elements in $\mathcal{S}$. Next, for the gradients in Eq. 3.11, computing $\frac{\partial L}{\partial U_i}$, $\frac{\partial L}{\partial V_j}$, $\frac{\partial L}{\partial P_i}$, $\frac{\partial L}{\partial Z_k}$ and $\frac{\partial L}{\partial \mathbf{E}}$ incur complexity $O(\rho L)$, $O(\rho L+\mu L)$, $O(\rho D\bar{C}) = O(\rho D)$, $O(\mu L)$ and $O(\rho D\bar{C}) = O(\rho D)$, respectively, where $\bar{C}$ is the average number of non-zero

elements in a CNN feature (f_j). Note that $\bar{C}$ is a small value, since the CNN feature is very sparse. Thus, we obtain a total complexity of $O(\rho(L+D)+\mu L)$. This analysis indicates that the complexity of P3S_1 is linear in the number of embedding dimensions.

Reduced Model Note that when images of products are not available, the final objective function is reduced to:

$$L(R,S,U,V,Z) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}^{R}\big(r_{ij} - g(U_i^T V_j)\big)^2 + \frac{\lambda_S}{2}\sum_{j=1}^{m}\sum_{k=1}^{m} I_{jk}^{S}\big(s_{jk} - g(V_j^T Z_k)\big)^2 + \frac{\lambda_U}{2}||U||_F^2 + \frac{\lambda_V}{2}||V||_F^2 + \frac{\lambda_Z}{2}||Z||_F^2 \tag{3.12}$$

As previously mentioned, in domains where appearance of a product is not significant for modeling user ratings, such product aspects as functionality and specifications are more influential to user ratings than appearance. Thus, the reduced model as in Eq. 3.12 can be still beneficial, especially for the product domains where the appearance is not important. We dub this model Matrix Co-Factorization (MCF), and demonstrate the benefit of the reduced model in product domains where appearance is not important in Section 3.6.2.

3.6 Experiments

In this section, we conduct experiments to verify the superiority of our methods by comparing their performance with several state-of-the-art methods on multiple real-world datasets. The experiments are designed to verify the following questions:

Q.1 How do MCF and VMCF perform compared with other competitors in both the visually-aware (Boys' and Girls' Clothing) and the visually non-aware product domains (Automotive, Pet Supplies and Office Products)?

Q.2 By leveraging *"also-viewed"* products, do we indeed obtain high-quality product embedding?

Q.3 How does model parameter λ_S and the number of embedding dimensions L affect the user rating prediction accuracy?

Table 3.2: Data Statistics. #Relations implies the number of edges in the *product-affinity network*

Dataset	#Users	#Prod.	#Ratings	#Relations
Boys' Clothing	4,496	6,391	15,997	31,370
Girls' Clothing	5,941	9,549	22,524	51,990
Automotive	84,418	126,934	406,852	2,162,853
Pet Supplies	85,115	49,048	427,543	1,066,131
Office Prod.	50,570	40,181	240,146	672,586

Table 3.3: Properties of methods being compared.

Baselines	Personalized?	Visually-Aware?	Incorporate *"also-viewed"*?
ItemCF	X	X	X
PMF	O	X	X
VMF	O	O	X
MCF	O	X	O
VMCF	O	O	O

3.6.1 Experimental Setting

Datasets We use five public datasets[2] extracted from *Amazon.com* by McAuley *et al.* [16]. The datasets include user ratings data and product metadata, which includes URLs for product image, price, a list of also bought product, a list of also viewed products, and etc. Among these metadata, we use the lists of also viewed products to construct our *product-affinity network*. Aiming at demonstrating the benefit of our model in general product domains, we not only use two datasets (*Boys' Clothing and Girls' Clothing*) from domains where the appearance of a product is significant, but also three datasets (*Automotive, Pet Supplies and Office Products*) from domains where other product aspects, such as product functionality and specifications, play a more significant role than the appearance. We preprocess all datasets so that each user rated at least three products. Moreover, every product in the *product-affinity network* also exists in the user ratings data. Table 6.1 shows the detailed statistics of the datasets.

Comparison Methods

- **ItemCF:** A traditional recommendation method based on the similarity of products [46], where Pearson's correlation coefficient is used as similarity measure.

[2]http://jmcauley.ucsd.edu/data/amazon/

- **PMF:** Matrix factorization-based recommendation method that considers only user ratings information [76].

- **VMF:** A visually-aware matrix factorization-based method that incorporates product images but not *"also-viewed"* product information. We use the rating prediction model introduced in [14], from which bias terms are excluded to clearly investigate the benefit of incorporating product images themselves. The rating assigned by user i to product j is modeled as $\hat{r}_{ij} = U_i^T V_j + P_i^T \mathbf{E} f_j$, and the objective function to minimize is formulated as:

$$L = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} I_{ij}^{R} (r_{ij} - g(\hat{r}_{ij}))^2 + \Omega(U, V, P, \mathbf{E})$$

  where $\Omega(\cdot)$ is the regularization to avoid model overfitting.

In order to provide a clear understanding of baseline methods, we provide a summary of their properties in Table 3.3. Furthermore, these baselines are chosen for the following reasons:

1. **PMF vs. VMF**

   a) To verify the benefit of incorporating product images in both visually-aware and visually non-aware product domain, and b) To verify that appearance is more significant in clothing domain than other domains.

2. **VMF vs. MCF**

   To verify that other aspects besides appearance are more significant in visually non-aware product domain than in visually-aware product domain.

3. **VMF, MCF vs. VMCF**

   To demonstrate the benefit of jointly modeling product images and *"also-viewed"* product information in both visually-aware and visually non-aware product domain.

Evaluation Metric We employ the Mean Squared Error (MSE), a metric that has been commonly used for evaluating the performance of user rating prediction on the *Amazon* dataset [87, 41, 13]. The MSE is defined as:

$$MSE = \frac{\sum_{i,j} (r_{ij} - \hat{r}_{ij})^2}{N} \tag{3.13}$$

where r_{ij} denotes the rating that user i assigned to product j, $\hat{r}_{ij}$ denotes the corresponding predicted rating, and N denotes the number of ratings in the test

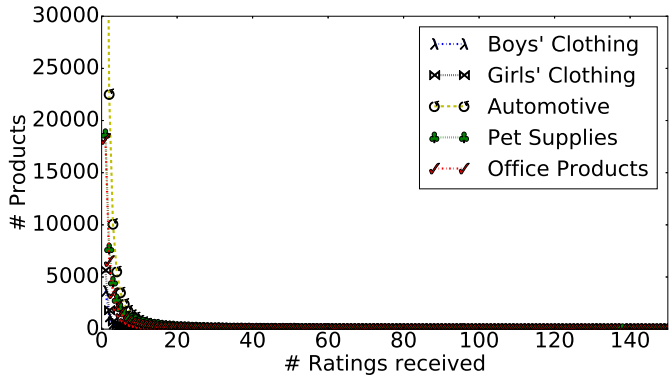


Figure 3.3: Skewness of the number of rated products in each dataset. Most products received very few ratings.

dataset. The MSE is an appropriate metric for our experiments because the objective functions of the methods being compared are originally designed to minimize the MSE.

Evaluation Protocol We randomly sample 80% of the user ratings datasets for training, and the remaining 20% is used for testing. Random sampling is independently conducted five times, and we evaluate the baselines and our methods on each dataset. Finally, we report here the mean and standard deviation (error bar) of the MSE on each test dataset.

Cold-start Evaluation In addition to the *"All"* setting where we evaluate on all the ratings in the test dataset, we also evaluate our methods on the *"Cold-start"* setting where ratings in the test dataset are sampled such that each product (cold-product) has fewer than four ratings in the training dataset. Note that as shown in Figure 3.3, most products received very few ratings. Precisely, cold-products constitute approximately 85% in Boys' Clothing, Girls' Clothing and Automotive, 70% in Pet Supplies and Office Products. These statistics indicate that most products are cold-products in the real-world online shopping environment, and thus *evaluations on cold-products are in fact more crucial than evaluations on every product.*

3.6.2 Performance Analysis

Figure 3.4 summarizes the evaluation results on each dataset in terms of MSE where performance is evaluated under two settings, i.e., *"All"* and *"Cold-start"*. Note that visually-aware product domain denote Boys' Clothing and Girls' Cloth-

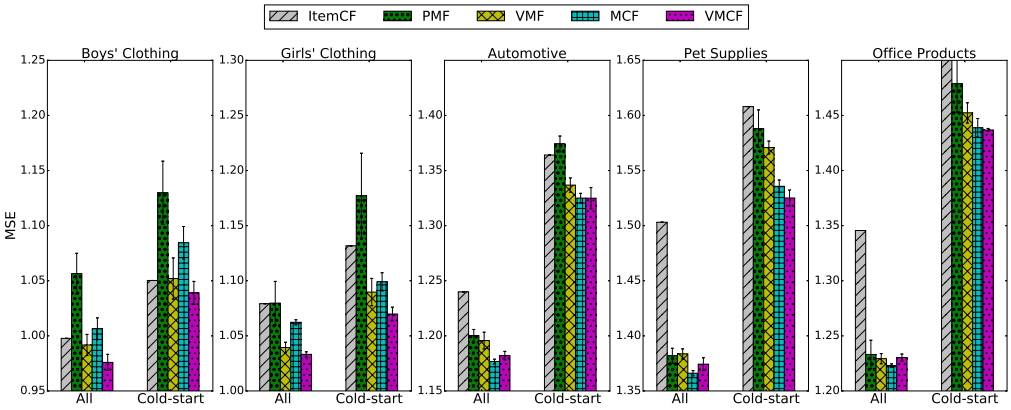


Figure 3.4: Performance comparison in terms of MSE. Lower MSE implies better performance.

ing, and visually non-aware product domain denote Automotive, Office Products and Pet Supplies.

1) Benefit of visual features in both domains (PMF vs. VMF)
We observe that in both the visually-aware and the visually non-aware product domain, *incorporating visual features helps to improve user rating prediction accuracy*, while the improvement is more significant in visually-aware product domain. This agrees with our expectations whereby a) appearance plays a more significant role in modeling user ratings in visually-aware product domain, and b) although other product aspects are more influential in visually non-aware product domain, visual features are indeed still helpful in modeling user ratings. Note that the *performance gain of VMF compared with PMF becomes more significant under the "Cold-start" setting*, which indicates that visual features become more valuable when a product only has a few ratings.

2) Varying significance of product aspects in different domains (VMF vs. MCF)
We observe that *in the visually-aware product domain, the user rating prediction accuracy of VMF outperforms MCF, whereas MCF outperforms VMF in the visually non-aware product domain.* This agrees with **Example 1** in Section 3.2, in that appearance plays the most significant role in visually-aware product domain, whereas other product aspects beyond appearance play a more significant role in visually non-aware product domain.

3) Benefit of jointly modeling product images and *"also-viewed"* product information (VMF, MCF vs. VMCF)

We observe that *in the visually-aware product domain, VMCF outperforms both VMF and MCF.* This indicates that in the visually-aware product domain, other aspects beyond appearance of products are also taken into account by modeling the *"also-viewed"* relationship, which in turn yields further improvements in user rating prediction accuracy. Such a benefit becomes clearer in the *"Cold-start"* setting.

Meanwhile, *under the "All" setting in the visually non-aware product domain, VMCF fails to outperform MCF.* This implies that when a sufficient number of ratings is assigned to each product, visual features in the visually non-aware product domain are in fact considered as *noise*, and thus degrading user rating prediction accuracy. However, it is worth noting that *under the "Cold-start" setting, the performance of VMCF is comparable to that of MCF or even better in Pet Supplies and Office Products.* This indicates that visual features remain helpful even in the visually non-aware product domain, when we are provided with only a few rated products.

3.6.3 Qualitative Analysis

Quality of product embedding Besides the performance evaluations in terms of the accuracy of user rating prediction, we additionally evaluate the quality of product embedding generated by our proposed method in comparison with VMF. To do so, we perform classification on both product embedding $V \in \mathcal{R}^{L \times m}$ and product visual embedding $Q \in \mathcal{R}^{D \times m}$. In order to demonstrate that the classification results are reliable and robust, we select four classification algorithms, i.e., Logistic regression (LR), Support vector machine (SVM), Random forest (RF) and Gradient boosting (GB). The input products for classification algorithms belong to the top-10 most frequently appeared categories in each dataset, and each product is labeled by its corresponding category. We perform five-fold cross-validation and report the mean and standard deviation (error bar). The followings are the top-10 categories used in this experiment:

Top-10 Most Frequent Categories

- *Boys' Clothing:* Athletic, Boots, Sneakers, Tops & Tees, Costumes & Accessories, Sandals, Kids & Baby, Pants, Shirts, Hoodies
- *Girls' Clothing:* Boots, Athletic, Sneakers, Costumes & Accessories, T-Shirts, Jewelry, Kids & Baby, Sandals, Special Occasion, Playwear

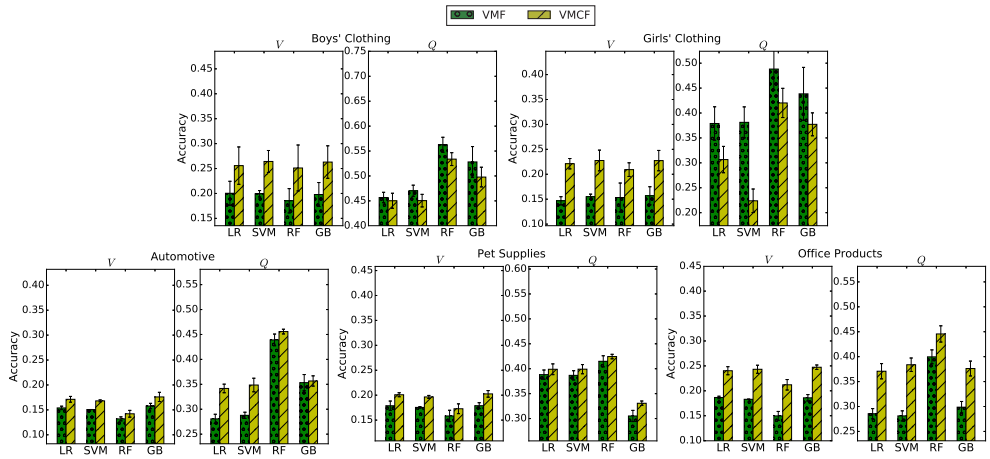


Figure 3.5: Classification results on product embedding V and product visual embedding Q.

- *Automotive:* 'Shocks, Struts & Suspension', 'Paint, Body & Trim', Filters, Brake System, Protective Gear, Bulbs, Decals & Bumper Stickers, Floor Mats & Cargo Liners, Towing Products & Winches, Car Care
- *Pet Supplies:* 'Collars, Harnesses & Leashes', Food, Health Supplies, Toys, Treats, Apparel & Accessories, Pumps & Filters, Beds, Carriers & Travel Products, Shampoos & Conditioners
- *Office Products:* Pens & Refills, Paper, Inkjet Printer Ink, 'Labels, Indexes & Stamps', Laser Printer Toner, Office Furniture & Lighting, 'Envelopes, Mailers & Shipping Supplies', Notebooks & Writing Pads, Telephones & Accessories, 'Forms, Recordkeeping & Money Handling'

Figure 3.5 shows the results of classification. We observe that in the visually non-aware product domain, classification accuracy on product embedding V and the product visual embedding Q generated by VMCF outperform those generated by VMF. This indicates that *VMCF generates more high-quality embeddings than VMF, which eventually results in higher accuracy in user rating prediction* as shown in Figure 3.4. Meanwhile, in the visually-aware product domain, while VMCF outperforms VMF in the user rating prediction task, the classification accuracy of VMCF on V is higher than that of VMF, but its the accuracy on Q is lower. We conjecture that using only product images is helpful for the product classification task because visual features are generated by a pre-trained CNN originally designed for image classification task. In contrast, for our user rating prediction task, *product embedding V, which models the "also-viewed" product information where various product aspects are reflected, works in synergy with Q to improve user rating prediction accuracy.*

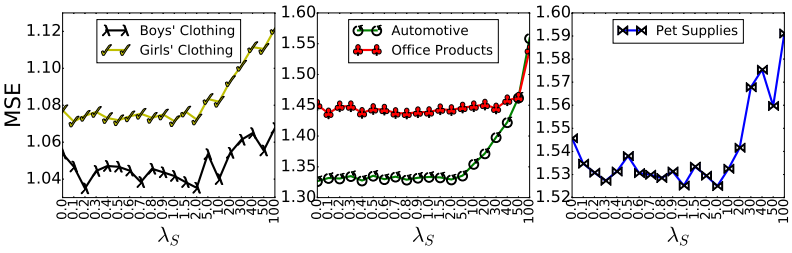


Figure 3.6: Impact of Parameter λ_S on VMCF.

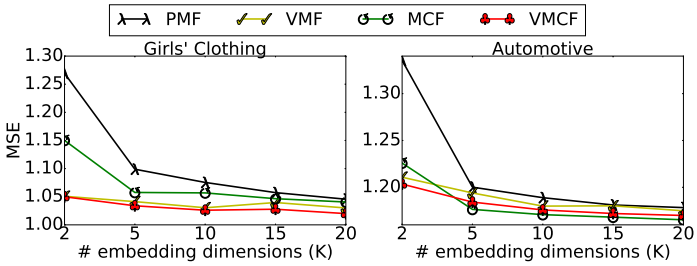


Figure 3.7: Results of various number of embedding dimensions L.

3.6.4 Sensitivity Analysis

Influence of balancing parameter λ_S

λ_S is a hyper-parameter that regulates the importance of *"also-viewed"* product information in our proposed method
VMCF. When $\lambda_S = 0$, we ignore *"also-viewed"* product information, whereas when $\lambda_S = \infty$, we only exploit *"also-viewed"* product information. Figure 3.6 shows the evaluation results of VMCF performed on each dataset under the *"Cold-start"* setting. We observe that incorporating *"also-viewed"* product information indeed improves user rating prediction performance, and the optimal value of λ_S is different in each dataset, which is mostly a value in the range [0.0, 1.0].

Influence of num. dimensions L of product embedding

Figure 3.7 shows the evaluation results on the test set of Girls' Clothing and Automotive over different numbers of embedding dimensions L under the *"All"* setting, given a fixed $D = 5$. We observe that for all methods (PMF, VMF, MCF and VMCF) the value of MSE decreases as the number of latent dimensions increases. However, since the total complexity of VMCF is linear in the number of embedding dimensions as demonstrated in Section 3.5.3, a proper value of L should be found such that complexity is practically acceptable within computational limitations. Evaluations on other datasets yield similar results,

and are thus omitted here for brevity.

3.7 Conclusion

Every product domain has dominant aspects that are more influential to user ratings than others. In clothing domain, the appearance of products plays the most significant role, whereas in other domains such as Automotive, Pet Supplies and Office Products, other aspects such as product functionality and specifications are more influential. In this paper, we propose a matrix co-factorization framework that jointly factorizes user ratings data and *"also-viewed"* product information that reflects various product aspects that are varyingly influential to user ratings in different product domains. We empirically show that this information is helpful for user rating prediction by generating more high-quality product embedding, especially under the *"Cold-start"* setting. Our method is useful for online retailers such as *Amazon* and *eBay*[3], where product images are provided to customers and their browsing histories are collected.

[3] http://www.ebay.com/

IV. Recommendation by Jointly Modeling Click and Purchase records

4.1 Overview

In this paper, we focus on dealing with the missing user–item interactions of purchase records, i.e., non-purchased items, and propose to leverage users' past click records, which not only reveal users' broad interests but are also considered as a context in which purchases have been made. More precisely, we perform experiments on various models, each assuming a different order of user preferences among purchased, clicked-but-not-purchased and non-clicked items, to study whether it is beneficial to leverage click records to complement the missing user–item interactions of purchase records. We postulate that the exploitation of click records for ranking needs a carefully designed model because these records are noisy. To address this issue, we propose a novel pairwise learning-to-rank method that 1) is robust to noisy click records, and 2) focuses on the accuracy of top-ranked items. The experimental results on two real-world e-commerce datasets demonstrate that our proposed method significantly outperforms the state-of-the-art implicit feedback-based recommendation methods, especially for top-ranked items.

4.2 Introduction

Implicit feedback, such as purchases and clicks, are easily obtained from system logs, but precisely eliciting users' preferences from implicit feedback for purchase prediction is challenging because negative feedback is not explicitly observed. In this respect, past research has focused on inferring users' negative feedback from missing user–item interactions. Specifically, a uniform weighting scheme [88, 26] in which all missing data are treated as negative feedback (i.e., *All Missing As Negative (AMAN)* assumption) has been introduced. However, this assumption is not entirely valid in that the reason why items are not observed is uncertain; whether a user does not like them or a user is simply not aware of them. To cope with the drawback of the *AMAN* assumption, sampling–based approaches such as user-oriented sampling [52] or item-popularity-oriented sampling [89, 90] have been proposed. However, the sampling–based approaches are

essentially based on predefined heuristic weights [91] that are not guaranteed to always hold in the real data.

In this paper, we present a framework that leverages users' past *click records* to complement the missing user–item interactions of purchase records, i.e., non-purchased items, aiming at *purchase prediction*. Precisely, we leverage users' past click records in conjunction with their purchase records, both of which are easily collected by e-commerce stores. Intuitively, click records reveal users' general interest because users click on numerous items before making purchases. Hence, we expect that users' click records will complement the missing user–item interactions of purchase records in a more data-driven manner compared with previous uniform weighting scheme or sampling–based approaches.

With the help of click records, we begin by formulating various model assumptions regarding the order of user preferences among the missing user–item interactions of purchase records, i.e., non-purchased items, which can be split into two disjoint sets; *clicked-but-not-purchased* items and *non-clicked* items. We empirically demonstrate that a model assumption in which users are assumed to prefer *purchased* (P) items to *clicked-but-not-purchased* (CBNP) items to *non-clicked* (NC) items, is beneficial for purchase prediction when implemented under the Bayesian Personalized Ranking (BPR) model [26], which is a pairwise bipartite ranking model that maximizes the AUC metric. To be precise, we make three different positive−negative pairs over three disjoint itemsets, i.e., P−CBNP, CBNP−NC and P−NC, and learn a ranking function that is expected to establish a total order in which positive instances precede negative ones in each positive−negative pair of itemsets, which is equivalent to maximizing the AUC.

However, click records are inherently noisier than purchase records in practice; a user may accidentally click on wrong items or may click on items to see more details and end up not liking it, whereas a user is more confident with purchased items. To make the matter worse, the number of click records greatly exceeds that of purchase records, implying that the bipartite ranking model such as BPR can be dominated by the noisy click records. Therefore, naively incorporating click records for the bipartite ranking can be detrimental to the performance of recommendation[1], and the model should be meticulously designed to properly harness the click records for purchase prediction under bipartite ranking. To this end, we propose a novel learning-to-rank method for purchase prediction, called

[1]Since the goal of recommender systems in e-commerce is to recommend items that are likely to be purchased by users, the purchase prediction can be cast as the task of item recommendation. Therefore, we use the terms, i.e., "purchase prediction" and "recommendation" interchangeably throughout this paper.

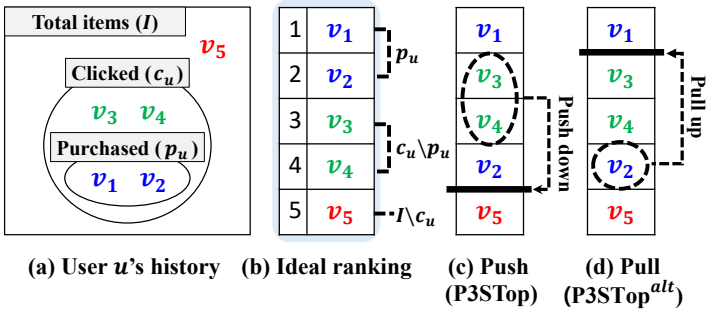


Figure 4.1: A toy example of the push/pull mechanism.

P3STop, that is customized to be robust to noisy click records. More precisely, P3STop minimizes the number of "negative" items ranked above the last-ranked "positive" item. As a concrete example, consider the following **Toy Example** in which we illustrate the push/pull mechanism of modeling the pairwise relationship between purchased ("positive") items, and clicked-but-not-purchased ("negative") items.

Toy Example. Figure 4.1a shows user u's interaction history with items $(v_1, v_2, v_3, v_4, v_5)$, and the ideal ranking list for user u is displayed in Figure 4.1b. That is, for user u, we want to train our model so that the items are ordered in the following order at the end of the model training: purchased items ($\mathbf{p}_u$), clicked-but-not-purchased items ($\mathbf{c}_u \backslash \mathbf{p}_u$), non-clicked items ($\mathcal{I} \backslash \mathbf{c}_u$). Assuming that items are incorrectly ranked as in Figure 4.1c during the training process, we aim to push down as many incorrectly ranked clicked-but-not-purchased ("negative") items, i.e., v_3, v_4, below the *bound set by the last-ranked purchased ("positive") item*, i.e., v_2. In other words, we push down the possibly noisy clicked items below the *bound set by a solid purchased item*, which makes our model more robust to noisy click records. As an alternative to the push mechanism described in Figure 4.1c, the pull mechanism shown in Figure 4.1d differs in the way that the bound is set. Precisely, it pulls up the purchased items, i.e., v_2, above the *bound set by the first-ranked ("negative") clicked-but-not-purchased item*, i.e., v_3, as shown in Figure 4.1d. This method is prone to being dominated by noisy click records, because the *bound is set by the possibly noisy clicked item*. In Section 4.5 and 4.5.1, we will describe the rationale behind each case[2].

It is important to note that the above push mechanism in Figure 4.1c enables the model to *particularly focus on the accuracy of the top-ranked items.*

[2]While the above push/pull mechanism is applied to the following pairs of itemsets, i.e., $(\mathbf{p}_u \leftrightarrow \mathbf{c}_u \backslash \mathbf{p}_u), (\mathbf{c}_u \backslash \mathbf{p}_u \leftrightarrow \mathcal{I} \backslash \mathbf{c}_u)$, and $(\mathbf{p}_u \leftrightarrow \mathcal{I} \backslash \mathbf{c}_u)$, we display here only the foremost pair for brevity.

More precisely, the upper bound of clicked-but-not-purchased items (v_3, v_4) is set to the last-ranked purchased item (v_2) in which the user is more confident with than any clicked-but-not-purchased item. In this regard, the bound set by a purchased item (Figure 4.1c) should be relatively high and robust compared with the bound set by a clicked item (Figure 4.1d). Therefore, pushing down the incorrectly ranked clicked-but-not-purchased items below the last-ranked purchase item allows greater focus on the accuracy of the top-ranked items. We argue that our proposed method generates more practical recommendation results for users, since the top-ranked items get much more attention by users in practice [92]. However, only a few recent studies have particularly considered it for the task of recommendation [93, 94, 95].

Our main contributions are summarized as follows:

1. To complement the missing user–item interactions of purchase records, we formulate various model assumptions regarding the order of user preferences among non-purchased items by taking the click records into account (**Section 4.4**).

2. After we find a valid model assumption under the BPR model, we propose P3STop that is customized to be robust to noisy click records by particularly focusing on the accuracy of the top-ranked items. (**Section 4.5**).

3. Experimental results on two real-world e-commerce datasets demonstrate that P3STop considerably outperforms the state-of-the-art implicit feedback–based recommendation methods, especially for the top-ranked items. (**Section 4.6**).

It is worth noting that click records have been used for various tasks such as click-through rate (CTR) prediction in online advertising [96, 97, 98] and Twitter [99], user intent prediction [100, 101], repeat-buyer prediction [102], conversion response prediction in display advertising [103], and session–based click prediction [104]. However, not much effort has been devoted to purchase prediction, and to the best of our knowledge, our work is the first to propose a framework that leverages click records to complement the missing user–item interactions of purchase records.

4.3 Problem Statement

Let $\mathcal{U}$ and $\mathcal{I}$ be the set of users and items, respectively, and we have n users and m items. The purchase records of users in $\mathcal{U}$ on items in $\mathcal{I}$ are represented by the purchase matrix $\mathbf{P} = [p_{ui}]_{n \times m}$, where $p_{ui} = 1$ if user u purchased item i, and 0 otherwise. Likewise, the click records of users in $\mathcal{U}$ on items in $\mathcal{I}$ are

represented by the click matrix $\mathbf{C} = [c_{ui}]_{n \times m}$, where $c_{ui} = 1$ if user u clicked item i, and 0 otherwise; counts are ignored in this work. $\mathbf{p}_u$ and $\mathbf{c}_u$ denote the sets of items purchased and clicked by user u, respectively.

Problem Definition

Given: The purchase matrix $\mathbf{P}$ and click matrix $\mathbf{C}$,

Goal: To recommend items $i \in \mathcal{I}n(\mathbf{p}_u \cup \mathbf{c}_u)$ to each user $u \in \mathcal{U}$; among items that the user has not previously interacted with (neither purchased nor clicked).

4.4 Ordering User Preferences among Non-purchased Items

In this section, we describe our framework that leverages click records to complement the missing user–item interactions of purchase records. i.e., non-purchased items. We begin by explaining our model assumptions regarding the order of user preferences among non-purchased items (**Section 4.4.1**). Next, we describe how our model assumptions are implemented under the BPR model (**Section 4.4.2**). Then, we discuss two shortcomings of naively incorporating click records under the BPR model (**Section 4.4.3**).

4.4.1 Defining the Model Assumptions

Recall the *AMAN* assumption made by previous pairwise learning-to-rank methods [26, 61, 105].

AMAN Assumption We assume that a user prefers *purchased* items to *non-purchased* items.

$$i \succ_u j, \text{if } i \in \mathbf{p}_u \wedge j \in \mathcal{I} \backslash \mathbf{p}_u \tag{4.1}$$

Eqn. 4.1 implies that user u prefers purchased items i to non-purchased items j. However, this assumption is oversimplified in that all non-purchased items are equally considered as negative feedback, whereas in reality some of the non-purchased items attract the user more than the others.

To overcome the above limitation of the *AMAN* assumption, we incorporate users' click records that reveal users' general interest, assuming that users click on numerous items before making purchases. Although the user preference reflected therein is not as strong as in purchase records, we expect that *click records will complement the missing user–item interactions of purchase records.* To this end, given *purchased* items, we split the non-purchased items into two disjoint sets, i.e., *clicked-but-not-purchased* items and *non-clicked* items, by using click records, and introduce three different model assumptions regarding the order of

user preferences among them. For each user u, we assume $\mathbf{p}_u \in \mathbf{c}_u \in \mathcal{I}$, i.e., all purchased items are selected from clicked items.

Assumption 1. We assume that a user prefers *purchased* items to *non-clicked* items.

$$i \succ_u j, \text{if } i \in \mathbf{p}_u \wedge j \in \mathcal{I} \backslash \mathbf{c}_u \tag{4.2}$$

Instead of regarding non-purchased items as negative feedback as in Eqn. 4.1, this time we regard non-clicked items as negative feedback. This narrows down the candidates for negative feedback, i.e., from $\mathcal{I}\backslash\mathbf{p}_u$ to $\mathcal{I}\backslash\mathbf{c}_u$, which is expected to ameliorate the *AMAN* assumption.

Assumption 2. We assume that a user prefers *purchased* items to *clicked-but-not-purchased* items, and *clicked-but-not purchased* items to *non-clicked* items.

$$i \succ_u j,\ j \succ_u k,\ i \succ_u k, \text{if } i \in \mathbf{p}_u \wedge j \in \mathbf{c}_u \backslash \mathbf{p}_u \wedge k \in \mathcal{I} \backslash \mathbf{c}_u \tag{4.3}$$

We extend **Assumption 1** by adding another set of items. i.e., clicked-but-not-purchased items ($\mathbf{c}_u\backslash\mathbf{p}_u$). Eqn. 4.3 is based on the assumption that 1) user u is more confident with purchased items ($\mathbf{p}_u$) than to clicked-but-not-purchased items ($\mathbf{c}_u\backslash\mathbf{p}_u$), because users generally decide to purchase items over many other candidates ($\mathbf{c}_u\backslash\mathbf{p}_u$) that reveal users' general interest, which implies that 2) user u prefers clicked-but-not-purchased items to the items that are neither purchased nor clicked ($\mathcal{I}\backslash\mathbf{c}_u$).

Assumption 3. We assume that a user prefers *purchased* items to *clicked-but-not-purchased* items, and *non-clicked* items to *clicked-but-not-purchased* items.

$$i \succ_u j,\ k \succ_u j,\ i \succ_u k, \text{if } i \in \mathbf{p}_u \wedge j \in \mathbf{c}_u \backslash \mathbf{p}_u \wedge k \in \mathcal{I} \backslash \mathbf{c}_u \tag{4.4}$$

Eqn. 4.4 implies that user u dislikes items that are clicked-but-not-purchased ($\mathbf{c}_u\backslash\mathbf{p}_u$) more than those that are not clicked at all ($\mathcal{I}\backslash\mathbf{c}_u$). This assumption is also intuitive in the sense that although being aware of clicked-only items ($\mathbf{c}_u\backslash\mathbf{p}_u$), the fact that the user still chose not to purchase them implies that the user dislikes them.

4.4.2 Verifying the Model Assumptions

To figure out which of our three model assumptions (Eqn. 4.2,4.3,4.4) is valid, we implement them under the BPR model [26], and name each of them P3S_1, P3S_2 and P3S_3, respectively (P3S stands for modeling pairwise relationships among three disjoint item sets). We only present here the equation for P3S_2, which is based on **Assumption 2**. The equations for P3S_1 and P3S_3 are

similarly formulated and hence omitted. Given parameters Θ for training, we maximize the following likelihood function for P3S_2:

$$\mathcal{J}_{\text{P3S\_2}}(\Theta) = \prod_{u \in \mathcal{U}} \Big(\prod_{i \in \mathbf{p}_u} \prod_{j \in \mathbf{c}_u \setminus \mathbf{p}_u} \Pr[i \succ_u j] \prod_{j \in \mathbf{c}_u \setminus \mathbf{p}_u} \prod_{k \in \mathcal{I} \setminus \mathbf{c}_u} \Pr[j \succ_u k] \prod_{i \in \mathbf{p}_u} \prod_{k \in \mathcal{I} \setminus \mathbf{c}_u} \Pr[i \succ_u k] \Big) \tag{4.5}$$

where $\Pr[i \succ_u j] = \sigma(\hat{x}_{ui} - \hat{x}_{uj})$ denotes the probability that user u prefers item i to item j, which is approximated by a sigmoid function of the form $\sigma(x) = 1/(1+e^{-x})$ [26], and $\hat{x}_{ui} = \alpha_u^T \beta_i + \gamma_i$ denotes the predicted preference of user u on item i computed by matrix factorization (MF); $\alpha_u \in \mathbb{R}^K$ and $\beta_i \in \mathbb{R}^K$ represent the K-dimensional latent factors for user u and item i, respectively, and $\gamma_i \in \mathbb{R}$ denotes the item bias term for item i. For more details of the optimization process, refer to the original paper [26] that proposed the BPR model. We later show in our experiments (Table 4.1) that P3S_2 outperforms P3S_1 and P3S_3, which implies that **Assumption 2** is the most valid model assumption.

Note that other scoring functions such as neural network (NN)–based functions [24, 104] can also be applied to our framework by simply replacing MF. However, as our focus is to propose a "framework" that can properly utilize click records for purchase prediction rather than to prove the superiority of NN over MF, we conduct experiments with MF as our scoring function in this paper.

4.4.3 Discussion: Shortcomings of P3S_2

Although P3S_2 is shown to be beneficial for purchase prediction when implemented under the BPR model, it has two shortcomings. The first shortcoming is caused by the noisiness of click records, whose amount even greatly exceeds that of purchase records. Unlike purchases, clicks can occur even without a user's intent to purchase; a user may accidentally click on wrong items or click on items out of simple curiosity, whereas a user is more confident with purchased items. That is to say, the click records are noisier in nature, and are more likely to be non-relevant to user preferences than the purchase records. Therefore, we argue that relying too much on the click records would be detrimental to the performance of recommendation. However, since BPR was developed for bipartite ranking in which every possible positive–negative instance pair is taken into account, the model can be easily dominated by the noisy click records as their amount greatly exceeds that of purchase records. The second shortcoming is caused by the objective of the BPR model. Although users are mainly interested in top-ranked items [92], BPR maximizes the AUC, which gives an equal weight to each training instance regardless of its position in the list. In other words, a mistake in the higher part of the recommendation list is equally penalized with one in the lower

part, implying that optimizing the AUC does not allow a particular focus on the accuracy of the top-ranked items. Therefore, in the following section, we propose a novel method that simultaneously addresses the above shortcomings.

4.5 The Proposed Method: P3STop

Here, we describe our novel learning-to-rank method, P3STop, that is customized to be *robust to relatively unreliable click records* by particularly *focusing on the accuracy of the top-ranked items*. Since P3S_2, which is based on **Assumption 2**, turned out to be the most valid model (Table 4.1), we adopt it as the underlying assumption of our proposed method, P3STop, hereinafter. Note that under **Assumption 2**, P and NC items are always regarded as positive and negative items, respectively. In contrast, CBNP items can be considered as either positive or negative items, depending on which pair of itemsets we are interested in.

Model Formulation. For each user u, we compute the sum of the number of 1) CBNP items ranked above the least relevant P item, 2) NC items ranked above the least relevant CBNP item, and 3) NC items ranked above the least relevant P item, and minimize the following:

$$\begin{aligned}\mathcal{L}_{\mathsf{P3STop}}(u) = &\frac{1}{|\mathbf{c}_u \backslash \mathbf{p}_u|} \sum_{j \in \mathbf{c}_u \backslash \mathbf{p}_u} \mathbb{I}\left[\min_{i \in \mathbf{p}_u} \hat{x}_{ui} \leq \hat{x}_{uj}\right] \\ &+ \frac{1}{|I \backslash \mathbf{c}_u|} \left[\sum_{k \in I \backslash \mathbf{c}_u} \mathbb{I}\left[\min_{j \in \mathbf{c}_u \backslash \mathbf{p}_u} \hat{x}_{uj} \leq \hat{x}_{uk}\right] + \sum_{k \in I \backslash \mathbf{c}_u} \mathbb{I}\left[\min_{i \in \mathbf{p}_u} \hat{x}_{ui} \leq \hat{x}_{uk}\right] \right]\end{aligned} \tag{4.6}$$

where $\hat{x}_{ui} = \alpha_u^T \beta_i$ and $\mathbb{I}[\cdot]$ is the indicator function. Note that in Eqn. 4.6 the bound is set with respect to positive items and thus more emphasis is placed on positive items than on negative items, making our model robust to relatively unreliable negative items. For example, consider the first term, where for user u, we set the upper bound of the CBNP items ($j \in \mathbf{c}_u \backslash \mathbf{p}_u$) to the score of the last-ranked P item ($\min_{i \in \mathbf{p}_u} \hat{x}_{ui}$) (Figure 4.1c). Although the last-ranked P item has the lowest score among all P items $\mathbf{p}_u$, its score $\min_{i \in \mathbf{p}_u} \hat{x}_{ui}$ should be higher than the score $\hat{x}_{uj}$ of any CBNP item $j \in \mathbf{c}_u \backslash \mathbf{p}_u$ because it was specifically chosen by user u from all the clicked items (**Assumption 2**). Consequently, the upper bound set by the last-ranked positive item will be high enough so that we obtain positive items near the top by pushing down relatively unreliable negative items below it. In summary, by minimizing $\mathcal{L}_{\mathsf{P3STop}}(u)$ for each user, we aim to put as many unreliable negative items below positive items as possible, which results

in high accuracy especially for the top-ranked items. As for the optimization of Eqn. 4.6, since $\mathbb{I}[\cdot]$ is non-convex, we replace it with the hinge loss function $\ell(x) = max(0, 1 - x)$, which is a widely used convex surrogate for the indicator function [106].

Comparison with P3S_2. Note that P3STop (Eqn. 4.6) differs from P3S_2 (Eqn. 4.5) in that rather than independently comparing all the positive–negative instance pairs, P3STop sets the bound to the least relevant positive item and pushes negative items below it. This approach enables the negative items to be less involved during the model training, which in turn makes the model more robust to unreliable negative items.

Optimization Objective. Given the loss function $\mathcal{L}_{\mathsf{P3STop}}(u)$ for each user $u \in \mathcal{U}$ as in Eqn. 4.6, the final objective function to minimize is formulated as follows:

$$\mathcal{J}_{\mathsf{P3STop}}(\Theta) = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathcal{L}_{\mathsf{P3STop}}(u) + \frac{\lambda_\alpha}{2} \sum_{u \in \mathcal{U}} \|\alpha_u\|_2^2 + \frac{\lambda_\beta}{2} \sum_{i \in \mathcal{I}} \|\beta_i\|_2^2 \tag{4.7}$$

where λ_α and λ_β are regularization parameters for the user and for the item latent factors, respectively. We set $\lambda_\alpha = \lambda_\beta = \lambda$ to reduce the model complexity. We adopt the widely used stochastic gradient descent (SGD) method to optimize the objective function in Eqn. 4.7. For each user u, we first sample a triple (i, j, k) from the training set $\mathcal{O} = \{\mathcal{O}_u | u \in |\mathcal{U}|\}$ where $\mathcal{O}_u = \{(i, j, k) | i \in \mathbf{p}_u \wedge j \in \mathbf{c}_u \backslash \mathbf{p}_u \wedge k \in \mathcal{I} \backslash \mathbf{c}_u\}$, $\mathcal{O}_u$ denoting the training set for user u. We compute the gradient for each parameter in Θ, i.e., $\alpha_u, \beta_i, \beta_j, \beta_k$, and update each of them by using SGD. As our focus is to verify the benefit of our model assumptions, we do not adopt advanced negative sampling techniques [89, 52, 90, 107]. The gradient for each parameter is computed as follows:

- The gradient of α_u for $u \in \mathcal{U}$:

$$\begin{aligned}
\frac{\partial L_{\mathsf{P3STop}}(u)}{\partial \alpha_u} &= \mathbb{I}[\min_{i \in \mathbf{p}_u} \hat{x}_{ui} - \hat{x}_{uj} \leq 1] \left(\beta_j - \frac{\partial \min_{i \in p_u} \hat{x}_{ui}}{\partial \alpha_u} \right) \\
&+ \mathbb{I}[\min_{j \in \mathbf{c}_u \backslash \mathbf{p}_u} \hat{x}_{uj} - \hat{x}_{uk} \leq 1] \left(\beta_k - \frac{\partial \min_{j \in \mathbf{c}_u \backslash \mathbf{p}_u} \hat{x}_{uj}}{\partial \alpha_u} \right) \\
&+ \mathbb{I}[\min_{i \in \mathbf{p}_u} \hat{x}_{ui} - \hat{x}_{uk} \leq 1] \left(\beta_k - \frac{\partial \min_{i \in \mathbf{p}_u} \hat{x}_{ui}}{\partial \alpha_u} \right)
\end{aligned} \tag{4.8}$$

- The gradient of β_i for $i \in \mathbf{p}_u$

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathsf{P3STop}}(u)}{\partial \beta_i} &= \mathbb{I}\left[\min_{i\in\mathbf{p}_u} \hat{x}_{ui} - \hat{x}_{uj} \leq 1\right]\left(-\frac{\partial \min\limits_{i\in\mathbf{p}_u} \hat{x}_{ui}}{\partial \beta_i}\right) \\ &+ \mathbb{I}\left[\min_{i\in\mathbf{p}_u} \hat{x}_{ui} - \hat{x}_{uk} \leq 1\right]\left(-\frac{\partial \min\limits_{i\in p_u} \hat{x}_{ui}}{\partial \beta_i}\right)\end{aligned} \tag{4.9}$$

- The gradient of β_j for $j \in \mathbf{c}_u \backslash \mathbf{p}_u$

$$\begin{aligned}\frac{\partial \mathcal{L}_{\mathsf{P3STop}}(u)}{\partial \beta_j} &= \mathbb{I}[\min_{i\in\mathbf{p}_u} \hat{x}_{ui} - \hat{x}_{uj} \leq 1](\alpha_u) \\ &+ \mathbb{I}\left[\min_{j\in\mathbf{c}_u\backslash\mathbf{p}_u} \hat{x}_{uj} - \hat{x}_{uk} \leq 1\right]\left(-\frac{\partial \min\limits_{j\in\mathbf{c}_u\backslash\mathbf{p}_u} \hat{x}_{uj}}{\partial \beta_j}\right)\end{aligned} \tag{4.10}$$

- The gradient of β_k for $k \in \mathcal{I} \backslash \mathbf{c}_u$

$$\frac{\partial \mathcal{L}_{\mathsf{P3STop}}(u)}{\partial \beta_k} = \mathbb{I}[\min_{j\in\mathbf{c}_u\backslash\mathbf{p}_u} \hat{x}_{uj} - \hat{x}_{uk} \leq 1](\alpha_u) + \mathbb{I}[\min_{i\in\mathbf{p}_u} \hat{x}_{ui} - \hat{x}_{uk} \leq 1](\alpha_u) \tag{4.11}$$

4.5.1 Alternative Method: P3STop^{alt}

As an alternative to our proposed method P3STop, we can consider another method that relies more heavily on click records. For each user u, we compute the sum of the number of 1) P items ranked below the most relevant CBNP item, 2) CBNP items ranked below the most relevant NC item, and 3) P items ranked below the most relevant NC item, and minimize the following:

$$\begin{aligned}\mathcal{L}_{\mathsf{P3STop}^{alt}}(u) &= \frac{1}{|\mathbf{c}_u\backslash\mathbf{p}_u|}\sum_{j\in\mathbf{c}_u\backslash\mathbf{p}_u} \mathbb{I}\left[\hat{x}_{uj} \leq \max_{k\in I\backslash\mathbf{c}_u} \hat{x}_{uk}\right] \\ &+ \frac{1}{|\mathbf{p}_u|}\left[\sum_{i\in\mathbf{p}_u} \mathbb{I}\left[\hat{x}_{ui} \leq \max_{j\in\mathbf{c}_u\backslash\mathbf{p}_u} \hat{x}_{uj}\right] + \sum_{i\in\mathbf{p}_u} \mathbb{I}\left[\hat{x}_{ui} \leq \max_{k\in I\backslash\mathbf{c}_u} \hat{x}_{uk}\right]\right]\end{aligned} \tag{4.12}$$

This method, named P3STop^{alt}, is distinguished from P3STop in that P3STop^{alt} resorts to the negative items to set the bound. To be precise, it sets the *lower bound of the positive items to the score of the top-ranked negative item*; in contrast to P3STop that sets the *upper bound of negative items to the last-ranked positive item*. Here, we want the lower bound to be high enough so that pulling up the positive items above it is meaningful (Figure 4.1d). However, the negative items always include relatively unreliable click records in this case, and thus the lower bound set by the negative items is not guaranteed to be sufficiently high and robust; in contrast to high and robust bound of P3STop set by positive items.

This implies that pulling up positive items above relatively low bound would not yield high accuracy at the top. We later present the performance of P3STop^{alt} in the experiments (Table 4.3) to show the unreliableness of click records compared with purchase records. We note that P3STop^{alt} is an enhanced version of Inf-Push [93], whose underlying *AMAN* assumption is replaced with **Assumption 2** [108].

Complexity Analysis. Another benefit of P3STop is the improved time complexity compared with previous pairwise methods developed for bipartite ranking, such as BPR and P3S. More precisely, the time complexity of evaluating $\mathcal{J}_{\text{P3S\_2}}$ is $O(|\mathbf{p}_u||\mathbf{c}_u \backslash \mathbf{p}_u| + |\mathbf{c}_u \backslash \mathbf{p}_u||\mathcal{I} \backslash \mathbf{c}_u| + |\mathbf{p}_u||\mathcal{I} \backslash \mathbf{c}_u|) = O(m^2)$ whereas that for $\mathcal{J}_{\text{P3STop}}$ is $O(2 \times (|\mathbf{p}_u| + |\mathbf{c}_u \backslash \mathbf{p}_u| + |\mathcal{I} \backslash \mathbf{c}_u|)) = O(m)$, which is clear when converted into a dual form [108].

4.6 Experiments

In this section, we will answer the following RQs:

RQ1. Are click records useful for purchase prediction?

RQ2. Does P3STop focus on the accuracy near the top?

RQ3. Is P3STop robust to unreliable click records?

4.6.1 Experimental Settings

Dataset. We evaluated our proposed method on two real-world datasets each of which contains both purchase records and click records for the same set of users. The RecSys2015 dataset[3] consists of sessions of click and purchase sequences extracted from an e-commerce website, where we regard each session is as a user. We additionally ran experiments on a proprietary dataset from NAVER shopping, which is a web portal that provides a platform for online shopping. We collected users' click and purchase records for three months (Jan. 2017 through Mar. 2017). For both datasets, we removed users having fewer than five purchases and twenty clicks. After preprocessing, the RecSys2015 dataset contained 30,867 purchase records on 5,869 items and 102,939 click records on 11,071 items from 7,076 users, and the NAVER shopping dataset contained 23,373 purchase records on 6,743 items and 243,908 click records on 10,738 items from 5,317 users.

Methods Compared.

[3] http://2015.recsyschallenge.com/challenge.html

- **BPR** [26]: A pairwise learning-to-rank method based on the *AMAN* assumption as in Eqn. 4.1.
- **CLiMF** [109]: A collaborative ranking method for implicit feedback that directly maximizes MRR.
- **PMF** [110]: As PMF is a common baseline method for rating prediction, we modify it to model click and purchase records; we assign 1 to clicked items, and 2 to purchased items.
- **P3S_1, P3S_2, P3S_3, P3STop, P3STop^{alt}**: Our proposed methods based on Eqn. 4.2, 4.3, 4.4, 4.6, and 4.12, respectively.
- **Inf-Push** [93]: A collaborative ranking method based on explicit feedback that focuses on the ranking performance at the top.
- **eALS** [89]: The state-of-the-art sampling–based MF method that samples non-purchased items based on their popularity, which is shown to surpass the uniform weighting scheme [88].
- **GRU4REC** [104]: State-of-the-art session–based click prediction method based on GRU. As its goal (click prediction) differs from ours (purchase prediction), we added a fully connected layer at the end of last hidden state of GRU4REC for predicting the purchased items.

As BPR is one of our main competitors, we tried variants of BPR. First, as a naive approach to incorporating both purchase and click records into BPR given the $(u, i \in \mathbf{p}_u, j \in \mathcal{I} \backslash \mathbf{p}_u)$ triple, we modified the value of $(\hat{x}_{ui} - \hat{x}_{uj})$ to $0.5 * (\hat{x}_{ui} - \hat{x}_{uj})$ for $j \in \mathbf{c}_u \backslash \mathbf{p}_u$ assuming that the difference should not be as large as when item j is not clicked at all. However, the performance improvement was not statistically significant, hence we excluded for brevity. Second, we tried the "click–based purchase prediction" for BPR (using only click records instead of purchase records to predict purchases), but its performance was very poor, and hence excluded in the paper. Moreover, we assume that explicit feedback such as ratings are not provided, and thus we compare our methods with implicit feedback–based methods.

Evaluation Setting. We adopt the *leave-one-out* evaluation, which has been widely used in literature [105, 24, 89, 26]. More precisely, for both datasets, we 1) chronologically ordered the sequence of purchase data and used the last records as test data and the remainder as training data, and 2) used the click records of the user up to the timestamp of the last purchase in the training data and discarded

the rest. Note that we aim to recommend to users, items that are unknown and new to the users, which makes our problem more challenging and practical [22].

Table 4.1: Comparisons of different assumptions for P3S.

Data	Metric	P3S_1	P3S_2	P3S_3
RecSys	R@10	0.2772±0.0009	**0.2955**±0.0021	0.0017±0.0004
	N@10	0.1773±0.0005	**0.1778**±0.0017	0.0014±0.0003
	M@10	**0.1463**±0.0006	0.1425±0.0012	0.0011±0.0003
	AUC	0.8716±0.0006	**0.9083**±0.0009	0.4602±0.0035
NAVER	R@10	0.0602±0.0016	**0.0618**±0.0022	0.0028±0.0010
	N@10	0.0298±0.0006	**0.0331**±0.0004	0.0013±0.0007
	M@10	0.0215±0.0006	**0.0233**±0.0003	0.0009±0.0006
	AUC	0.8765±0.0029	**0.9420**±0.0005	0.4948±0.0050

Evaluation Metrics. As our objective is to optimize the accuracy at the top, we measured the ranking performance using three metrics that emphasize the accuracy at the top (Recall@N, NDCG@N, MRR@N) when N is small, and one that does not (AUC). Precision is not employed because each user has only one test data in which case Precision is proportional to Recall. We do not consider metrics, such as RMSE and MAE, as they are suitable for explicit feedback datasets [1].

Table 4.2: Performance comparisons for different methods. (*Imp.*: improvement over the best competitor.)

Metric	eALS	BPR	CLiMF	PMF	GRU4REC	P3S_2	P3STop	*Imp.*
RecSys2015								
Recall@10	0.2132±0.0006	0.2801±0.0011	0.0015±0.0007	0.2280±0.0024	0.1916±0.0018	0.2955±0.0021	**0.3119**±0.0047	11.35%
Recall@20	0.2549±0.0008	0.3489±0.0012	0.0022±0.0008	0.2883±0.0034	0.2580±0.0015	0.3853±0.0011	**0.3959**±0.0068	13.47%
NDCG@10	0.1405±0.0004	0.1792±0.0010	0.0007±0.0003	0.1418±0.0019	0.0895±0.0010	0.1778±0.0017	**0.1913**±0.0050	6.75%
NDCG@20	0.1499±0.0006	0.1966±0.0009	0.0008±0.0003	0.1566±0.0017	0.1063±0.0007	0.2006±0.0011	**0.2125**±0.0055	8.09%
MRR@10	0.1178±0.0005	0.1483±0.0009	0.0005±0.0003	0.1154±0.0020	0.0585±0.0009	0.1425±0.0012	**0.1542**±0.0051	3.98%
MRR@20	0.1204±0.0004	0.1531±0.0009	0.0006±0.0003	0.1194±0.0019	0.0631±0.0008	0.1483±0.0014	**0.1600**±0.0052	4.51%
AUC	0.7914±0.0006	0.8714±0.0005	0.5523±0.0380	0.8688±0.0130	0.7957±0.0009	**0.9083**±0.0009	0.9039±0.0008	4.23%
NAVER								
Recall@10	0.0475±0.0003	0.0608±0.0010	0.0076±0.0038	0.0360±0.0017	0.0466±0.0022	0.0618±0.0022	**0.0690**±0.0015	13.49%
Recall@20	0.0739±0.0005	0.1031±0.0011	0.0115±0.0039	0.0646±0.0030	0.0750±0.0036	0.1074±0.0024	**0.1130**±0.0029	9.60%
NDCG@10	0.0222±0.0001	0.0311±0.0004	0.0040±0.0029	0.0180±0.0005	0.0220±0.0015	0.0331±0.0004	**0.0350**±0.0007	12.54%
NDCG@20	0.0288±0.0002	0.0417±0.0007	0.0049±0.0025	0.0244±0.0009	0.0291±0.0017	0.0443±0.0003	**0.0460**±0.0008	10.31%
MRR@10	0.0147±0.0001	0.0224±0.0007	0.0029±0.0029	0.0126±0.0005	0.0147±0.0013	0.0233±0.0003	**0.0244**±0.0010	8.93%
MRR@20	0.0165±0.0001	0.0252±0.0008	0.0032±0.0028	0.0143±0.0006	0.0166±0.0013	0.0263±0.0004	**0.0274**±0.0010	8.73%
AUC	0.7371±0.0156	0.8622±0.0018	0.7942±0.0156	0.8520±0.0003	0.7233±0.0011	**0.9420**±0.0005	0.9313±0.0014	9.26%

Parameters. As the last purchased item of each user is used as test data, conventional cross-validation is not applicable here. Instead, we made a validation dataset by using the last purchased item of training data, and performed grid search on the validation dataset to find the best hyperparameters. For all baselines, we tuned the hyperparameters by performing grid searches with $K \in \{10, 20, ..., 200\}$, and $\eta, \lambda \in \{0.01, 0.05, 0.1\}$. For experiments, we report the mean and the standard deviation over five runs with different random seeds for initialization.

4.6.2 Performance Analysis

RQ1) Usefulness of Click Records. We begin by showing which of our model assumptions (Eqn. 4.2,4.3,or 4.4) performed the best in Table 4.1. We observe that P3S_2, which is based on **Assumption 2** (Eqn. 4.3), generally outperforms P3S_1 and P3S_3 on both datasets, with P3S_3 performing extremely poorly. This implies that CBNP items of a user are definitely more helpful in eliciting the user's preference than NC items, and thus this relationship should be taken into account for purchase prediction.

Given that P3S_2 is the right choice among the P3Ss, we now compare its performance with state-of-the-art implicit feedback based–recommendation methods in Table 4.2. We observe that P3S_2 consistently outperformed the purchase record–based baselines, i.e., eALS, CLiMF and BPR, with very few exceptions. This indicates that when click records are properly combined with purchase records to define the order of user preferences among non-purchased items, we can complement the missing user–item interactions of purchase records, thereby obtaining better prediction results. Other observations from Table 4.2 are as follows: **1)** BPR consistently outperformed PMF. Although PMF utilizes both the click and purchase records for making recommendations, it performed worse than BPR, which leverages only the purchase records. This indicates that we should meticulously design a method when jointly modeling both the click and purchase records, which in fact was one of the objectives for this study. **2)** GRU4REC, which is a click-based purchase prediction method, generally performs worse than other methods based on either *only purchase* or *both click and purchase*. This shows that while click records can be directly used for click prediction as in [104], using only click records for purchase prediction limits the prediction performance because click records are relatively weaker signal than purchases, which corroborates the benefit of our model assumptions for purchase prediction. However, we note that GRU4REC performs better than click–based version of BPR (not shown in Table 4.2), confirming the advantage of considering the sequential infor-

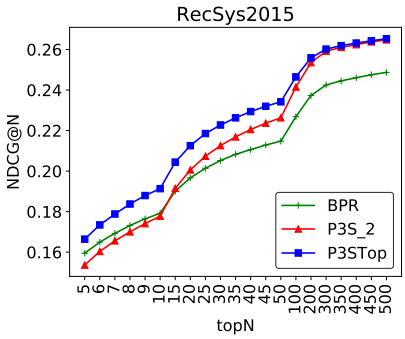


Figure 4.2: Comparisons over various Ns (RecSys2015).

mation of click records by using GRU. **3)** BPR consistently outperformed eALS. This demonstrates the superiority of the pairwise learning-to-rank method upon which our method is built. **4)** Although we expected CLiMF to perform better than BPR as in the original study [109], its performance turned out to be very poor. We attribute this poor performance to the difference in the target task, which leads to a different model formulation. More precisely, CLiMF ignores all the missing user–item interactions and only considers positive feedback, and we argue that this can be effective for tasks with sufficient positive feedback such as friend recommendations (about 70 friends per user on average for the datasets used by [109]). However, the poor performance of CLiMF in our task implies that this formulation is not appropriate for purchase prediction where the amount of positive feedback is usually small (four purchased items per user on average for the datasets used in this work). In addition, the poor performance of CLiMF compared with BPR, which leverages the negative feedback, corroborates the importance of leveraging negative feedback in different target tasks.

RQ2) Focus on the Accuracy of the Top-Ranked Items. We observe from Figure 4.2 that although P3S_2 outperforms BPR for relatively large Ns ($N \geq 20$), their performance is similar to BPR for smaller Ns less than 10; in this range, BPR even performs better than P3S_2. However, providing accurate recommendations in the lower part of the recommendation list as P3S_2 is not desired for e-commerce stores because *users are mainly interested in the top-ranked items* in practice [92]. Recall that the objective of our ultimate proposed method, P3STop, is to focus on the accuracy of the top-ranked items. The following observations are made regarding the performance of P3STop: **1)** From Table 4.2, in terms of Recall, NDCG and MRR, P3STop considerably outperforms all competitors including P3S_2 for $N = 10, 20$, which are relatively small Ns. **2)** More importantly, for even smaller Ns less than 10 (Figure 4.2), we observe that P3STop still outperforms both BPR and P3S_2, whereas for large

Table 4.3: Comparisons among "*push*" algorithms.

Metric	Inf-Push	P3STop	P3STop^{alt}	P3STop^{mix}
RecSys2015				
R@10	0.163$_{\pm 0.0018}$	**0.311**$_{\pm 0.0047}$	0.189$_{\pm 0.0034}$	0.184$_{\pm 0.0024}$
N@10	0.081$_{\pm 0.0011}$	**0.191**$_{\pm 0.0050}$	0.102$_{\pm 0.0032}$	0.098$_{\pm 0.0015}$
M@10	0.057$_{\pm 0.0011}$	**0.154**$_{\pm 0.0051}$	0.076$_{\pm 0.0032}$	0.073$_{\pm 0.0015}$
AUC	0.740$_{\pm 0.0019}$	**0.903**$_{\pm 0.0008}$	0.880$_{\pm 0.0016}$	0.864$_{\pm 0.0013}$
NAVER				
R@10	0.017$_{\pm 0.0015}$	**0.069**$_{\pm 0.0015}$	0.048$_{\pm 0.0026}$	0.044$_{\pm 0.0018}$
N@10	0.008$_{\pm 0.0008}$	**0.035**$_{\pm 0.0007}$	0.021$_{\pm 0.0013}$	0.022$_{\pm 0.0014}$
M@10	0.005$_{\pm 0.0007}$	**0.024**$_{\pm 0.0010}$	0.015$_{\pm 0.0011}$	0.016$_{\pm 0.0012}$
AUC	0.601$_{\pm 0.0026}$	**0.931**$_{\pm 0.0014}$	0.862$_{\pm 0.0028}$	0.828$_{\pm 0.0013}$

Ns ($N \geq 30$), the performance gap between P3S_2 and P3STop starts to get smaller, and the performance of P3STop almost equals to that of P3S_2 above $N = 300$. This implies that P3STop focuses on the accuracy of the top-ranked items ($N \leq 20$) at the expense of the accuracy in the lower part of the recommendation list ($N \geq 30$), which answers RQ2. We observed similar results for other metrics as well. **3)** Above results are corroborated by the performance in terms of AUC, a metric that treats a mistake in the higher part of the recommendation list as equal to one the lower part. More precisely, P3S_2 consistently outperforms P3STop in both datasets in terms of AUC, which implies that P3S_2 provides a more balanced recommendation list; this conversely shows that P3S_2 does not particularly focus on the top. We attribute this performance to the fact that P3S_2 is built upon the BPR model, whose objective is to optimize for the AUC metric (Eqn. 4.5).

RQ3) Robustness to Unreliable Click Records. Table 4.3 shows the comparisons among "*push*" algorithms that focus on the accuracy of the top-ranked items. Our proposed method P3STop (Figure 4.1c), which places more emphasis on positive items than on negative items, considerably outperforms P3STop^{alt} (Figure 4.1d), which places more emphasis on negative items than on positive items. This verifies that resorting to negative items deteriorates the prediction performance implying that click records are indeed relatively more unreliable than purchase records, which answers RQ3. Other observations from Table 4.3 are as follows; Note that Inf-Push [93] is a collaborative ranking method based on explicit feedback that pulls the incorrectly ranked relevant items above non-relevant items. Because this method was originally designed for explicit feedback, we cannot directly compare it with our proposed method. Instead, we treat purchased items as relevant and all the non-purchased items as non-relevant. **1)** Although P3STop^{alt} performs worse than P3STop, P3STop^{alt} slightly outperforms

Inf-Push, which verifies again that defining the order of user preferences among non-purchased items as specified in **Assumption 2** is indeed beneficial. **2)** The performance of Inf-Push is very poor compared with not only other "*push*" algorithms but also the competitors listed in Table 4.2. Recall that Inf-Push is distinguished from P3STop^{alt} in that the underlying assumption of P3STop^{alt}, i.e., **Assumption 2**, is replaced with the *AMAN* assumption. Hence, analogous to the case of P3STop^{alt}, we want the lower bound of purchased items set by the top-ranked non-purchased item to be high enough so that pulling up purchased items above it is meaningful. However, the poor performance of Inf-Push implies that non-purchased items should not be equally considered as negative, and that we need to define the order of user preferences among non-purchased items. **3)** P3STop^{mix} is a method that jointly minimizes the objective functions of P3STop and P3STop^{alt}. The performance of P3STop^{mix} is worse than both P3STop and P3STop^{alt}, which implies that jointly learning these two methods provides no benefit owing to the unreliableness of click records.

4.7 Related Work

Recommender Systems with Implicit Feedback. Although explicit feedback, such as rating, is a valuable source of information that reveals user preferences, it is difficult to obtain a large quantity of such data. Hence, the vast majority of work has focused on eliciting user preferences from implicit feedback such as bookmarks [7], item purchases [26, 105], and TV channel tuning history [52]. These methods adopted the MF technique to model the preference of users on items [1]. Specifically, Hu *et al.* proposed WMF that [88] introduced the concept of confidence to measure the influence of observed items and unobserved items on users' preferences. Later, various sampling strategies to generate negative examples from unobserved items were proposed [52, 89, 90]. Moreover, several pairwise learning-to-rank methods [26, 61, 107, 111] based on pairwise comparisons between observed items and unobserved items have been proposed. However, all the aforementioned methods are based on the *AMAN* assumption or predefined heuristic weights, which limits further performance improvement.

To cope with the aforementioned challenges, users' social network information has been leveraged. For example, a method introduced by Zhao *et al.* assigned higher ranks to the items that a user's friends prefer than to the items that neither he nor his friends prefer [7]. This work was extended by Wang *et al.* , who introduced a method that categorizes unobserved items into three groups regarding users' strong and weak ties with other users [2]. However, these methods are

only applicable when users' social network information is available, which is usually not the case for most e-commerce stores. Various other methods that incorporate side information for solving the data sparsity issue of implicit feedback have been proposed: review text [9], item image [105] and temporal information [22]. However, this line of research is not directly related to our proposed method in that ours does not consider any side information. Lastly, Parra *et al.* [112] proposed a parametric model to map implicit feedback to explicit feedback under the assumption that there is some correlation between implicit and explicit feedback. However, it requires a minimal amount of explicit feedback, whereas ours entirely resort to implicit feedback.

Modeling User Behavior. With the advent of e-commerce, much work has been devoted to understanding behavior of online users [100, 101], and specifically to predicting purchase behaviors [103, 102]. As the former line of work, Lo *et al.* [100] studied user activity and purchasing behaviors that vary over time, especially focusing on user purchasing intent. Most recently, Cheng *et al.* [101] extended Lo *et al.* 's work [100] by generalizing their analysis on characterizing the relationship between a user's intent and his behavior. Our goal is different in that we focus on predicting users' purchases, rather than predicting users' various intents from their online behaviors. Meanwhile, as the latter line of work, given user demographics and implicit feedback including click record and purchase record, Liu *et al.* [102] proposed an ensemble method to predict which customers would return to the same merchant within six months period. They formulated the problem as a classification task and trained various classification methods. While similarly using both purchase record and click record, our task is different in that we aim to predict items that users will purchase rather than to predict repeat buyers. Moreover, Li *et al.* [103] proposed a MF–based method that predicts the conversion response of users in display advertising, the goal of which inherently differs from our task.

Optimizing the Accuracy at the Top. Considering that users are mainly interested in the top-ranked items [92], optimizing for the accuracy near the top is of great importance in practice. Thanks to the success of the above approaches in general ranking tasks [113, 114, 106], they have been recently adopted in the field of recommender systems. Weimer *et al.* proposed CoFiRank [115], which directly optimizes Normalized Discounted Cumulative Gain (NDCG) by minimizing its convex upper bound. Later, Shi *et al.* proposed CLiMF [109], xCLiMF [116], and GAPfm [117], which optimize mean reciprocal rank, expected reciprocal rank and graded average precision, respectively. Among these methods, CLiMF is based on implicit feedback, whereas others are based on explicit feedback, and thus we

compared our proposed method with CLiMF in our experiments in Section 4.6. Furthermore, Christakopoulou and Banerjee proposed PushCR, which applies p-norm push, infinite push and reverse-height push [106] to a collaborative ranking task in which the ranking loss focuses on the accuracy of the top-ranked items for each user [93]. Hu and Li recently proposed DCR, which focuses on the accuracy at the top by modeling user ratings based on an ordinal classification framework [94]. Lastly, Forsati *et al.* [118] and Rafailidis and Crestani [95] incorporated user social network data as side information to enhance the accuracy at the top. However, these methods cannot be directly compared with ours because 1) PushCR and DCR consider explicit feedback, whereas ours is solely based on implicit feedback, and 2) the latter works incorporate side information related to users, whereas ours does not.

Position Bias of Click Models. Position bias is a fundamental problem pertaining to click records, where users tend to click on higher ranked items regardless of their relevance [119, 120]. To tackle the position bias issue for CTR prediction, previous click models assume that the click probability depends on the probability of examining a position, and the relevance of the document displayed at that position. However, we focus on purchase prediction rather than CTR prediction; we aim to overcome the data sparsity of purchase records by leveraging their relationships with click records. Hence, the position bias in terms of click models is out of scope for our current work.

4.8 Conclusion

In this paper, we introduced a framework that leverages users' past click records to complement the missing user–item interactions of purchase records. To this end, we formulated various model assumptions that define the order of user preferences regarding the non-purchased items, and demonstrated that click records are indeed useful for purchase prediction. We then proposed a novel learning-to-rank method, P3STop, that is customized to be robust to relatively unreliable click records by particularly focusing on the accuracy of the top-ranked items. We conducted extensive experiments on two real-world e-commerce datasets and verified the benefit of our proposed method compared with the state-of-the-art baselines.

For future work, we plan to extend our framework to model the temporal and sequential information of clicks and purchases by using deep learning–based approaches such as recurrent neural networks [98, 121] and convolutional neural networks [122].

V. Overcoming Class-imbalance Problem for Recommendation

5.1 Overview

The goal of *RecSys Challenge 2015* [123] is: (1) to predict which user will end up with a purchase and if so, (2) to predict items that he/she will buy given click/purchase data provided by *YOOCHOOSE*. It is hard to achieve the goal of this *Challenge* because (1) the data does not contain user demographics information and it contains a lot of missing values and (2) the volume of the dataset is massive with about 33 million clicks and 1 million purchase history and the class distribution (the ratio of non-purchased clicks to purchased clicks) is highly imbalanced. In order to efficiently solve these problems, we propose (1) Comprehensive Feature Engineering method (CFE) including imputation of missing values to make up for insufficiency of information and (2) Decision Boundary Focused Under-Sampling method (DBFUS) to cope with class imbalance problem and to reduce learning time and memory usage.

5.2 Introduction

RecSys Challenge 2015 is a competition for predicting user behavior given a large sequence of click events data provided by *YOOCHOOSE*. Data consists of (1) click log data, (2) purchase log data and (3) test data. The click log contains about 9 million sessions (users) each of which contains sessionID, browsed itemID, browsed timestamp and category of browsed item. Each session consists of one or more clicks. The purchase log data contains about 1.1 million buys each of which contains itemPrice, itemQuantity and timestamp. The test dataset of about 2 million sessions was released for leaderboard evaluation.

There are two tasks that we need to solve. (1) For each session, we predict whether the session will end up with buying and (2) if so, we predict which specific items will be bought in the session. Since the given data does not contain sufficient information and involves a lot of missing values, it is challenging to effectively predict user purchases. Furthermore, due to the massive size of highly imbalanced data, it is not easy to efficiently train a prediction model.

In this paper, we suggest two methods for solving the problems: (1) Com-

prehensive Feature Engineering with imputation of missing values (CFE) and (2) Decision Boundary Focused Under-Sampling (DBFUS). *Feature engineering* refers to the process of extracting informative features from the original data and imputing missing information to improve the prediction accuracy. *Decision Boundary Focused Under-Sampling (DBFUS)* performs under-sampling on the data of majority class while keeping all the data in the minority class in order to alleviate class imbalance problem [124]. While DBFUS performs under-sampling on the majority class, it considers the distance to the decision boundary such that more data is sampled near the decision boundary. Thereby, DBFUS helps to improve the prediction accuracy by alleviating class imbalance problem and also helps to reduce the training time and the memory usage.

After applying CFE and DBFUS to the data, we run 25 gradient boosting classifiers, each of which is trained from a set of data generated by DBFUS, and we calculated the mean of their prediction outputs to generate the final prediction result. Each gradient boosting classifier consists of about 5,000 predictors (or weak learners).

Although *RecSys Challenge 2015* has two different tasks, they can be solved by a simple binary classification problem on each click instance. Specifically, for a given click instance, we ignore the sessionID and predict whether a user will buy the currently clicked item or not. Once we get the prediction results for each clicked instance, we can tell that a session with any positively predicted click will end with purchase. Note that for each purchased click in purchase log data, we marked the corresponding click in the click log data with the same sessionID and itemID as positive class.

5.3 Preliminaries

5.3.1 Class imbalance problem

Class imbalance problem occurs when the class of interest (minority class) rarely happens and thus the size of the class is outnumbered by the size of the majority class (i.e., binary class is assumed) [125]. In such cases, the conventional classifiers are biased towards the majority class, and eventually result in poor accuracy. Since low accuracy implies that the classifier cannot correctly predict the class of interest, the whole prediction results become meaningless. This problem occurs in various real-world applications including fraud detection, anomaly detection, and medical diagnosis. We observed that *RecSys Challenge 2015* dataset also suffers from class imbalance problem since the number of clicks associated with actual buying is naturally far smaller than those with non-buying.

Many researches have been conducted to deal with class imbalance problem, and most of them are categorized into three folds: 1) algorithmic-level approach [125], which tries to make existing classifier to be biased towards the class of interest, 2) data-level approach [126, 124, 127, 128], which alleviates the size difference between the classes by over-sampling or under-sampling, and 3) cost-sensitive learning approach [129], which gives more weights to the class of interest to impose high cost on misclassification on the class of interest.

In this challenge, we combined data-level approach, specifically under-sampling, with cost-sensitive learning to deal with class imbalance problem. Specifically, we adjust class distribution of the training data to have the equal amount of instances so that the prediction model is less biased towards the majority class. We choose under-sampling rather than over-sampling because under-sampling also helps to reduce the size of training dataset, and thus eventually reduces the training time and memory usage.

Based on under-sampling, we propose Decision Boundary Focused Under-Sampling (DBFUS) inspired by [124] that under-samples majority class from the decision boundary to retain the classification accuracy.

5.3.2 Gradient Boosting Classifier

We use 25 gradient boosting classifiers for our tasks. Specifically, the prediction model is a weighted combination of weak classifiers, which is built successively based on the residual error of preceding weak classifiers [130]. Assume that we have input variable $X = \{x_1, x_2, ..., x_n\}$ and output variable y. Given training data $\{x_i, y_i\}_1^n$, the goal is to find F^* that minimizes the expectation of the given loss function L, which is the exponential loss function in our case:

$$F^* = \arg\min_F E_{x,y}[L(y, F(x))] \tag{5.1}$$

Gradient boosting assumes F in the form of weighted sum of weak learners $h_i(x)$

$$F(x) = \sum_{i=1}^{n} \gamma_i h_i(x) + const. \tag{5.2}$$

, where γ_i is the weight (importance) of $h_i(x)$. It starts with a model consisting of a constant function $F_0(x)$ and expands it in forward-stage manner.

$$F_0(x) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma) \tag{5.3}$$

$$F_m(x) = F_{m-1}(x) + \arg\min_f \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + f(x_i)) \tag{5.4}$$

Then, the standard steepest gradient descent method is applied to solve the minimization problem. Note that gradient boosting method automatically detects feature interactions so that we can calculate the importance of each feature relative to other features. This property is used later in our method.

5.4 Proposed Methods

5.4.1 System Overview

The proposed methods consist of three steps: 1) data preprocessing (feature engineering) step, 2) sampling step, and 3) prediction model learning step. Figure 5.1 illustrates the overall framework of our model. In the first step, we conduct data preprocessing and feature engineering, which are explained in Section 3.2. Next, using the preprocessed data, we perform DBFUS as described in Section 3.3. Finally, we train our prediction model based on ensemble method that is described in Section 3.4.

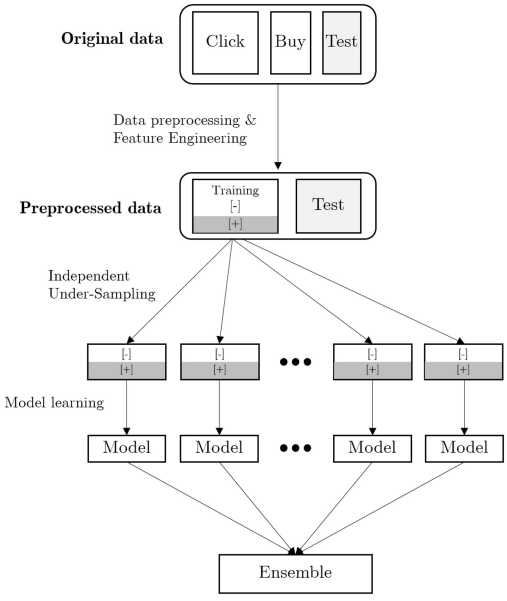


Figure 5.1: The architecture of proposed model. [+] implies positive class and [-] implies negative class.

5.4.2 Comprehensive Feature Engineering (CFE)

Since the amount of information given by the raw data is very limited and a lot of information is missing, it is important to extract useful information from it. Our feature engineering process (1) fills in missing values, (2) generates derived features based on comprehensive data analysis, and (3) selects important features based on their scores computed by gradient boosting classifier.

Imputation of missing values

The given dataset contains some missing information about item category, price and quantity of items purchased, especially in April and May. Therefore, we fill in missing values and in fact the prediction accuracy improved by this imputation. The following describes how we imputed missing values.

- **Category**: Instances with category 0, which represents missing category are imputed with valid categories from 1 to 12 if possible, and if not, remain as 0. Missing categories can be induced by looking at the complete log data. For example, an item with missing its category information in a month has its category information in other months. All the other categories greater than 12 that represent brands are converted to valid categories if possible, and if not, converted into category 13.

- **Price and Quantity**: If the price and quantity information is missing on a specific item, we filled it with the mean value of the prices and quantities of the item in the month that the item belongs to. If we do not find the price and quantity in the entire month, we filled it with the mean value of the item in the entire data. If there is no price and quantity information on the item at all, we filled it with the mean value of all items.

Engineered Features

The following is the list of derived features.

- **Day**: 31 days of a month are divided into 4 bins according to the number of clicks forming a binary vector of length 4
- **Weekday**: 7 weekdays of a week form binary vector of length 7
- **Hour**: 24 hours of a day are divided into 5 bins according to the number of clicks forming binary vector of length 5
- **Category**: a binary vector of length 14 is formed after the imputation step.

- **Price(P)**, **Quantity(Q)**: price and quantity of items purchased
- **Category S** (T/F): whether an item is in sale or not
- **Last Session** (T/F): whether an instance is the last click in the session or not
- **One category in a session**(T/F): whether the user browsed only one category in a session or not
- **Weekend** (T/F): whether it is weekend or not
- **Category ratio vector**: a vector of category ratio of an instance. For example, if there are 3 clicks occurred in a session and each one of them clicked on a different category, then the vector should be (0.33, 0.33, 0.33)
- **SNC**: number of clicks in a session
- **INW**: number of clicks of an item among the whole training data
- **INC**: number of clicks of an item in a session
- **IBW**: number of purchases of an item among the complete training data
- **DUR**: duration of a session in seconds
- **S1**: INC / SNC. Higher value implies higher probability of ending with purchase
- **S2**: IBW / INW. Higher value implies higher probability of ending with purchase
- **IMC**: number of clicks of an item in a month
- **IMB**: number of purchases of an item in a month
- **IR1**: ratio of an item in a session
- **IR2**: ratio of an item clicks in a session
- **CR1**: ratio of a category in a session
- **CR2**: ratio of a category clicks in a session

To verify the quality of engineered features, we performed principal component analysis (PCA) [131] on the data represented by the features, and took the first two principal components to visualize them. Figure 5.2 shows positive and negative instances projected on the first two principal components. *The instances are surprisingly well divided according to the two principal components*, which implies our feature engineering process made success in extracting valuable features that represent the data presumably well.

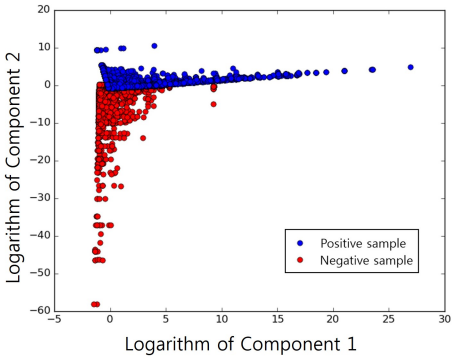


Figure 5.2: Result of PCA

Feature Selection

As explained in Section 2.3, gradient boosting classifier has an ability to provide feature importance information, thus we use this information to select important features for our task. Since categorical features give useful information as a whole, we included all the categorical features and only calculated feature importance scores for numerical features. Table 5.1 shows the mean values of feature importance scores for numerical features. After cross-validation process, we observed that when excluding two numerical features of the lowest scores, i.e., IBW and CR1, we get the best performance.

Feature	Importance	Feature	Importance	Feature	Importance
P	0.036	IBW	**0.005**	IMB	0.037
Q	0.042	DUR	0.146	IR1	0.034
SNC	0.027	S1	0.025	IR2	0.027
INW	0.034	S2	0.062	CR1	**0.007**
INC	0.05	IMC	0.04	CR2	0.031

Table 5.1: Feature importances of numerical features

5.4.3 Decision Boundary Focused Under-Sampling (DBFUS)

After the feature engineering stage, we carry out Decision Boundary Focused Under-Sampling (DBFUS) in order to alleviate class imbalance problem and reduce the training time and memory usage [124]. In the given dataset, the number of negative instances (clicks that did not end with buying) is about 25 times larger than that of positive instances (clicks that ended with buying). Thus, we kept all the positive instances and performed under-sampling only on the negative class of data.

Since we discovered in Section 3.2.2 that there exists a decision boundary

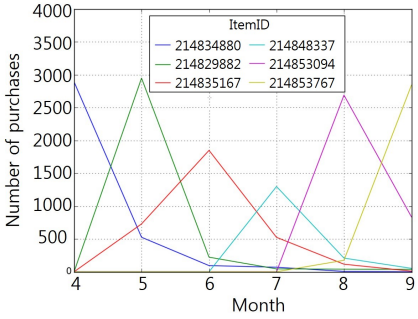


Figure 5.3: Examples of items displayed monthly.

that separates the training data well, we first calculated (by cross-validation) the decision boundary $\theta \in [0, 1]$ and used it as a criterion for sampling. Specifically, a half of the data is sampled near the boundary, i.e., $\theta < instances \leq \theta + \epsilon$, and the other half is sampled from the other area, i.e., $instances > \theta + \epsilon$. (For our case, $\theta = 0.32$, $\epsilon = 0.1$). From our experiments, we find that the prediction accuracy is the best when the ratio of the negative instances to positive instances is 3:1. Therefore, we sampled 1.5 times as much instances as positive training data from each side of the negative class. That is, 1.5 times from $\theta < instances \leq \theta + \epsilon$ and the remaining 1.5 times from $instances > \theta + \epsilon$.

Note that by reducing the class imbalance ratio from about 25:1 to 3:1, we reduce the training time, but also lose potentially useful information. To resolve this information loss problem, we independently performed DBFUS 25 times and constructed 25 different models because the size of negative class is 25 times larger than that of positive class. Note that since each model can be trained in parallel because they are independent to each other, this process did not increase training time. Each model becomes a weak learner of our ensemble classifier (as shown in Figure 1). The mean of the output values from 25 models are calculated to get the final prediction result. Note that our method can be considered as *"ensemble of ensembles"*

5.4.4 Learning Strategy

Splitting Monthly

We observed from our analysis that the purchase patterns for each month are significantly different. Figure 5.3 shows monthly patterns of item purchases where it shows clear patterns of specific items especially purchased frequently in specific months. That is, there exist certain items that are popular in April but not in August. Thus, we split the data monthly and constructed our model

for each month. The test data is predicted by the corresponding model that the data belongs to the same month. Note that we could get improvements by more than 5,000 points just by splitting the data monthly. Although we tried this exact same process by splitting weekdaily (Monday, Tuesday, etc.), we could not get as much improvement as we got from splitting monthly. Moreover, we tried to combine the results of monthly splitting and weekdaily splitting but without success.

Cost-sensitive learning

Gradient boosting algorithm is devised to reduce the bias towards the majority class by concentrating on misclassified training data. During the classification process, cost-sensitive learning scheme is adopted where larger misclassification cost is assigned to minority class. That is, it forces the classifier to concentrate more on the minority class. Note that we achieve this by adjusting weights inversely proportional to class frequencies in the training data. In particular, we use three times higher weight for positive class than that for negative class.

5.5 Experiment

The dataset given by *YOOCHOOSE* consists of click log data, purchase log data and test data each of which contains 33,003,944 entries, 1,150,753 entries and 8,251,791 entries, respectively. All the processes we described throughout this paper are implemented in *Python* using *Scikit-learn*, *Numpy*, *h5py*, and *Scipy*.

Although gradient boosting classifier is known to have a small number of hyperparameters to be tuned and thus easy to be used, we discovered that carefully tuning each parameter in fact improves the accuracy significantly. Therefore, we conducted stratified 5 fold cross-validation on our training data and found that the following parameters work best: n_estimators: 5000, max_leaf_nodes: 20, max_depth: N/A, min_samples_split: 1, learning_rate: 0.17, max_features: number of whole features. (All scikit-learn parameters)

After applying our method, we could get 54403.6 on the leaderboard. Although it takes about 7 hours to train our submitted model, it only takes a few minutes for predicting about 8 million clicks, which is reasonable because training process can be done off-line. Figure 5.4 shows changes on the training time as the number of weak learners increases, and our submitted model used 5,000 weak learners.

Lastly, to verify the benefit of our sampling method, DBFUS, we compared with the case without DBFUS. Figure 5.5 shows the result of the comparison.

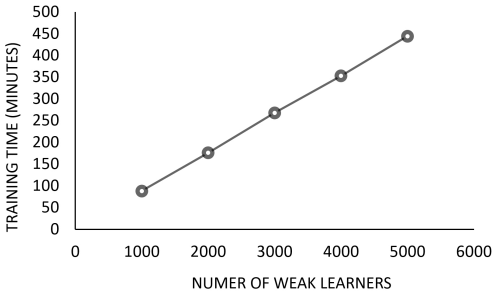


Figure 5.4: Training time w.r.t. number of weak learners of GBDT.

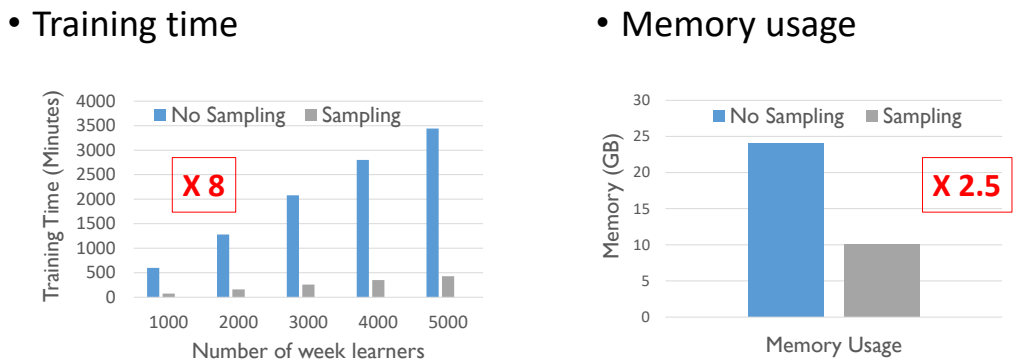


Figure 5.5: Benefit of DBFUS.

As shown in the figure, DBFUS runs 8 times faster and consumes 2.5 times less memory compared with the case without DBFUS; we achieve this even without compromising the accuracy of the method, which verifies the benefit of DBFUS.

5.6 Conclusion

In this paper, we described Decision Boundary Focused Under-Sampling (DBFUS) based on Comprehensive Feature Engineering (CFE) for predicting user purchase in E-commerce. By running PCA, we found out that our *feature engineering process* extracts features that are effective for our task. In addition, although DBFUS removes potentially informative data by under-sampling, we minimized the information loss by multiple ensembles, and thus the model with DBFUS performed even better than the model with the original data without under-sampling process. As for the learning algorithm, we tried various algorithms such as SVM, ANN, and linear methods with various loss functions, but the gradient boosting classifier with our feature engineering and DBFUS performed the best. Experimental results on *RecSys Challenge 2015* dataset demonstrated the effectiveness of our proposed method.

VI. Beyond Matrix Factorization: Metric Learning-based Recommendation using Translation Mechanism.

6.1 Overview

Recently, matrix factorization–based recommendation methods have been criticized for the problem raised by the triangle inequality violation. Although several metric learning–based approaches have been proposed to overcome this issue, existing approaches typically project each user to a single point in the metric space, and thus do not suffice for properly modeling the *intensity* and the *heterogeneity* of user–item relationships in implicit feedback. In this paper, we propose TransCF to discover such latent user–item relationships embodied in implicit user–item interactions. Inspired by the translation mechanism popularized by knowledge graph embedding, we construct user–item specific translation vectors by employing the neighborhood information of users and items, and translate each user toward items according to the user's relationships with the items. Our proposed method outperforms several state-of-the-art methods for top-N recommendation on seven real-world datasets by up to 17% in terms of hit ratio. We also conduct extensive qualitative evaluations on the translation vectors learned by our proposed method to ascertain the benefit of adopting the translation mechanism for implicit feedback–based recommendations.

6.2 Introduction

The recent explosive growth of information on the Internet inundates users with choices, and recommender systems play a crucial role not only in helping users in their decision making, but also in increasing the revenues of e-commerce companies. Among various recommendation techniques, matrix factorization (MF)–based collaborative filtering (CF) [73] has been shown to be the most successful; it assumes that users who have had similar interests in the past will tend to share similar interests in the future [132]. More specifically, MF learns low-rank vector representations of users and items from their previous interaction history and models the similarity of user–item pairs using inner products. However, a critical yet not widely recognized flaw of employing the inner product as

a similarity metric is that it violates the triangle inequality [133]. As a concrete example, if user u_1 liked items v_1 and v_2, MF will put both items close to u_1, but will not necessarily place v_1 and v_2 close to each other. That is to say, the seemingly obvious item–item similarity (between v_1 and v_2) is not guaranteed to be learned when the triangle inequality is violated [25].

To address the above limitation of MF–based recommendation, several metric learning approaches have been proposed [134, 135, 136, 25]. They commonly project entities (users and items) into a low-dimensional metric space, i.e., Euclidean space, where the similarity between points is inversely proportional to the Euclidean distance that satisfies the triangle inequality. Specifically, CML [25] is the state-of-the-art metric learning–based recommendation method for implicit feedback (e.g., clicks or purchases); it minimizes the distances between embeddings of a user and items that the user has interacted with, under the assumption that a user should be closer to the items the user likes than to those that the user does not. In this way, these approaches not only expect to propagate positive user–item relationships to other unknown user–item pairs, but also to capture the similarity within user–user and item–item pairs by satisfying the triangle inequality.

Although existing metric learning approaches have shown their effectiveness by satisfying the triangle inequality, they suffer from an inherent limitation, which is that *each user is projected to a single point in the metric space*. Such a rigid restriction imposed on user embeddings makes it hard to properly model the **intensity** and the **heterogeneity** of user–item relationships in implicit feedback. More precisely, a user's implicit feedback on multiple items does not necessarily indicate that he holds an equal preference for these items; rather, some of the items are more relevant to the user than others. This implies that the *intensity* of the user–item relationships embodied in implicit user–item interactions differ from one another. Moreover, a user may have a wide variety of tastes in different item categories, implying that the type of user–item relationship is *heterogeneous* with regard to the user's tastes in various item categories[1]. However, it is by no means an easy task to project each user to a single point such that his intense and heterogeneous relationships with items are fully considered, and this phenomenon compounds as the number of users and items increases.

This paper presents a novel method called TransCF to overcome the above limitation of existing metric learning approaches. We achieve this by adopting the *translation mechanism*, which has been proven to be effective for knowledge graph

[1]In Section 6.3, we show the existence of the intensity and the heterogeneity of user–item relationships in implicit feedback.

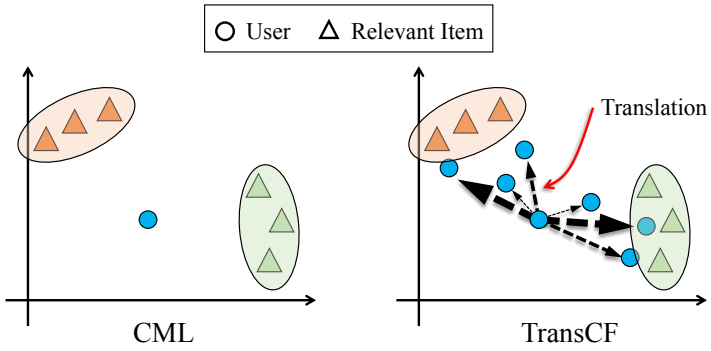


Figure 6.1: Toy example illustrating the benefit of the user–item specific translation vectors. Items are assumed to be grouped according to their categories.

embedding [137, 138], by which the relations between entities are interpreted as translation operations between them. In our work, we embed users and items as points in a low-dimensional metric space, and additionally introduce translation vectors to translate each user to multiple points. Equipped with the user–item specific translation vectors, a user is translated toward his relevant items by considering his relationships with the items, which facilitates the modeling of the intensity and the heterogeneity of user–item relationships in implicit feedback that were overlooked by the previous metric learning approaches. A further appeal of our proposed method is the ability to handle the *complex* nature of CF where it is common for a user to interact with multiple items, i.e., one-to-many mapping. Whereas it is not feasible for the previous metric learning approaches to pull a user closer to all of the items (one-to-many mapping) because a user is projected to a single point, our proposed method alleviates this issue because once a user is translated to multiple points, a one-to-many mapping can be deemed as multiple one-to-one mappings. As an illustration, consider the following toy example.

Toy Example. In Figure 6.1, whereas CML tries to find a single point for a user that minimizes the distances between the user and his relevant items, TransCF introduces user–item specific translation vectors to translate the user to multiple points according to the intensity and the heterogeneity of the user–item relationships. The more intense (thickness of the vectors) the user–item relationship, the closer the user is expected to be translated to the item. Besides, the direction of the vectors and the angles between them reflect the heterogeneity of the relationship with regard to the user's tastes in various item categories.

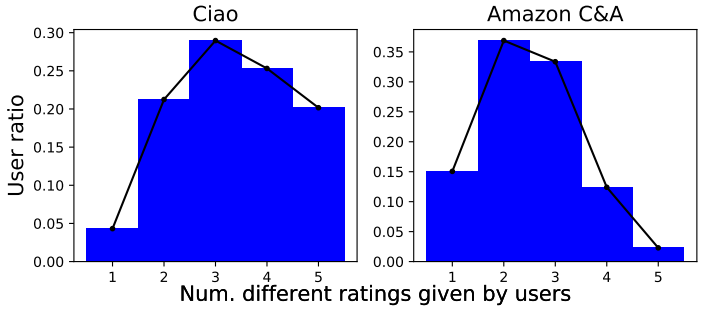
However, directly applying the translation mechanism into a general recommendation framework for implicit feedback is not viable because *user–item interactions in the implicit feedback dataset are not labeled*, unlike in knowledge

graphs where each relation between entities is given a label, such as "was_born_in" or "performed_in". Therefore, the key for successfully adopting the translation mechanism in our framework boils down to defining labels for implicit user–item interactions, and materializing them into the user–item specific translation vectors. Inspired by the highly effective neighborhood–based CF algorithms [73, 139, 28, 140, 141, 46], whose assumptions are that similar users display similar item preferences and that similar items are consumed by similar users, we employ the neighborhood information of users and items to construct the translation vectors (**Section 6.4.3**). Combined with a regularizer, specifically tailored to TransCF, that guides each user/item to its neighbors (**Section 6.4.4**), TransCF explicitly models the neighborhood information, in contrast to CML, which expects to achieve it implicitly by satisfying the triangle inequality. It is worth noting that the user–item specific translation vectors are constructed without introducing any new parameters, which prevents our proposed method from overfitting. Furthermore, we introduce a second regularizer to further improve the accuracy of TransCF in case the user–item relationships become more complex (**Section 6.4.4**). Our extensive experiments on seven real-world datasets (**Section 6.5.1**) show that TransCF considerably outperforms the state-of-the-art recommendation methods for implicit feedback by up to 17% in terms of hit ratio. We also perform various experiments to qualitatively ascertain that TransCF indeed benefit from modeling the intensity and the heterogeneity of user–item relationships in implicit feedback as illustrated in Figure 6.1 (**Section 6.5.2**). The source code is available at https://github.com/pcy1302/TransCF.

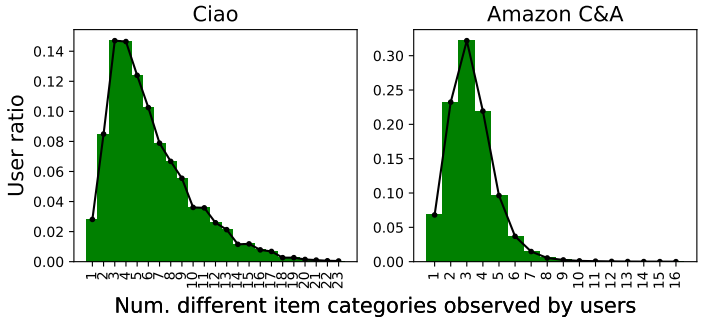
6.3 Data Analysis: Intensity and Heterogeneity

In this section, we perform data analyses on Ciao [142] and Amazon C&A [17] datasets to provide evidences of the existence of the *intensity* and the *heterogeneity* of user–item relationships in implicit feedback. The ratings in both datasets are integers from 1 to 5, and the numbers of unique item categories are 28 and 45, respectively. In our experiments, we regard every user–item interaction with a rating as an implicit feedback record[2]. Figure 6.2 shows the distribution of the number of different ratings and different item categories given and observed by users. In Figure 6.2a, we observe that most users (95% in Ciao and 85% in Amazon C&A) give two or more different ratings. This implies that implicit user–item interactions do indeed encode different **intensities** of user–item rela-

[2]It is unavoidable to use explicit feedback datasets as it is not feasible to judge the intensity of user–item relationships from implicit feedback.



(a) Distribution of the number of different ratings given by users.



(b) Dist. of the num. different item categories observed by users.

Figure 6.2: Data analysis on Ciao and Amazon C&A datasets.

tionships, assuming that the rating is a proxy for the intensity; a higher rating implies higher intensity. Moreover, in Figure 6.2b we observe that only a few users (3% in Ciao and 6% in Amazon C&A) interact with a single category, whereas the vast majority of users interact with two or more different item categories. This implies that users have diverse tastes in various item categories, and thus the type of user–item relationship is **heterogeneous** with regard to the users' tastes in various item categories.

In Section 6.5.2, we show by our experiments that TransCF can inversely infer knowledge regarding the intensity and the heterogeneity from the implicit user–item interactions.

6.4 Proposed Method: TransCF

In this section, we give details of our proposed method, which adopts the translation mechanism for modeling the intensity and the heterogeneity of user–item relationships in implicit feedback.

6.4.1 Problem Formulation

In this work, we focus on recommendations for implicit feedback. Let $\mathcal{U}$ and $\mathcal{I}$ denote a set of users and a set of items, respectively. $\mathcal{N}^{\mathcal{I}_u}$ denotes the set of items that user u has previously interacted with. Given implicit user–item interactions (e.g., bookmarks and purchase history), our goal is to recommend items $i \in \mathcal{I} \backslash \mathcal{N}^{\mathcal{I}_u}$ to each user $u \in \mathcal{U}$. Note that we use neither the rating information nor any auxiliary information regarding users and items (e.g., user social network or item category).

6.4.2 Scoring Function

Recall that the objectives of this work are 1) to address the limitation of the inner product as a scoring function, which violates the triangle inequality, and 2) to model the intensity and the heterogeneity of user–item relationships in implicit feedback. To this end, we adopt Euclidean distance as our distance metric to satisfy the triangle inequality, and we introduce translation vectors to model the intensity and the heterogeneity of implicit user–item interactions. Given K-dimensional embedding vectors $\boldsymbol{\alpha}_u \in \mathbb{R}^K$ for user u, $\boldsymbol{\beta}_i \in \mathbb{R}^K$ for item i, and $\boldsymbol{r}_{ui} \in \mathbb{R}^K$ for the translation of user u with regard to item i, the similarity score $s(u, i)$ between user u and item i is:

$$s(u, i) = - \left\| \boldsymbol{\alpha}_u + \boldsymbol{r}_{ui} - \boldsymbol{\beta}_i \right\|_2^2 \tag{6.1}$$

where a higher similarity score $s(u, i)$ implies a higher probability that user u will like item i. In other words, the similarity score between user u and item i is computed by the distance between the translated embedding vector of user u, given by $(\boldsymbol{\alpha}_u + \boldsymbol{r}_{ui})$, and the embedding vector $\boldsymbol{\beta}_i$ of item i.

Optimization Objective. Given the scoring function $s(u, i)$ as in Equation 6.1, we minimize a popular margin–based pairwise ranking criterion [25, 137, 138], i.e., hinge loss, as follows:

$$\mathcal{L}(\Theta) = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{N}^{\mathcal{I}_u}} \sum_{j \notin \mathcal{N}^{\mathcal{I}_u}} [\gamma - s(u, i) + s(u, j)]_+ \tag{6.2}$$

where $[x]_+ = max(x, 0)$, and γ is the margin. Our objective is to ensure that the similarity score of an observed (positive) user–item pair (u, i) is higher than that of an unobserved (negative) pair (u, j) by a margin of at least γ. By doing so, we aim to translate each user u closer to his relevant item i while translating him farther away from his non-relevant item j. One thing to note is that the

translation vector $\boldsymbol{r}_{ui}$ is user–item specific: *it translates user u with respect to item i according to the user's specific relationship with the item i.* This property enables TransCF not only to capture the intensity and the heterogeneity of user–item relationships in implicit feedback, but also to handle the complex nature of CF.

6.4.3 Modeling Relations: Neighborhood Information

As mentioned previously, the key for successfully adopting the translation mechanism in our framework is modeling the translation vectors so that they reflect the intensity and the heterogeneity of user–item relationships in implicit feedback. The translation vectors can be flexibly constructed as long as the user–item interactions are properly modeled. Although it is possible to introduce a new parameter for each translation vector corresponding to every user–item pair, we note that not only is this approach prone to overfitting owing to the large number of parameters, but also it does not allow the collaborative information to be explicitly modeled as no parameters are shared among users and among items. Therefore, in this work we employ the neighborhood information of users and items to construct the translation vectors without introducing any new parameters. Neighborhood information, which has been shown to be highly effective for recommendation [28, 140, 141, 46], implies that users are represented through the items that they prefer [73, 139], and items are represented through the users that prefer them. More precisely, considering that user embedding vectors reveal users' tastes, each item can be regarded as the average tastes of the users that have interacted with that item:

$$\boldsymbol{\beta}_i^{\text{nbr}} = \frac{1}{|\mathcal{N}_i^{\mathcal{U}}|} \sum_{k \in \mathcal{N}_i^{\mathcal{U}}} \boldsymbol{\alpha}_k \tag{6.3}$$

where $\boldsymbol{\beta}_i^{\text{nbr}} \in \mathbb{R}^K$ denotes the neighborhood embedding of item i, and $\mathcal{N}^{\mathcal{U}_i}$ denotes the set of users that have previously interacted with item i. Likewise, considering that item embedding vectors capture the prominent features of items, e.g., the item category, each user's taste can be represented by the average features of the items that the user has interacted with:

$$\boldsymbol{\alpha}_u^{\text{nbr}} = \frac{1}{|\mathcal{N}_u^{\mathcal{I}}|} \sum_{k \in \mathcal{N}_u^{\mathcal{I}}} \boldsymbol{\beta}_k \tag{6.4}$$

where $\boldsymbol{\alpha}_u^{\text{nbr}} \in \mathbb{R}^K$ denotes the neighborhood embedding of user u, and $\mathcal{N}^{\mathcal{I}_u}$ denotes the set of items that user u has previously interacted with. Given the

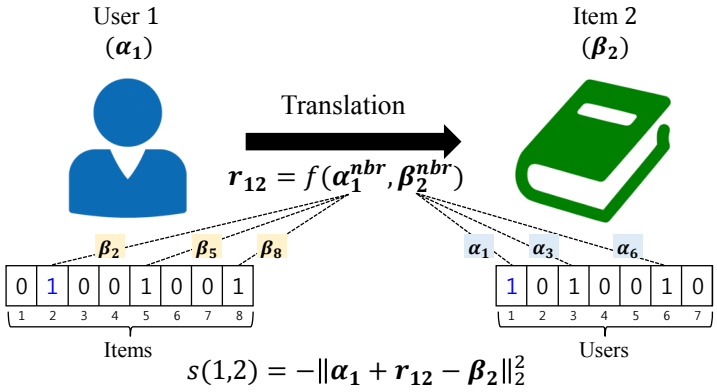


Figure 6.3: An overview of TransCF.

respective representations of an item and a user from the neighborhood perspective as in Equations 6.3 and 6.4, we use them to model the user–item interactions as follows:

$$\boldsymbol{r}_{ui} = f(\boldsymbol{\alpha}_u^{\text{nbr}}, \boldsymbol{\beta}_i^{\text{nbr}})$$

where $f(x, y)$ denotes a function to model the interaction between the two input vectors x and y. Note that although we could employ various functions such as a multi-layer perceptron, it turns out that a simple element-wise vector product provides the highest accuracy.

Figure 6.3 illustrates a working example of TransCF. Given that user 1 has interacted with items $\{2, 5, 8\}$ and item 2 has been interacted with by users $\{1, 3, 6\}$, we obtain the neighborhood embedding vector $\boldsymbol{\alpha}_1^{\text{nbr}}$ of user 1 from the set of items $\{2, 5, 8\}$, and the neighborhood embedding vector $\boldsymbol{\beta}_2^{\text{nbr}}$ of item 2 from the set of users $\{1, 3, 6\}$. Then, using these neighborhood embedding vectors $\boldsymbol{\alpha}_1^{\text{nbr}}$ and $\boldsymbol{\beta}_2^{\text{nbr}}$, we construct the translation embedding vector $\boldsymbol{r}_{12} = f(\boldsymbol{\alpha}_1^{\text{nbr}}, \boldsymbol{\beta}_2^{\text{nbr}})$, which is eventually used for computing the similarity score $s(1, 2) = -\|\boldsymbol{\alpha}_1 + \boldsymbol{r}_{12} - \boldsymbol{\beta}_2\|_2^2$.

Discussion

If users' social network information or auxiliary information related to items is given, it would be more intuitive to represent a user by his friends' tastes, and an item by its semantically related items, e.g., items that belong to the same category. However, such side information on users and items is not always provided in practice, and thus we do not consider any side information in our current work. Additionally, instead of simply averaging the embeddings of neighbors as in Equations 6.3 and 6.4, we could expect further improvements by applying the attention mechanism [143], whereby we give higher weights to more influential neighbors. Lastly, we could try swapping the role of users and items such that we

conversely translate items towards users. However, since the above extensions are straightforward and our main focus is to incorporate the translation mechanism into recommender systems, we leave these for future studies.

6.4.4 Model Regularization

In this section, we introduce two regularizers that are tailored to TransCF. We later show in our experiments (Section 6.5.1) that these regularizers are indeed beneficial.

Regularizer 1 – Neighborhood Regularizer

In Section 6.4.3, we explained how we reflect the neighborhood information of users and items into our translation vectors, under the assumption that users and items can be represented by their neighbors. In the case of users, we implicitly assumed that $\boldsymbol{\alpha}_u$ can be represented by $\boldsymbol{\alpha}_u^{\text{nbr}}$ (Equation 6.4), and likewise for items, that $\boldsymbol{\beta}_i$ can be represented by $\boldsymbol{\beta}_i^{nbr}$ (Equation 6.3). However, to ensure that the neighborhood information is explicitly incorporated into our model, we need to guide $\boldsymbol{\alpha}_u$ to be close to $\boldsymbol{\alpha}_u^{\text{nbr}}$, and $\boldsymbol{\beta}_i$ to be close to $\boldsymbol{\beta}_i^{\text{nbr}}$. To this end, we introduce a regularizer named $reg_{\text{nbr}}(\Theta)$ that minimizes the distance between users/items and their neighbors:

$$reg_{\text{nbr}}(\Theta) = \sum_{u \in \mathcal{U}} \left(\boldsymbol{\alpha}_u - \frac{1}{|\mathcal{N}_u^{\mathcal{I}}|} \sum_{k \in \mathcal{N}_u^{\mathcal{I}}} \boldsymbol{\beta}_k \right)^2 + \sum_{i \in \mathcal{I}} \left(\boldsymbol{\beta}_i - \frac{1}{|\mathcal{N}_i^{\mathcal{U}}|} \sum_{k \in \mathcal{N}_i^{\mathcal{U}}} \boldsymbol{\alpha}_k \right)^2 \tag{6.5}$$

Through Equation 6.5, we aim to explicitly inject the neighborhood information into the users and items, leading to more robust representations of users/items and their neighbors.

Regularizer 2 – Distance Regularizer

The objective function shown in Equation 6.2 trains the user and item embeddings so that given a positive user–item interaction (u, i), the item embedding $\boldsymbol{\beta}_i$ is the nearest neighbor of the user embedding $\boldsymbol{\alpha}_u$ translated by the translation vector $\boldsymbol{r}_{ui}$; i.e., $\boldsymbol{\alpha}_u + \boldsymbol{r}_{ui} \approx \boldsymbol{\beta}_i$. In other words, the objective function (Equation 6.2) expects that the positive item i will be pulled toward user u by pushing the negative item j away from user u, which is in fact what is done in CML. However, the relations become more complex as the number of user–item interactions grows, and it is crucial to guarantee that the actual distance between them is small. Therefore, we introduce a second regularizer named $reg_{\text{dist}}(\Theta)$

to explicitly pull the item embedding closer to the translated user embedding as follows:

$$reg_{\text{dist}}(\Theta) = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{N}^{\mathcal{I}_u}} -s(u,i) = \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{N}^{\mathcal{I}_u}} \|\boldsymbol{\alpha}_u + \boldsymbol{r}_{ui} - \boldsymbol{\beta}_i\|_2^2 \tag{6.6}$$

Notably, $reg_{\text{dist}}(\Theta)$ is equivalent to the loss $\mathcal{L}_{\text{pull}}$, which is introduced in the paper that proposed CML [25]. However, CML does not employ $\mathcal{L}_{\text{pull}}$ because *"an item can be liked by many users and it is not feasible to pull it closer to all of them"* as the authors mentioned in Section 3.1 of their paper [25]. This infeasibility is essentially caused by the fact that each user is projected to a single point, and we argue that translating a user to multiple points by introducing the user–item specific translation vectors as in our method makes it feasible to pull the item closer to all of the translated users, allowing TransCF to employ $reg_{\text{dist}}(\Theta)$.

Final Objective Function. Given the margin–based pairwise ranking function (Equation 6.2) and the two regularizers (Equations 6.5 and 6.6), our final objective function $\mathcal{J}(\Theta)$ to minimize is as follows:

$$\mathcal{J}(\Theta) = (\mathcal{L}(\Theta) + \lambda_{\text{nbr}} \cdot reg_{\text{nbr}}(\Theta) + \lambda_{\text{dist}} \cdot reg_{\text{dist}}(\Theta))$$

where λ_{nbr} and λ_{dist} are regularization coefficients for the neighborhood regularizer and the distance regularizer, respectively. We compute the gradient for parameters in $\Theta = \{\boldsymbol{\alpha}_u, \boldsymbol{\beta}_i\}$, and update them by using mini-batch stochastic gradient descent (SGD) with learning rate η as follows: $\Theta \leftarrow \Theta - \eta \times \frac{\partial \mathcal{J}(\Theta)}{\partial \Theta}$. As our focus is to verify the benefit of translation mechanism for recommendation, we do not adopt advanced negative sampling techniques [89, 52, 90, 107] for the model training. Instead, for each user $u \in \mathcal{U}$, we randomly generate 100 samples of $\{(i,j)|i \in \mathcal{N}^{\mathcal{I}_u} \cap j \notin \mathcal{N}^{\mathcal{I}_u}\}$ in every epoch. Moreover, as done in CML, we apply regularizations on $\boldsymbol{\alpha}_*$ and $\boldsymbol{\beta}_*$ after each epoch by bounding them within a unit sphere to mitigate 'curse of dimensionality' issue [137, 138]: $\|\boldsymbol{\alpha}_*\|^2 \leq 1$ and $\|\boldsymbol{\beta}_*\|^2 \leq 1$, which is achieved by $\boldsymbol{\alpha}_* \leftarrow \boldsymbol{\alpha}_* / \max(1, \|\boldsymbol{\alpha}_*\|^2)$ and $\boldsymbol{\beta}_* \leftarrow \boldsymbol{\beta}_* / \max(1, \|\boldsymbol{\beta}_*\|^2)$.

6.5 Experiments

The experiments are designed to answer the following research questions (RQs):

- **RQ 1** How does TransCF perform compared with other state-of-the-art competitors?

- **RQ 2** Are the newly introduced regularizers tailored to TransCF beneficial for the performance of TransCF?
- **RQ 3** Do the user–item specific translation vectors indeed translate the users close to their corresponding items (as illustrated in Figure 6.1)?
- **RQ 4** What is encoded in the translation vectors?
  - *Intensity / Heterogeneity* of user–item relationships.

Table 6.1: Data Statistics. (Rat. denotes the range of ratings, and #Cat. denotes the number of unique item categories.)

Dataset	#Users	#Items.	#Inter.	Density	Rat.	#Cat.
Delicious	1,050	1,196	7,698	0.61%	-	-
Tradesy	3,352	5,547	32,710	0.13%	-	-
Ciao	6,760	11,166	146,996	0.19%	1-5	28
Amazon	59,089	17,969	332,236	0.03%	1-5	45
Bookcr	19,571	39,702	605,178	0.08%	1-10	-
Pinterest	55,187	9,329	1,462,895	0.28%	-	-
Flixster	69,482	25,687	8,000,690	0.45%	0.5-5.0	-

Datasets. We evaluate our proposed method on *seven* real-world datasets: Delicious [144], Tradesy [105], Ciao [142], Amazon Cellphone and Accessories (Amazon C&A) [17], Bookcrossing [145], Pinterest [146], and Flixster [58]. Delicous, Tradesy and Pinterest datasets contain implicit feedback records, i.e., bookmark, purchased and pinning history. Ciao and Amazon C&A datasets contain rating information of users given to items, and also contain item category information, which will later be used in our experiments pertaining to verifying the heterogeneity of user–item relationships. Bookcrossing dataset contains both the rating (from 1 to 10) information and the implicit feedback (denoted as 0) from users on items. Flixster dataset contains the rating (from 0.5 to 5.0 by an interval of 0.5) information. For datasets with rating information, we regard each observed rating as an implicit feedback record as done in previous research [26, 24, 105, 19, 17, 89, 139]. We use several explicit feedback datasets and convert them into implicit feedback to be able to experimentally ascertain that the translation vectors indeed encode the intensity of user–item relationships; this is not possible with pure implicit feedback datasets as the interactions are not labeled. We include Bookcrossing and Flixster datasets with 10-level ratings in addition to the 5-level ratings (Ciao and Amazon C&A) to verify that TransCF can better infer knowledge regarding the intensity of user–item relationships when finer grained user preferences are given. We remove users and items having

fewer than five ratings. The statistics of the preprocessed datasets used in our experiments are summarized in Table 6.1.

Evaluation Protocol and Metrics. We employ the widely used leave-one-out evaluation protocol [105, 24, 89, 26, 19, 147, 17], whereby the last interacted item for each user is held out for testing, and the rest are used for training. Since it is time consuming to rank all the items in $\mathcal{I}$, we sample 99 items for each user that the user had not interacted with, and compute the ranking scores for those items plus the user's test item (100 in total for each user) [24, 147]. For each user, we additionally hold out the last interacted item from the training data for validation set on which we tune the hyperparameters for all the methods. As we are focused on recommendations for implicit feedback, we employ two ranking metrics widely used for evaluating the performance of recommender systems [19, 24]: hit ratio (H@N) and normalized discounted cumulative gain (N@N). H@N measures whether the item is present in the top-N list, whereas N@N is a position-aware ranking metric that assigns higher scores to hits at upper ranks.

Table 6.2: Test performance of different methods. Best results are in bold face. (*Imp.* denotes the improvement of TransCF over the best competitor, which is CML.)

Datasets	Metrics	BPR	FISM	AoBPR	eALS	CDAE	NeuMF	CML	TransCF^{dot}	TransCF^{alt}	TransCF	*Imp.*
Delicious	H@10	0.1981	0.2203	0.2243	0.1992	0.1319	0.1164	0.2470	0.2150	0.2174	**0.2586**	4.70%
	H@20	0.3177	0.3391	0.3602	0.2942	0.2414	0.2171	0.3649	0.3377	0.3084	**0.3786**	3.75%
	N@10	0.1122	0.1124	0.1114	0.1035	0.0674	0.0558	0.1389	0.1101	0.1281	**0.1475**	6.19%
	N@20	0.1418	0.1424	0.1452	0.1271	0.0949	0.0789	0.1678	0.1412	0.1494	**0.1781**	6.14%
Tradesy	H@10	0.2481	0.2676	0.2597	0.2058	0.1652	0.1167	0.3031	0.2846	0.2648	**0.3198**	5.51%
	H@20	0.4174	0.4109	0.4256	0.3314	0.2867	0.2290	0.4413	0.4266	0.3823	**0.4505**	2.08%
	N@10	0.1248	0.1309	0.1300	0.1042	0.0831	0.0538	0.1685	0.1449	0.1466	**0.1767**	4.87%
	N@20	0.1673	0.1670	0.1715	0.1356	0.1136	0.0817	0.2031	0.1806	0.1760	**0.2095**	3.15%
Ciao	H@10	0.1569	0.2100	0.1873	0.1419	0.1770	0.1535	0.2085	0.2011	0.1991	**0.2292**	9.93%
	H@20	0.2811	0.3482	0.3146	0.2570	0.3153	0.2788	0.3337	0.3185	0.3270	**0.3740**	12.08%
	N@10	0.0751	0.1027	0.0891	0.0670	0.0862	0.0741	0.1053	0.1017	0.0989	**0.1167**	10.83%
	N@20	0.1063	0.1374	0.1209	0.0957	0.1208	0.1040	0.1358	0.1311	0.1309	**0.1525**	12.30%
Book-crossing	H@10	0.2425	0.2178	0.2563	0.1655	0.2244	0.2286	0.2885	0.2802	0.2828	**0.3329**	15.39%
	H@20	0.3761	0.3938	0.3916	0.2864	0.3610	0.3747	0.4053	0.3932	0.4069	**0.4744**	17.05%
	N@10	0.1250	0.1002	0.1338	0.0791	0.1164	0.1158	0.1663	0.1618	0.1578	**0.1865**	12.15%
	N@20	0.1585	0.1444	0.1676	0.1093	0.1506	0.1482	0.1956	0.1903	0.1890	**0.2221**	13.55%
Amazon C&A	H@10	0.2489	0.2470	0.2646	0.2161	0.2817	0.1317	0.3011	0.3003	0.3184	**0.3436**	14.11%
	H@20	0.3821	0.3782	0.3946	0.3480	0.4117	0.2390	0.4123	0.4184	0.4509	**0.4658**	12.98%
	N@10	0.1276	0.1247	0.1391	0.1064	0.1613	0.0613	0.1752	0.1648	0.1766	**0.2019**	15.24%
	N@20	0.1610	0.1577	0.1718	0.0739	0.1939	0.0880	0.2031	0.1945	0.2094	**0.2323**	14.38%
Pinterest	H@10	0.4759	0.4444	0.4921	0.3301	0.5244	0.4546	0.5378	0.5485	0.4899	**0.5504**	2.34%
	H@20	0.7564	0.6720	0.7618	0.5621	0.7393	0.6852	0.7771	0.7822	0.7514	**0.8108**	4.34%
	N@10	0.2034	0.2048	0.2143	0.1373	0.2644	0.2165	0.2558	0.2549	0.2259	**0.2580**	0.86%
	N@20	0.2744	0.2626	0.2827	0.1959	0.3188	0.2748	0.3166	0.3143	0.2922	**0.3242**	2.40%
Flixster	H@10	0.6836	0.5985	0.6904	0.6320	0.6797	0.6596	0.7009	0.6014	0.7123	**0.7309**	4.28%
	H@20	0.8087	0.7597	0.8124	0.7359	0.7973	0.7816	0.8081	0.7147	0.8163	**0.8374**	3.63%
	N@10	0.3701	0.2794	0.3830	0.3513	0.4526	0.4588	0.4608	0.3417	0.4704	**0.4986**	8.20%
	N@20	0.4020	0.3206	0.4140	0.3778	0.4824	0.4895	0.4881	0.3705	0.4969	**0.5257**	7.70%

Methods Compared. As TransCF is a pair-wise *1) learning-to-rank* method based on *2) metric learning* that employs *3) neighborhood information*, we choose the following baselines.

1. Learning-to-rank baselines.
    - Point-wise methods.
        - **eALS** [89]: The state-of-the-art MF–based method for implicit feedback that non-uniformly weights the unobserved interactions based on the item popularity.
        - **NeuMF** [24]: A pointwise neural collaborative filtering framework for implicit feedback that combines MF and multi-layer perceptron (MLP). We report the best results from among MF, MLP, and NeuMF.
    - Pairwise methods.
        - **BPR** [26]: A pairwise learning-to-rank method for implicit feedback in which observed items are assumed to be highly preferred by users to unobserved items.
        - **AoBPR** [90]: An extension of BPR that samples popular items as negative feedback with a higher probability.
2. Neighborhood–based baselines.
    - **FISM** [139]: A neighborhood–based recommendation method based on MF in which a user is represented by the items that the user has interacted with as in Eqn. 6.4.
    - **CDAE** [28]: The state-of-the-art neighborhood–based method in which the user–item relationship is computed between a user and his neighbors, i.e., the items that the user has previously interacted with.
3. Metric learning–based baselines & Ablations of TransCF.
    - **CML** [25]: The state-of-the-art metric learning–based recommendation method for implicit feedback in which Euclidean distance is used for the scoring function. i.e., $s(u,i) = -\|\boldsymbol{\alpha}_u - \boldsymbol{\beta}_i\|_2^2$.
    - **TransCF**$^{\text{dot}}$: An ablation of TransCF in which instead of Euclidean distance, inner product is used for the scoring function. i.e., $s(u,i) = (\boldsymbol{\alpha}_u + \boldsymbol{r}_{ui})^T \boldsymbol{\beta}_i$.
    - **TransCF**$^{\text{alt}}$: Another ablation of TransCF in which the translation vector is computed by $\boldsymbol{r}_{ui} = f(\boldsymbol{\alpha}_u, \boldsymbol{\beta}_i)$. i.e., we employ the current user and item embeddings for constructing $\boldsymbol{r}_{ui}$ *instead of their neighborhoods.*

Implementation Details. For each data, the hyperparameters are tuned on the validation set by grid searches with $K \in \{8, 16, 32, 64, 128\}$, $\eta \in \{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}$, $\gamma \in \{0.0, 0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$, $\lambda, \lambda_{\text{nbr}}, \lambda_{\text{dist}} \in \{0.0, 0.001, 0.01, 0.1\}$, where λ is the regularization coefficient for the baseline methods. We report the test performance measured using the hyperparameter values that give the best HR@10 on the validation set. For reliability, we repeat our evaluations five times with different random seeds for the model initialization, and we report mean test errors. We fix the number of samples in a mini-batch to 1000 for mini-batch SGD.

6.5.1 Performance Analysis

Recommendation performance (RQ 1)

Table 6.2 shows the performance of different methods in terms of various ranking metrics. We have the following observations from Table 6.2. 1) We observe that CML outperforms the MF–based competitors (BPR, FISM, AoBPR, eALS, and NeuMF). This is consistent with the previous work [25], indicating that metric learning approaches overcome the inherent limitation of MF by learning a metric space wherein the triangle inequality is satisfied. 2) We observe that TransCF considerably outperforms the state-of-the-art competitor, namely CML, by up to 17.05% (achieved for HR@20 on Bookcrossing dataset). This verifies the benefit of the translation vectors that translate each user toward items according to the user's relationships with those items. 3) $\text{TransCF}^{\text{alt}}$ generally performs worse than CML, which implies that the translation vectors should be carefully designed, or else the performance will rather deteriorate. 4) The superior performance of TransCF over $\text{TransCF}^{\text{alt}}$ confirms that incorporating the neighborhood information is indeed crucial in collaborative filtering [73, 139, 28]. 5) $\text{TransCF}^{\text{dot}}$ generally performs worse than CML, which verifies that the inner product operation limits the performance despite the benefit of the translation mechanism adopted to $\text{TransCF}^{\text{dot}}$. 6) By further comparing HR@10 of $\text{TransCF}^{\text{alt}}$ on Delicious and Bookcrossing datasets (0.2174 and 0.2828 in Table 6.2) with HR@10 of TransCF without any regularization (0.2388 and 0.2983 in Table 6.3 when $\lambda_{\text{dist}} = \lambda_{\text{nbr}} = 0.0$), we observe that TransCF without the regularizers still outperforms $\text{TransCF}^{\text{alt}}$. This again verifies that the neighborhood information itself is indeed beneficial even without the help of the neighborhood regularizer.

Table 6.3: Effect of the regularization coefficients.

Delicious (HR@10)		λ_{nbr}			
		0.0	0.001	0.01	0.1
λ_{dist}	0.0	0.2388	0.2388	0.2401	0.2454
	0.001	0.2375	0.2401	0.2401	0.2454
	0.01	0.2401	0.2401	0.2427	0.2467
	0.1	0.2520	0.2520	0.2533	**0.2586**
Bookcrossing (HR@10)		λ_{nbr}			
		0.0	0.001	0.01	0.1
λ_{dist}	0.0	0.2983	0.3033	0.3227	0.3185
	0.001	0.3038	0.3074	0.3236	0.3181
	0.01	0.3213	0.3219	0.3264	0.3187
	0.1	**0.3329**	0.3329	0.3324	0.3151

Benefit of regularizers (RQ 2)

Table 6.3 shows the effect of the regularization coefficients on the performance of TransCF on Delicious and Bookcrossing datasets, where λ_{nbr} and λ_{dist} denote the strengths of the neighborhood regularizer (reg_{nbr}) and of the distance regularizer (reg_{dist}), respectively. Larger values imply a stronger contribution of the regularizer to the model, and $\lambda_* = 0.0$ indicates no regularization. We have the following observations: 1) Both regularizers are indeed beneficial for the model performance, and their impact varies across different datasets. 2) Although reg_{nbr} is beneficial, its impact on the model performance decreases as the reg_{dist} dominates the model; i.e., as λ_{dist} increases. 3) reg_{dist} is more helpful for the model performance compared with reg_{nbr}; the benefit of reg_{dist} becomes more clear on Bookcrossing dataset in which the user–item relations are more complex compared with Delicious dataset. This confirms that explicitly pulling each translated user toward all of the positive items rather than merely pushing negative items away from each user helps to model the complex user–item interactions, which aligns with our motivation for the distance regularizer reg_{dist} as mentioned in Section 6.4.4.

Table 6.4: Analysis on the user–item specific translation vectors. (*Obs./Unobs.* denotes results on observed/unobserved user–item interactions.)

Dataset	*Obs.*	*Unobs.*	Dataset	*Obs.*	*Unobs.*
Delicious	64.63%	43.75%	Amazon	75.57%	31.96%
Tradesy	56.02%	43.01%	Pinterest	36.25%	33.08%
Ciao	54.63%	38.42%	Flixster	22.24%	2.88%
Bookcr.	55.42%	35.57%			

6.5.2 Qualitative Evaluations

Benefit of translation vectors (RQ 3)

In this section, we conduct experiments to verify whether the translation vectors learned by TransCF indeed translate each user closer to the observed (positive) items as in **Toy example** illustrated in Figure 6.1. To this end, for each user u and his observed item i, we check whether the following holds:

$$\|\boldsymbol{\alpha}_u - \boldsymbol{\beta}_i\|_2^2 > \|\boldsymbol{\alpha}_u + \boldsymbol{r}_{ui} - \boldsymbol{\beta}_i\|_2^2 \tag{6.7}$$

That is, we expect the distance between the item embedding vector $\boldsymbol{\beta}_i$ of observed item i and the translated user embedding vector $(\boldsymbol{\alpha}_u + \boldsymbol{r}_{ui})$ to be smaller than the distance between the item embedding vector $\boldsymbol{\beta}_i$ and the user embedding vector $\boldsymbol{\alpha}_u$ before translation. We calculate the percentage of observed/unobserved user–item interaction pairs that satisfy Equation 6.7 among all possible observed/unobserved pairs. We sample as many unobserved items as the number of observed items for each user for the comparisons. Here, we expect more observed pairs to satisfy Equation 6.7 than unobserved pairs. Table 6.4 shows the results on the seven datasets.

We observe that users are generally translated closer toward their observed (positive) items, and at the same time translated farther away from the unobserved (negative) items. For instance, for Amazon C&A dataset, 75.57% of the observed user–item interactions satisfy Equation 6.7, whereas only 31.96% of the unobserved interactions satisfy it, which conversely implies that users in 68% ($\approx 100 - 31.96$) of the unobserved interactions translate users farther away from the unobserved items. We also note that for Flixster dataset, although only 22.24% of the observed interactions satisfy Equation 6.7, the overwhelming majority of the unobserved interactions, i.e., 97% ($\approx 100 - 2.88$), violate it, which means that the effect of users being translated away from the unobserved items results in placing the translated users relatively closer toward their observed items. Hence, we argue that by simultaneously translating a user toward his relevant items and away from his irrelevant items, TransCF achieves superior recommendation performance.

What is encoded in the translation vectors? (RQ 4)

In this section, we investigate why TransCF outperforms the state-of-the-art methods. For this purpose, we conduct various experiments with the translation vectors to verify our claims that the translation vectors encode the i) **intensity**, and the ii) **heterogeneity** of user–item relationships in implicit feedback.

Table 6.5: Rating classification accuracy using translation vectors. (*Rand* denotes a random classifier, and RF denotes the random forest classifier [148].)

Acc.(%)	Ciao		Amazon		BookCr.[3]		Flixster	
	Rand	RF	*Rand*	RF	*Rand*	RF	*Rand*	RF
CML	|	50.3	|	50.1	|	39.1	|	20.5
TransCF$^{\text{emb}}$	19.9	50.3	20.1	50.3	13.8	40.1	10.0	20.5
TransCF	|	**53.0**	|	**50.8**	|	**43.7**	|	**23.4**
vs. CML	-	5.3%	-	1.5%	-	11.7%	-	14.2%

***Intensity* of user–item interactions.** With regard to the first claim, i.e., intensity, we assume that the rating information is a proxy for the intensity of user–item relationships. In other words, the higher the rating of an item given by a user, the higher the intensity of the user–item relationship. Since Ciao, Amazon C&A, Bookcrossing and Flixster datasets contain rating information, we use them to study whether we can conversely infer some knowledge about ratings using the translation vectors *even though the ratings are not utilized during the model training*. To this end, we perform rating classification to predict the rating of user u on item i given the user–item specific translation vector $\boldsymbol{r}_{ui}$ as input; for instance, there are five classes for datasets with ratings ranging from 1 to 5. Note that we perform classification instead of regression to clearly emphasize the difference between the numbers; the RMSE values of regression usually differ in the second decimal place [73]. To ascertain the benefit of the translation vectors learned by TransCF, i.e., $\boldsymbol{r}_{ui}^{\text{TransCF}}$, we compare the rating classification performance with that of CML. Since CML does not model the translation vectors [25], we alternatively define the translation vectors for CML, i.e., $\boldsymbol{r}_{ui}^{\text{CML}} := (\boldsymbol{\alpha}_u - \boldsymbol{\beta}_i)$, where $\boldsymbol{\alpha}_u$ and $\boldsymbol{\beta}_i$ are both trained by CML. Likewise, we additionally define synthetic translation vectors $\boldsymbol{r}_{ui}^{\text{TransCF}^{\text{emb}}} := (\boldsymbol{\alpha}_u - \boldsymbol{\beta}_i)$ for TransCF , where $\boldsymbol{\alpha}_u$ and $\boldsymbol{\beta}_i$ are both trained by TransCF, and name this method TransCF$^{\text{emb}}$. Table 6.5 summarizes the classification results[4]. We observe that the rating classification accuracy on translation vectors of TransCF outperforms that of CML and TransCF$^{\text{emb}}$, and that the improvement is the greatest on Flixster dataset. This implies that the **intensity** of user–item relationships is encoded in the translation vectors learned by TransCF, which explains the superior performance of TransCF as shown in Table 6.2. However, we note that the improvements of classification accuracies compared with CML are not sufficiently large on Ciao and Amazon C&A

[3]We regard ratings of less than 4 as a single class as they account for only 3.86%.

[4]We balance the class distribution by randomly sampling a fixed number of samples (num. samples in the minority class) from each class. We then perform 5-fold cross validation. We use the default setting of the random forest classifier in Scikit-learn [149], but the accuracies could be further improved by tuning the classifier.

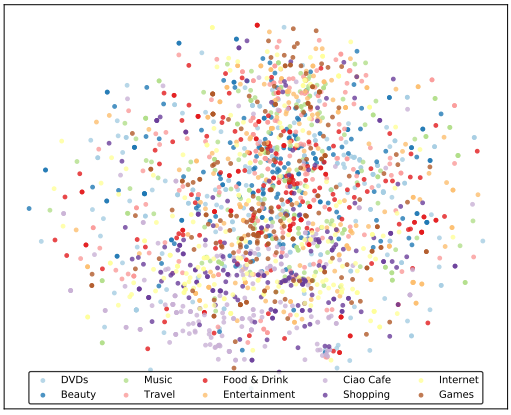
Table 6.6: Analysis on the user–item specific translation vectors regarding the ratings. (*Eqn 6.7.* denotes the percentage of interactions that satisfy Equation 6.7, and Ptn. denotes the portion of each class.)

	Rating						
Ciao	1	2	3	4	5		
Eqn 6.7	61.5%	51.4%	55.4%	52.2%	55.4%		
Ptn.	4.8%	5.1%	11.4%	29.0%	49.7%		
Amazon	1	2	3	4	5		
Eqn 6.7	76.7%	76.3%	75.7%	75.2%	75.4%		
Ptn.	7.0%	5.7%	10.7%	20.1%	56.5%		
BookCr.	1-4	5	6	7	8	9	10
Eqn 6.7	55.3%	52.7%	55.2%	56.1%	57.2%	58.4%	58.8%
Ptn.	3.8%	10.3%	7.9%	17.0%	24.5%	17.3%	19.2%
Flixster	0.5-2.5	3.0	3.5	4.0	4.5	5.0	
Eqn 6.7	19.6%	19.9%	19.9%	22.2%	25.7%	27.2%	
Ptn.	17.3%	17.0%	16.8%	19.6%	10.1%	19.2%	

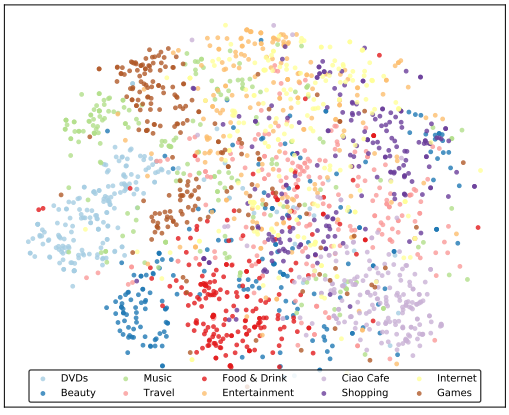
datasets, reaching only 5.3% and 1.5%, respectively. This motivates us to further investigate the translation vectors with regard to the intensity of user–item relationships.

To further study the translation vectors, we group the results of the observed user–item interactions from Table 6.4 according to their ground truth ratings, and calculate the percentage of interactions that satisfy Equation 6.7 in each rating group. Table 6.6 shows the results. Under the assumption that higher ratings imply a higher intensity of user–item relationships, we expect more observed interactions to satisfy Equation 6.7 in higher rating groups. However, we observe that the results on Ciao and Amazon C&A datasets do not agree with our expectation, which explains the relatively small improvements in Table 6.5. We conjecture that it is inherently challenging to infer users' fine-grained preferences from the user–item interactions of Ciao and Amazon C&A datasets, because 1) the range of the ratings is relatively small (integers from 1 to 5), and more importantly, 2) the majority (over 75%) of interactions belong to ratings from 4 to 5. In contrast, for Bookcrossing[5] and Flixster datasets whose ratings are in a wider range (from 1 to 10, and from 0.5 to 5.0, respectively), and relatively evenly distributed throughout the rating classes, the percentage of interactions satisfying Equation 6.7 increases as the ratings increase from 5 to 10 (for Flisxter, from 3.0 to 5.0). These results on Bookcrossing and Flixster datasets (Table 6.6) are in line with the results in Table 6.5, where the relative improvement of the classification results of TransCF is the highest for Bookcrossing and Flixster datasets. To summarize our findings from Tables 6.5 and 6.6, we conclude that the inten-

[5]Note that for Bookcrossing dataset, the number of observed interactions whose ratings are less than 4 account for a small proportion (3.86%), and are therefore negligible.



(a) Visualization of $\boldsymbol{r}_{ui}^{CML}$



(b) Visualization of $\boldsymbol{r}_{ui}^{\mathsf{TransCF}}$

Figure 6.4: t-SNE visualization of translation vectors of Ciao dataset regarding the item categories.

sity of user–item relationships is indeed encoded in the translation vectors, and that it becomes clearer with datasets from which users' preferences can be more precisely inferred, e.g., Bookcrossing and Flisxter datasets.

However, up to this point, it is still uncertain why the improvements of TransCF on Ciao and Amazon C&A datasets are considerable in terms of the quantitative evaluation (Table 6.2), even though the rating information (intensity) is not as clearly encoded in the translation vectors as in the case of Bookcrossing and Flixster dataset. Hence, in the following section, we study whether the translation vectors of TransCF trained on Ciao and Amazon C&A datasets encode meaningful information other than the rating information.

***Heterogeneity* of user–item interactions.** In this section, we investigate the translation vectors from a different perspective, i.e., heterogeneity. Recall that TransCF aims to translate each user toward multiple items according to the

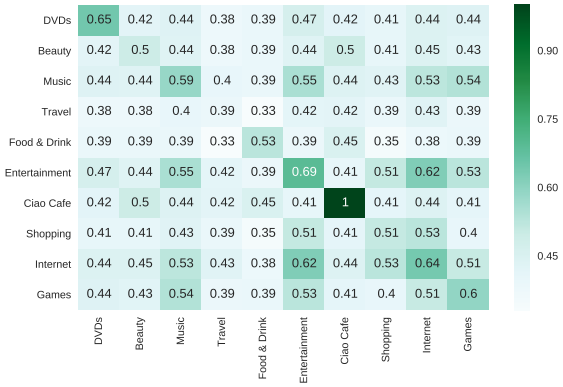


Figure 6.5: Heat map of cosine similarity between translation vectors of Ciao dataset.

user's heterogeneous tastes in various item categories, implying that the translation vectors encode information related to item categories. To verify this, we study whether we can conversely infer the categories of items using the translation vectors, *even though the item category information is not utilized during the training of TransCF*. To this end, we label each translation vector $\boldsymbol{r}_{ui}$ with its corresponding category, and select vectors from the ten most frequently appearing categories to perform classification. Table 6.7a shows the item category classification accuracy on the translation vectors[4]. We observe that TransCF considerably outperforms CML. This implies that the user–item specific translation vectors indeed encode the **heterogeneity** of the user–item relationships with regard to users' tastes in various item categories, which provides a justification for the superior performance of TransCF on Ciao and Amazon C&A datasets.

We further conduct another experiment on the item embedding vectors $\boldsymbol{\beta}_i$ to ascertain that the considerable performance improvement with TransCF is indeed derived from the translation vectors; not from the high-quality item embedding vectors. This time, we select items whose categories belong to the ten most frequent appearing categories. Table 6.7b shows the classification accuracy on the item embedding vectors $\boldsymbol{\beta}_i$ for all $i \in \mathcal{I}$ trained by CML and TransCF. We observe that CML and TransCF show comparable classification performance, implying that the superior performance of TransCF is derived not from the high-quality item embedding vectors, but from the translation vectors.

To provide a more intuitive understanding of the numerical results shown in Table 6.7a, we visualize in Figure 6.4 the translation vectors learned by CML and TransCF on Ciao dataset by using t-distributed Stochastic Neighbor Embedding[6](t-SNE) [150] with perplexity 30. Each point represents a translation

[6]We sample 200 samples from each category for visualization.

Dataset	Method	Rand.	Random Forest
Ciao	CML		67.86±0.47%
	TransCF^{emb}	10.01%	67.27±0.28%
	TransCF		**80.97**±0.73%
Amazon C&A	CML		54.26±0.74%
	TransCF^{emb}	10.40%	54.85±0.51%
	TransCF		**81.24**±0.46%

(a) Classification on translation vectors ($\boldsymbol{r}_{ui}$).

Dataset	Method	Rand.	Random Forest
Ciao	CML	10.92%	80.41±1.59%
	TransCF		81.61±1.54%
Amazon C&A	CML	9.40%	47.94±3.34%
	TransCF		47.90±2.54%

(b) Classification on item embeddings ($\boldsymbol{\beta_i}$).

Table 6.7: Results of item category classification.

vector, and the color represents the item category. We can clearly see that the translation vectors of TransCF are generally grouped together according to their corresponding categories, unlike CML; this supports the superior classification results of TransCF over CML in Table 6.7a.

Lastly, based on the above observation that similar translation vectors encode similar information with regard to item categories, we further investigate whether the translation vectors might even reveal the correlations among the item categories. To this end, we compute the sum of the cosine similarity scores between all pairs of translation vectors from each category. We use the same sampled translation vectors that were used for the visualization in Figure 6.4. Figure 6.5 shows the heat map of cosine similarity between translation vectors learned by TransCF on Ciao dataset with regard to the item categories. The number in each cell denotes the average cosine similarity score normalized by the overall largest score. We observe that the similarity score is generally the highest within the same item category (the diagonal line). Moreover, interestingly, semantically related categories also show high correlations. For example, "Entertainment" is highly correlated with "Internet", "Games", "Music", and "Shopping" all of which are related to entertainment. From the above analyses, we can conclude that TransCF can even determine the correlations among the item categories by encoding the heterogeneity of user–item relationships into the translation vectors, which again helps explain the superior performance of TransCF. This experiment also verifies our assumption illustrated in Figure 6.1 that the angles between the translation vectors reflect the heterogeneity of the user–item relationships regarding the user's tastes in various item categories.

6.6 Related Work

Recommendations for Implicit Feedback. Although explicit feedback such as rating or review comment is a valuable source of information that reveals user preferences, in most cases it is difficult to obtain a large quantity of such data. Hence, the vast majority of past research has focused on recommendations for implicit feedback [107, 61, 151, 152, 26, 52, 89, 105]. These methods generally adopt the matrix factorization (MF) technique, which uses the inner product to model the similarity of user–item pairs [73]. However, the inner product violates the triangle inequality, and thus it may fail to capture fine-grained user preferences [25].

Incorporating the neighborhood information [141] into CF has been shown to be effective for memory–based [46] and model–based [140, 139, 28, 73] CF. ItemCF [46] computes the similarity scores, such as Pearson Correlation and Cosine Similarity, between users based on the items they interacted with in the past. SLIM [140] and FISM [139] improve ItemCF by learning the item-item similarity directly from the data through factorizing the item similarity matrix as a product of two latent factor matrices. CDAE [28] is the state-of-the-art neighborhood–based CF method implemented by using denoising auto-encoder. However, previously proposed neighborhood–based CF methods are generally based on MF, and thus suffer from the violation of triangle inequality.

In the following, we briefly introduce metric learning approaches that address this limitation of MF.

Metric Learning. Metric learning [153] learns a distance metric that preserves the distance relation among the training data; i.e., it assigns shorter distances to semantically similar pairs. It has been popularized in various domains such as computer vision [154] and natural language processing [155]. Recently, metric learning approaches have been adopted in collaborative filtering to address the limitation of MF. Khoshneshin and Street firstly introduced the Euclidean embedding scheme for explicit feedback-based CF, a scheme in which users and items are embedded according to their Euclidean similarity rather than their inner product [134]. For music recommendation, Chen *et al.* represent songs as points in Euclidean space, and model the transition probability based on the Euclidean distance between songs [135]. For point-of-interest (POI) recommendation, Feng *et al.* model personalized check-in sequences by projecting each POI into one object in Euclidean space [136]. The above methods can be subsumed by the recently proposed CML [25], which is a general recommendation framework for implicit feedback. CML projects users and items into a common Eu-

clidean space in which the similarity of a user–item pair is computed based on the Euclidean distance between the latent vectors of the user and of the item. Given user u and item i, the scoring function $s(u, i)$ of CML is computed by: $s(u, i) = -\|\boldsymbol{\alpha}_u - \boldsymbol{\beta}_i\|_2^2$. In spite of its state-of-the-art performance, CML projects each user to a single point, and thus it does not suffice for modeling the intensity and the heterogeneity of the user–item relationships in implicit feedback.

Knowledge Graph Embedding. Knowledge graph embedding refers to the projection of entities and relations in knowledge graphs into low-dimensional vector spaces. Among various knowledge graph embedding methods, translation–based methods [137, 138] have shown state-of-the-art performance and scalability compared with traditional factorization–based embedding methods [156, 157]. Recently, the translation mechanism has been adopted for recommendation algorithms. Zhang *et al.* [158] integrate collaborative filtering and a knowledge base for recommendation by adopting TransR [138] to extract the structural knowledge of items from knowledge graphs. He *et al.* [19] embed items into a transition space, where each user is modeled by a translation vector to model the transition from the previously consumed item to the next item. However, our work is distinguished from the above methods in that we adopt the translation mechanism to model the latent relationships of implicit user–item interactions rather than to extract some knowledge from knowledge graphs, and we do not model the temporal information. Hence, we do not compare them with TransCF in our experiments.

6.7 Conclusion

Each implicit user–item interaction encodes a different intensity of user–item relationship, and the relationship is heterogeneous regarding the user's taste in different item categories. In this paper, we propose a novel metric learning–based recommendation method called TransCF that captures not only the intensity and the heterogeneity of user–item relationships in implicit feedback, but also the complex nature of CF, all of which have been overlooked by previous metric learning–based recommendation approaches. TransCF employs the neighborhood information of users and items to construct translation vectors whereby a user is translated toward items according to his relationships with the items. Through extensive experiments, we demonstrate that TransCF considerably outperforms several state-of-the-art methods by generating meaningful translation vectors. Regarding possible directions for future studies, refer to Section 6.4.3.

VII. Conclusion

In this dissertation, we proposed several data-centric approaches to alleviating the data sparsity issue for building recommender systems. We categorized our approaches into two categories, i.e., auxiliary data-driven approach and algorithmic approach, and introduced several research directions in both approaches.

In the future, we plan to continue our research on recommender system, specifically, on **cross-domain recommendation**. Cross-domain recommender system aims to transfer knowledge from a rich auxiliary domain to a sparse target domain to alleviate the data sparsity problem of the target domain. However, while a vast majority of previous works on cross-domain recommendation require multiple domains to share either common users or items or both, such requirement is hard to fulfill in practice as not many systems provide such platform where users simultaneously rate items from two completely different domains. In other words, traditional cross-domain recommendation methods can be useful to only large Internet companies with various online services in different domains whereas small companies (start-ups) that focus only on a single domain cannot benefit from the traditional methods. To address the above challenge, we are interested in a cross-domain recommendation method that does not require users or items to overlap between two different domains. Particularly, by interpreting users and items as nodes in a graph, we am interested in adopting **graph representation** learning techniques to **align users and items in multiple graphs**. Moreover, by projecting users and items into a graph, we will be able to use various auxiliary information regarding the nodes as **distant supervision** to better learn the representation of nodes, which in turn will be eventually used for recommendation.

요 약 문

본 논문은 추천 시스템의 고전적인 문제인 데이터 부족 현상을 해결하기 위한 데이터 중심적 접근 방법을 제안한다. 특히, 행렬분해 기법, 메트릭 학습 기법을 기반으로 한 방법들을 소개한다. 데이터 중심적 접근 방법은 크게 두가지로 나뉜다. 첫번째는, 사용자 및 품목에 관련된 보조 데이터를 기반으로 한 접근 방법이며, 두번째는, 알고리즘적인 접근 방법이다. 본 학위 논문에서는 다양한 실험을 통해 본 논문에서 제안한 방법들의 우수성을 입증하며, 이는 추천 시스템의 데이터 부족 현상을 해결하는데 새로운 방향을 제시 한다.

References

[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[2] Xin Wang, Wei Lu, Martin Ester, Can Wang, and Chun Chen. Social recommendation with strong and weak ties. In *CIKM*. ACM, 2016.

[3] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008.

[4] Hao Ma, Irwin King, and Michael R Lyu. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 203–210. ACM, 2009.

[5] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[6] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, pages 123–129, 2015.

[7] Tong Zhao, Julian McAuley, and Irwin King. Leveraging social connections to improve personalized ranking for collaborative filtering. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 261–270. ACM, 2014.

[8] Steffen Rendle. Social network and click-through prediction with factorization machines. In *KDD Cup 2012 Workshop*.

[9] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[10] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015.

[11] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 233–240. ACM, 2016.

[12] Donghyun Kim, Chanyoung Park, Jinoh Oh, and Hwanjo Yu. Deep hybrid recommender systems via exploiting document context and statistics of items. *Information Sciences*, 417:72–87, 2017.

[13] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[14] Ruining He and Julian McAuley. VBPR: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 144–150, 2016.

[15] Ruining He, Chunbin Lin, Jianguo Wang, and Julian McAuley. Sherlock: Sparse hierarchical embeddings for visually-aware one-class collaborative filtering. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3740–3746, 2016.

[16] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

[17] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of WWW*. International World Wide Web Conferences Steering Committee, 2016.

[18] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 4642–4650. IEEE, 2015.

[19] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation. In *Proceedings of RecSys*, pages 161–169. ACM, 2017.

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

[21] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[22] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. ACM, 2010.

[23] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222. SIAM, 2010.

[24] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th WWW*. International World Wide Web Conferences Steering Committee, 2017.

[25] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. Collaborative metric learning. In *Proceedings of WWW*. International World Wide Web Conferences Steering Committee, 2017.

[26] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[27] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM, 2015.

[28] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 2016.

[29] Sheng Li, Jaya Kawale, and Yun Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 811–820. ACM, 2015.

[30] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[31] Matt Marshall. *Aggregate Knowledge raises 5M from Kleiner, on a roll*, 2006.

[32] A Tejeda-Lorente, Juan Bernabé-Moreno, Carlos Porcel, and Enrique Herrera-Viedma. Integrating quality criteria in a fuzzy linguistic recommender system for digital libraries. *Procedia Computer Science*, 31:1036–1043, 2014.

[33] Alvaro Tejeda-Lorente, Carlos Porcel, Eduardo Peis, Rosa Sanz, and Enrique Herrera-Viedma. A quality based recommender system to disseminate information in a university digital library. *Information Sciences*, 261:52–69, 2014.

[34] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

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

[36] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[37] Xiwang Yang, Harald Steck, Yang Guo, and Yong Liu. On top-k recommendation using social networks. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 67–74. ACM, 2012.

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

[39] G. Guo, J. Zhang, and N. Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *IJCAI*, 2013.

[40] G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith. Etaf: An extended trust antecedents framework for trust prediction. In *ASONAM*, 2014.

[41] Guang Ling, Michael R Lyu, and Irwin King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 105–112. ACM, 2014.

[42] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[43] Carlos Porcel and Enrique Herrera-Viedma. Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries. *Knowledge-Based Systems*, 23(1):32–39, 2010.

[44] Weilong Yao, Jing He, Guangyan Huang, and Yanchun Zhang. Sorank: incorporating social information into learning to rank models for recommendation. In *Proceedings of the companion publication of the 23rd international conference on World wide web companion*, pages 409–410. International World Wide Web Conferences Steering Committee, 2014.

[45] Bo Yang, Yu Lei, Dayou Liu, and Jiming Liu. Social collaborative filtering by trust. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2747–2753. AAAI Press, 2013.

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

[47] Zhi-Dan Zhao and Ming-Sheng Shang. User-based collaborative-filtering recommendation algorithms on hadoop. In *Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on*, pages 478–481. IEEE, 2010.

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

[49] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

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

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

[52] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

[53] Jiliang Tang, Xia Hu, and Huan Liu. Social recommendation: a review. *Social Network Analysis and Mining*, 3(4):1113–1133, 2013.

[54] Mohsen Jamali and Martin Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 397–406. ACM, 2009.

[55] Paolo Massa and Paolo Avesani. Trust-aware recommender systems. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24. ACM, 2007.

[56] Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. Exploiting local and global social context for recommendation.

[57] Jiliang Tang, Huiji Gao, and Huan Liu. mtrust: discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 93–102. ACM, 2012.

[58] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[59] Xiwang Yang, Harald Steck, and Yong Liu. Circle-based recommendation in online social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1267–1275. ACM, 2012.

[60] Sean M McNee, John Riedl, and Joseph A Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI'06 extended abstracts on Human factors in computing systems*, pages 1097–1101. ACM, 2006.

[61] Weike Pan and Li Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI*, volume 13, pages 2691–2697, 2013.

[62] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 269–272. ACM, 2010.

[63] Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[64] John I Marden. *Analyzing and modeling rank data*. CRC Press, 1996.

[65] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008.

[66] Jinoh Oh, Wook-Shin Han, Hwanjo Yu, and Xiaoqian Jiang. Fast and robust parallel sgd matrix factorization. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 865–874. ACM, 2015.

[67] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. Maximum margin matrix factorization for collaborative ranking.

[68] Bo Yang, Yu Lei, Dayou Liu, and Jiming Liu. Social collaborative filtering by trust. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2747–2753. AAAI Press, 2013.

[69] Bin Cao, Nathan N Liu, and Qiang Yang. Transfer learning for collective link prediction in multiple heterogenous domains. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 159–166, 2010.

[70] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[71] Carmen Martinez-Cruz, Carlos Porcel, Juan Bernabé-Moreno, and Enrique Herrera-Viedma. A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling. *Information Sciences*, 311:102–118, 2015.

[72] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173, 2011.

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

[74] Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. *arXiv preprint arXiv:1603.04259*, 2016.

[75] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 233–240, New York, NY, USA, 2016. ACM.

[76] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[77] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 287–296, New York, NY, USA, 2011. ACM.

[78] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. Improving top-k recommendation with truster and trustee relationship in user trust network. *Information Sciences*, 374:100 – 114, 2016.

[79] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. Trecso: Enhancing top-k recommendation with social information. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 89–90, 2016.

[80] Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 91–100. ACM, 2010.

[81] Chaochao Chen, Xiaolin Zheng, Yan Wang, Fuxing Hong, Zhen Lin, et al. Context-aware collaborative topic regression with social matrix factorization for recommender systems. In *AAAI*, volume 14, pages 9–15, 2014.

[82] Sanjay Purushotham, Yan Liu, and C-c J Kuo. Collaborative topic regression with social matrix factorization for recommendation systems. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 759–766, 2012.

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

[84] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015.

[85] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[86] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[87] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: simultaneously exploiting ratings and reviews for recommendation. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2–8. AAAI Press, 2014.

[88] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*. Ieee, 2008.

[89] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR*, pages 549–558. ACM, 2016.

[90] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *WSDM*, pages 273–282. ACM, 2014.

[91] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*. ACM, 2017.

[92] Amazon.com Search Click-Through Rates By Position, 2014.

[93] Konstantina Christakopoulou and Arindam Banerjee. Collaborative ranking with a push at the top. In *WWW*. International World Wide Web Conferences Steering Committee, 2015.

[94] Jun Hu and Ping Li. Decoupled collaborative ranking. In *WWW*, 2017.

[95] Dimitrios Rafailidis and Fabio Crestani. Joint collaborative ranking with social relationships in top-n recommendation. In *CIKM*. ACM, 2016.

[96] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *SIGKDD*. ACM, 2013.

[97] Zeyuan Allen Zhu, Weizhu Chen, Tom Minka, Chenguang Zhu, and Zheng Chen. A novel click model and its applications to online advertising. In *WSDM*. ACM, 2010.

[98] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. *arXiv preprint arXiv:1404.5772*, 2014.

[99] Cheng Li, Yue Lu, Qiaozhu Mei, Dong Wang, and Sandeep Pandey. Click-through prediction for advertising in twitter timeline. In *SIGKDD*. ACM, 2015.

[100] Caroline Lo, Dan Frankowski, and Jure Leskovec. Understanding behaviors that lead to purchasing. In *SIGKDD*. ACM, 2016.

[101] Justin Cheng, Caroline Lo, and Jure Leskovec. Predicting intent using activity logs: How goal specificity and temporal range affect user behavior. In *WWW Companion*, WWW '17 Companion, 2017.

[102] Guimei Liu, Tam T Nguyen, Gang Zhao, Wei Zha, Jianbo Yang, Jianneng Cao, Min Wu, Peilin Zhao, and Wei Chen. Repeat buyer prediction for e-commerce. In *SIGKDD*. ACM, 2016.

[103] Sheng Li, Jaya Kawale, and Yun Fu. Predicting user behavior in display advertising via dynamic collective matrix factorization. In *SIGIR*. ACM, 2015.

[104] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[105] Ruining He and Julian McAuley. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI*, 2016.

[106] Cynthia Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, 10(Oct), 2009.

[107] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie: scaling up to large vocabulary image annotation. In *IJCAI*. AAAI Press, 2011.

[108] Nan Li, Rong Jin, and Zhi-Hua Zhou. Top rank optimization in linear time. In *Advances in neural information processing systems*, 2014.

[109] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *RecSys*, 2012.

[110] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, 2008.

[111] Huayu Li, Richang Hong, Defu Lian, Zhiang Wu, Meng Wang, and Yong Ge. A relaxed ranking-based factor model for recommender system from implicit feedback. In *IJCAI*. AAAI Press, 2016.

[112] Denis Parra, Alexandros Karatzoglou, Xavier Amatriain, and Idil Yavuz. Implicit feedback recommendation via implicit-to-explicit ordinal logistic regression mapping. *CARS-2011*, 2011.

[113] Harikrishna Narasimhan and Shivani Agarwal. A structural svm based approach for optimizing partial auc. In *ICML*, 2013.

[114] Purushottam Kar, Harikrishna Narasimhan, and Prateek Jain. Surrogate functions for maximizing precision at the top. In *ICML*, 2015.

[115] Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems*, 2008.

[116] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. xclimf: optimizing expected reciprocal rank for data with multiple levels of relevance. In *RecSys*. ACM, 2013.

[117] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. Gapfm: Optimal top-n recommendations for graded relevance domains. In *CIKM*. ACM, 2013.

[118] Rana Forsati, Iman Barjasteh, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. Pushtrust: An efficient recommendation algorithm by leveraging trust and distrust relations. In *RecSys*, 2015.

[119] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *WWW*. ACM, 2007.

[120] Ye Chen and Tak W Yan. Position-normalized click prediction in search advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 795–803. ACM, 2012.

[121] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. Context-aware sequential recommendation. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016.

[122] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. A convolutional click prediction model. In *CIKM*. ACM, 2015.

[123] D Ben-Shimon, A Tsikinovsky, M Friedman, B Shapira, L Rokach, and J Hoerle. Recsys challenge 2015 and the yoochoose dataset. In *Proceedings of the 9th ACM conference on Recommender systems*. ACM, 2015.

[124] P. Hart. The condensed nearest neighbor rule (corresp.). *Information Theory, IEEE Transactions on*, 14(3):515–516, May 1968.

[125] Rushi Longadge and Snehalata Dongre. Class imbalance problem in data mining review. *arXiv preprint arXiv:1305.1707*, 2013.

[126] Ivan Tomek. Two modifications of cnn. *IEEE Trans. Syst. Man Cybern.*, 6:769–772, 1976.

[127] Gustavo EAPA Batista, Andre CPLF Carvalho, and Maria Carolina Monard. Applying one-sided selection to unbalanced datasets. In *MICAI 2000: Advances in Artificial Intelligence*. Springer, 2000.

[128] Jorma Laurikkala. *Improving identification of difficult small classes by balancing class distribution*. Springer, 2001.

[129] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem: bagging-, boosting-, and hybrid-based approaches. *Systems, Man, and Cybernetics, IEEE Transactions on*, 42(4):463–484, 2012.

[130] Jerome H Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[131] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[132] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46, 2013.

[133] Parikshit Ram and Alexander G Gray. Maximum inner-product search using cone trees. In *Proceedings of SIGKDD*. ACM, 2012.

[134] Mohammad Khoshneshin and W Nick Street. Collaborative filtering via euclidean embedding. In *Proceedings of RecSys*. ACM, 2010.

[135] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of SIGKDD*. ACM, 2012.

[136] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new poi recommendation. In *Proceedings of AAAI*. AAAI Press, 2015.

[137] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in NIPS*, 2013.

[138] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*. Citeseer, 2015.

[139] Santosh Kabbur, Xia Ning, and George Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of SIGKDD*. ACM, 2013.

[140] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *Proceedings of ICDM*. IEEE, 2011.

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

[142] J. Tang, H. Gao, and H. Liu. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of WSDM*, pages 93–102. ACM, 2012.

[143] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[144] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of RecSys*, 2011.

[145] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of WWW*. ACM, 2005.

[146] Xue Geng, Hanwang Zhang, Jingwen Bian, and Tat-Seng Chua. Learning image and user features for recommendation in social networks. In *Proceedings of ICCV*, 2015.

[147] Hong-Jian Xue, Xin-Yu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. Deep matrix factorization models for recommender systems. 2017.

[148] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[149] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 2011.

[150] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov), 2008.

[151] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local collaborative ranking. In *Proceedings of WWW*. ACM, 2014.

[152] Christopher C Johnson. Logistic matrix factorization for implicit feedback data. In *NIPS*, 2014.

[153] Liu Yang. Distance metric learning: A comprehensive survey. 2006.

[154] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[155] Guy Lebanon. Metric learning for text documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 2006.

[156] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of ICML*, 2011.

[157] Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In *Advances in NIPS*, 2012.

[158] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *Proceedings of SIGKDD*, pages 353–362. ACM, 2016.

Acknowledgements

가장먼저 저의 지도교수님인 유환조 교수님께 감사의 말씀을 드리고 싶습니다. 제가 교수님의 지도를 받게 되어서 큰 행운이었다고 생각합니다. 5년전, 부족한 저를 연구실에 받아주시고, 박사 과정동안 지도해 주시며 많은 기회를 주셔서 정말 감사드립니다. 저를 믿어주시고 아낌없는 지원을 해주신 덕분에 무사히 박사학위를 받을 수 있었습니다. 졸업 후에도 좋은 기회를 주신만큼, 앞으로도 열심히 노력하여 좋은 연구자가 되도록 노력하겠습니다. 그리고 바쁘신 와중에도 제 박사 학위 심사를 수락해 주시고 아낌없는 조언을 해주신 전치혁 교수님, 고영명 교수님, 조민수 교수님, 곽수하 교수님께도 진심으로 감사드립니다.

제가 연구실에 처음 들어와서 연구 방향을 잡지 못하고 있을 때, 바쁘신 와중에도 저와 매주 미팅을 하며 많은 도움을 주신 진오형께 감사드립니다. 초기에 형의 도움이 없었더라면, 제가 박사학위를 받는데 더 많이 힘들었을 것이라고 생각합니다. 그리고 연구 뿐만 아니라, 생활 전반에 걸쳐서 많은 도움을 주셔서 감사드립니다. 그리고 5년간의 포항 생활 동안 연구적, 정신적으로 가장 많은 의지를 하고 가장 가깝게 지냈던 동현이. 실질적으로 모든 연구를 너와 같이 하면서 정말 많은 도움을 받았어. 항상 뭔가에 막혀있을 때, 너와 함께 디스커션을 하고나면 그 막혔던 부분이 해결되는 적이 많았고, 그럴때마다 너의 비상한 머리에 감탄을 했지. 내가 따라갈 수 없는 너의 그 비상한 머리와 긍정적인 마음가짐이 부럽다. 졸업하고도 지금 계속 승승장구 하고 있는데, 앞으로도 그럴 것 이라고 믿어 의심치 않는다. 저의 롤모델 성철이형. 제가 형을 보고 대학원생활을 바람직하고 성실하게 하는 방법을 배운 것 같아요. 감사합니다. 그리고 나와 많은 시간을 함께 보냈던 동민이. 항상 뭐든지 꼼꼼히 열심히 하기 때문에 앞으로 좋은 일이 많이 있을 거라고 믿어. 내 동기 병주. 동기로서 그리고 형으로서 조금더 챙겨 줬어야 했는데, 연구분야도 다르고 그래서 뜻대로 잘 되지 않은것 같아. 마지막까지 잘 마무리하고 유종의 미를 거두기를 바랄게. 우리 연구실의 자타공인 수재 동하. 머리도 좋은데 성실하기 까지 한 너를 보면서 내가 많이 배운 것 같아. 내가 연구실로 꼬셔서 들어온 준수. 지금 랩장을 맡아서 바쁘겠지만, 지금처럼만 계속 꾸준히 연구를 하면 성공할 거야. 항상 쾌활한 성보. 연구를 항상 즐겁게 하는 천생 연구자 경석이. 알고리즘 천재 준영이. 파견근무가서 고생하는 너희들을 보면서 많이 도와주고 싶었는데 그러지 못해서 미안한 마음이야. 항상 열심히 하는 현준이. 그리고 진균이, 명섭이, 재형이, 성제, 성구, 성현이, 세훈이 모두 내가 더 많이 신경 써 줬어야 했는데 그러지 못한 것 같아서 미안하네. 워낙 다들 열심히 하니까 모두들 좋은 성과를 낼 수 있을거라고 생각해. 지금은 졸업했지만 제가 처음 들어왔을 때 직속 선배였던 동은이형과 유진이, 동기 진하, 그리고 내가 좋아하고 의지하던 형 지환이형. 더 오랜시간 같이 연구실 생활을 했으면 좋았을텐데, 그래도 졸업하고 다들 즐겁게 지내는 것 같아 보기 좋네요. 네이버에서 간혹 뵈었는데 회사생활 즐겁게 하고 계시는 태훈이형, 승걸이형. 지금은 교수님이 된 예진이. 비록 연구실에서는 겹치는 기간이 없었지만, 네이버에 가서 친해졌던 영철이형. 항상 잘 챙겨주셔서 감사합니다.

주변에 항상 능력있는 분들이 계셔서 많은 도움을 받을 수 있었던 것 같아서, 저는 정말 과분하게 운이 좋은 사람이라고 생각합니다. 앞으로도 계속 연락을 주고 받고 지낼 수 있길 바랍니다.

멀리있어서 자주 보지 못했지만, 항상 연락을 주고받으며 내 인생의 버팀목이었던 내 친

구들. 너희들이 있어서 내가 길었던 대학원 생활을 잘 마무리 할 수 있었던 것 같아. 그리고 대학교때부터 오랜시간 항상 내 곁을 지켜줬던 세정이. 언제나 힘이 되어줘서 정말 고마워. 앞으로 함께 할 날들이 많으니 많은 추억 만들고 행복하게 살자.

마지막으로, 무엇보다 제가 무얼 하든 항상 저를 믿어주시고 모든 면에서 지원을 아끼지 않으신 부모님께 감사드립니다. 그리고 사랑합니다. 제가 지치고 힘들 때마다 조언과 격려를 해주신 덕분에 제가 무사히 박사학위를 받을 수 있게 된 것 같습니다. 오랜 시간동안 공부한다고 집안에 보탬이 되지 못하고, 아들로써의 역할을 다하지 못했었는데, 앞으로 그 이상으로 보답하도록 하겠습니다. 믿음과 사랑을 주신만큼 평생 부모님께 효도하고 살아가겠습니다. 그리고 항상 저를 아껴주시고 저를 위해 기도를 해주시는 할머니 할아버지. 진심으로 감사드립니다. 그리고 내 동생 윤경이. 오랫동안 떨어져 지낸다고 오빠로서 의 역할을 많이 하지 못한 것 같아서 미안하게 생각해. 앞으로 서로 더 아껴주고 부모님께도 효도하자.

Curriculum Vitae

Name : Chanyoung Park

Education

2008. 3. – 2014. 2. Department of Computer Science and Engineering, Sogang University (B.S.)

2014. 3. – 2019. 2. Department of Computer Science and Engineering, Pohang University of Science and Technology (Ph.D)

Experience

2017. 3. – 2017. 6. Research Intern. NAVER, South Korea

2017. 9. – 2017. 12. Research Intern. Microsoft Research, Beijing, China (Mentor: Xing Xie)

Publication

1. **Chanyoung Park**, Donghyun Kim, Xing Xie, Hwanjo Yu "Collaborative Translational Metric Learning", The IEEE International Conference on Data Mining **(ICDM 2018)**
2. Dongmin Hyun, **Chanyoung Park**, Min-Chul Yang, Ilhyeon Song, Jung-Tae Lee and Hwanjo Yu "Review Sentiment-Guided Scalable Deep Recommender System", ACM SIGIR Conference on Research and Development in Information Retrieval **(SIGIR 2018 Short)**
3. Donghyun Kim, **Chanyoung Park**, Jinoh Oh, Hwanjo Yu, "Deep Hybrid Recommender Systems via Exploiting Document Context and Statistics of Items", in **Information Sciences** (2017) (SCI) (IF. 4.832)
4. **Chanyoung Park**, Donghyun Kim, Jinoh Oh, Hwanjo Yu, "Do "Also-Viewed" Products Help User Rating Prediction?", in International World-Wide Web Conference (**WWW 2017**)

5. **Chanyoung Park**, Donghyun Kim, Jinoh Oh, Hwanjo Yu, "Improving top-K recommendation with truster and trustee relationship in user trust network", in **Information Sciences** (2016) (SCI) (IF. 4.832)
6. Donghyun Kim, **Chanyoung Park**, Jinoh Oh, Hwanjo Yu, "Convolutional Matrix Factorization for Document Context-Aware Recommendation", in ACM international conference on Recommender System (**RecSys 2016**)
7. **Chanyoung Park**, Donghyun Kim, Jinoh Oh, Hwanjo Yu, "TRecSo: Enhancing Top-k Recommendation With Social Information", in International World-Wide Web Conference (**WWW 2016 Poster**)
8. **Chanyoung Park**, Donghyun Kim, Jinoh Oh, Hwanjo Yu, "Predicting User Purchase in E-commerce by Comprehensive Feature Engineering and Decision Boundary Focused Under-Sampling", ACM international conference on Recommender System Challenge (**ACM RecSysChallenge 2015**, 10th place, top 1.1%)

Awards

- NAVER Ph.D Fellowship (2016)
- ACM RecSys Challenge 2015 (10th/850 teams, top 1.1%)


POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY
1986
포항공과대학교