Cross-Platform App Recommendation by Jointly Modeling Ratings and Texts

DA CAO, Xiamen University
XIANGNAN HE, National University of Singapore
LIQIANG NIE, Shandong University
XIAOCHI WEI, Beijing Institute of Technology
XIA HU, Texas A&M University
SHUNXIANG WU, Xiamen University
TAT-SENG CHUA, National University of Singapore

Over the last decade, the renaissance of Web technologies has transformed the online world into an application (App) driven society. While the abundant Apps have provided great convenience, their sheer number also leads to severe information overload, making it difficult for users to identify desired Apps. To alleviate the information overloading issue, recommender systems have been proposed and deployed for the App domain. However, existing work on App recommendation has largely focused on one single platform (*e.g.,* smartphones), while ignores the rich data of other relevant platforms (*e.g.,* tablets and computers).

In this paper, we tackle the problem of cross-platform App recommendation, aiming at leveraging users' and Apps' data on multiple platforms to enhance the recommendation accuracy. The key advantage of our proposal is that by leveraging multi-platform data, the perpetual issues in personalized recommender systems — data sparsity and cold-start — can be largely alleviated. To this end, we propose a hybrid solution, STAR (short for "croSs-plaTform App Recommendation"), that integrates both numerical ratings and textual content from multiple platforms. In STAR, we innovatively represent an App as an aggregation of *common features* across platforms (*e.g.,* App's functionalities) and *specific features* that are dependent on the resided platform. In light of this, STAR can discriminate a user's preference on an App by separating the user's interest into two parts (either in the App's inherent factors or platform-aware features). To evaluate our proposal, we construct two real-world datasets which are crawled from the App stores of iPhone, iPad and iMac. Through extensive experiments, we show that our STAR method consistently outperforms highly competitive recommendation methods, justifying the rationality of our cross-platform App recommendation proposal and the effectiveness of our solution.

CCS Concepts:•**Information systems** → **Recommender systems;**

Additional Key Words and Phrases: App recommendation, Cross-platform, Hybrid system, Cold-start

ACM Reference Format:
Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, Tat-Seng Chua, 2016. Cross-Platform App Recommendation by Jointly Modeling Ratings and Texts. *ACM Trans. Inf. Syst.* x, x, Article x

This work was finished when Da Cao was a visiting student in the National University of Singapore. The first author claims that this work is under the supervision of Dr. Xiangnan He and Dr. Liqiang Nie. This work is supported by the NExT research centre, which is supported by the National Research Foundation, Prime Ministers Office, Singapore under its IRC@SG Funding Initiative. This work is also supported by the National Natural Science Foundation of China under Grant No. 61673327.

Author's addresses: Da Cao, Xiamen University; email: caoda0721@gmail.com; Xiangnan He, National University of Singapore; email: xiangnan@comp.nus.edu.sg; Liqiang Nie, Shandong University; email: nieliqiang@gmail.com; Xiaochi Wei, Beijing Institute of Technology; email: wxchi@bit.edu.cn; Xia Hu, Texas A&M University; email: hu@cse.tamu.edu; Shunxiang Wu (corresponding author), Xiamen University; email: sxwu@xmu.edu.cn; Tat-Seng Chua, National University of Singapore; email: dcscts@nus.edu.sg.

DOI: 0000001.0000001

(November 2016), 27 pages.
DOI: 0000001.0000001

1. INTRODUCTION

With the flourishing of network technologies, people now can easily access Internet through a variety of ways, such as smartphones and tablets. Regardless of the platform, Apps play a central role in providing services for users. For example, people can chat with friends with *WhatsApp* on mobile phones, watch videos of interest with *YouTube* on tablets, and handle some business stuffs with *Microsoft Office* on PCs. Although the wide variety of Apps have offered great convenience to people's lives, they have also made users difficult to identify the most desirable Apps. In other words, users are overwhelmed by the vast number of Apps — for one functionality, such as reading news, there can be over hundreds or even thousands of similar Apps available; even for one specific App, the different versions on different platforms (*e.g.*, smartphones, tablets, and computers) make the App identification process even complicated.

The recommender system is a well-known solution for addressing the information overloading problem and helping users identify relevant information on Web. It suggests items of interest to a user by mining the history of other similar users (collaborative filtering), or matching the user's profile with items' attributes (content filtering), or integrating both (hybrid filtering). Popularized by the Netflix challenge [Koren 2010], recommender systems have been intensively studied by the research community in recent years, and many recommendation techniques have been explored, such as the association rule-based [Parameswaran et al. 2011], graph-based [He et al. 2015], and latent factor-based models [Ge et al. 2014]. However, we argue that existing work is far from sufficient to address the App recommendation problem. This is because 1) most previous efforts have focused on item recommendation of general domains such as movies, books and products, rather than the specific domain of Apps that has quite different properties; 2) although a few recent work has targeted at the App domain [Lin et al. 2013; 2014a], they have primarily considered a single platform (mostly, Apps of the mobile stores). As a key and unique feature of App, we point out that the platform plays an important role in influencing a user's decision on items. For example, people tend to do some light entertainments on mobile phones (for the sake of convenience), such as listening to music and reading news, while may choose computers to perform some complicated jobs (for the sake of efficiency), such as making slides and planning activities. If a recommender system simply suggests Apps while ignoring their platform properties, it may end up with unsuitable Apps and adversely hurt users' experience. To provide a quality App recommendation service for users, it is inevitable to take the factor of platform into account.

In this work, we pay special attention to the platform-aware App modeling, focusing on the problem of cross-platform recommendation. Our key consideration is that, by accounting for users' cross-platform behaviours as well as items' platform-specific properties, the accuracy of App recommendation can be significantly improved. As illustrated in Fig. 1, in contrast to traditional App recommender systems that treat the data of each platform independently, our proposed cross-platform recommender system jointly models the App data of all platforms as a whole, enhancing the accuracy by explicitly modeling an App's common features (across all platforms) and specific features (of one single platform). Specifically, our method is designed to address the following four key challenges in App recommendation:

(1) **Platform Variance.** A platform corresponds to the specific hardware device that users use to operate Apps. Considering that different devices exhibit varying

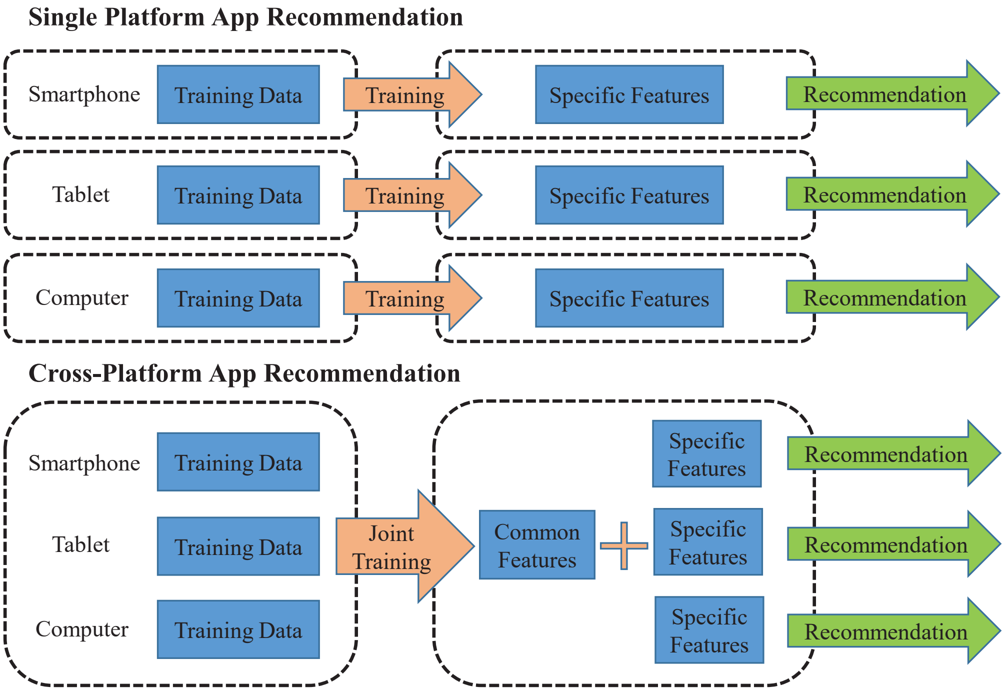


Fig. 1. Illustration of two App recommendation solutions: 1) traditional single platform-based recommendation that treats the data of each platform independently, and 2) our proposed cross-platform recommendation that considers the correlation of data on different platforms.

properties (*e.g.*, computers have a larger screen but are less portable compared with smartphones and tablets), App developers usually customize an App for a platform. This leads to significant differences of Apps' features on different platforms. As such, this kind of platform variance calls for a method that can capture both general characteristics and platform-specific features of Apps to provide quality recommendations for users.

(2) **Data Heterogeneity.** Apps are usually downloaded from online App markets, the data on which exhibits dual-heterogeneities, consisting of numerical ratings and textual content. Numerical ratings explicitly quantify users' satisfaction on an App, which are the most widely used source for recommender systems. While the textual content is much richer — including users' reviews on Apps, developers' descriptions of Apps, among others — which is complementary to users' ratings. More importantly, the textual content helps to uncover the specificities of Apps and platforms. How to effectively leverage the valuable information in the heterogeneous data and seamlessly sew them up is a highly challenging problem in recommender systems.

(3) **Data Sparsity.** Data sparsity is a well-known issue faced by personalized recommender systems. It is evidenced by that in the user–item rating matrix, only a small proportion of entries are known (usually less than 1%). We find the uneven popularity distribution of an App on different platforms makes the data sparsity issue even complicated. For example, the game *Angry Birds* is a free-download App on phones and tablets, but it charges for the PC version. As a result, seldom people play *Angry Birds* on PCs, resulting in few ratings of the game on the PC

platform. But considering that an App may have more ratings on other platforms, how to integrate data from multiple platforms to alleviate the data sparsity issue is important to enhance the recommendation accuracy.

(4) **Cold-Start Problem.** The extreme case of data sparsity is cold-start, where we need to provide recommendation for new users or items that have no historical ratings. As a perpetual issue in personalized recommender systems, cold-start is usually approached by utilizing some content-based side information [Schein et al. 2002; Zhou et al. 2011; Lin et al. 2013]. Here, we argue that our proposal of cross-platform recommendation can help alleviate the cold-start problem of single-platform to a certain extent; particularly for two scenarios: 1) a user is new to a platform (user cold-start) but has rating history on other platforms; 2) an App is first released on a platform (item cold-start) but has counterparts on other platforms. For these two cold-start cases, the key challenge is how to transfer the knowledge learnt from the non-cold-start platforms to effectively enrich users' and Apps' representation of cold-start platforms.

To address the aforementioned challenges, we present a hybrid filtering system STAR (short for "croSs-plaTform App Recommendation") that models both numerical ratings and textual content from multiple platforms for enhanced recommendation. The core component of our STAR method is a matrix factorization-based latent factor model, which is extended to support: 1) factorizing rating matrices from multiple platforms, and 2) incorporating textual content which is abstracted from topic models. We use item latent factors (learnt from rating matrices) to encode an App's common features across platforms, and use item latent topics (learnt from textual content) to differentiate an App's specific features of each platform. The aggregation of an App's common features and specific features is finally served as the representation of the App. In light of this, the data sparsity and cold-start problems of one single platform can be mitigated to a large extent.

To demonstrate the effectiveness of our solution, we conduct extensive experiments on our crawled real-world App data. First, we compare the overall prediction performance with the standard latent factor model [Koren 2008], semantics enhanced method [Ling et al. 2014], context-aware method [Rendle et al. 2011], and cross-domain method [Singh and Gordon 2008]. Our results and statistical significance test show that STAR significantly outperforms state-of-the-art methods. Second, we investigate the new-user and new-App cold-start scenarios, admitting that STAR is superior to the previous solutions. At last, to show the rationality of our design, we further dissect our method and perform some fine-grained analyses.

To summarize, this paper contributes on the following three aspects:

- We explore a new problem of cross-platform App recommendation. To the best of our knowledge, this is the first work that attempts to solve this problem in the domain of App recommendation.
- We develop a novel solution, STAR, to improve the App recommendation by jointly modeling of numerical ratings and textual content. Such a hybrid system is well-suited for resolving the data sparsity and cold-start issues in personalized recommender systems.
- We construct new datasets for studying the cross-domain App recommendation problem. To facilitate the research community and encourage future research on the emerging topic, we have released the datasets and our implementations[1].

[1] http://apprec.wixsite.com/star

The paper is structured as follows. After discussing related work in Section 2, we elaborate our solution STAR that combines cross-platform matrix factorization and platform-aware App modeling in Section 3. Before diving into the experimental evaluation in Section 5, we describe the construction of the datasets in Section 4. At last, we conclude the full paper in Section 6.

2. RELATED WORK

In this section, we provide a comprehensive review of the relevant literatures. We group the related work into five categories. First, we briefly review the collaborative filtering technique, which models ratings only and is the most widely used method in recommendation systems. Then, we discuss some work that attempts to combine ratings and textual content, which we refer as “semantics enhanced recommendation”. The third category covers context-aware recommender systems which model additional information that can affect users’ rating behaviours (*e.g.*, location, time, weather, and companion). Context-aware methods are relevant with our proposal, since the platform information can also be seen as a “context” in the App recommendation. Furthermore, we review work of cross-domain recommendation, which leverage the data of auxiliary domains to improve the recommendation of the target domain. At last, we discuss existing work on mobile App recommendation to position our work among them.

2.1. Collaborative Filtering

In terms of collaborative filtering approaches, work can be classified into two categories, namely, neighbourhood methods [Breese et al. 1998; Herlocker et al. 1999; Sarwar et al. 2001] and latent factor methods [Hofmann and Puzicha 1999; Hofmann 2004; Koren and Bell 2011]. Latent factor methods focus on fitting the user-item rating matrix using low-rank approximations and applying the matrix to identify new user-item associations, which have be proven to be superior to neighbourhood methods [Takács et al. 2008]. As one of the most representative realization of latent factor models, matrix factorization (MF) [Koren et al. 2009] characterizes both items and users by latent vectors inferred from observed ratings. The realization of MF can be generalized as a probabilistic model, known as probabilistic matrix factorization (PMF) [Mnih and Salakhutdinov 2007]. An effective way to solve the matrix approximations is to minimize the sum of squared errors, which can be tackled using singular value decomposition (SVD) [Hu et al. 2008]. The SVD++ [Koren 2008] model offers an improved accuracy by accounting for the implicit information recording which items were rated (regardless of their rating values). Collaborative filtering with implicit feedback is usually formulated as an item recommendation task, for which accounting for missing entries is crucial to the performance. The weight matrix factorization (WMF) [Hu et al. 2008] follows a regression framework to optimize a squared loss, carefully weighting the observed entries and missing data with different weights. Another well-known solution is Baysian Personalized Ranking (BPR) [Rendle et al. 2009], which optimizes models with a ranking-aware pairwise loss. The idea is that an observed entry should be assigned a higher score than a missing entry. However, collaborative filtering methods only model numerical ratings and forgo auxiliary information, such as texts and contexts. In this work, we supercharge collaborative filtering with user reviews, item descriptions and cross-domain information to improve App recommendation performance.

2.2. Semantics Enhanced Recommendation

Semantics enhanced methods attempt to improve recommendation quality by using ratings and textual content simultaneously. For Apps, the textual content can be App descriptions, version descriptions, and reviews. As representative text processing techniques, topic models, such as probabilistic latent semantic indexing (PLSI) [Hofmann 1999], probability latent semantic analysis (PLSA) [Hofmann 2001], latent Dirichlet allocation (LDA) [Blei et al. 2003], and non-negative matrix factorization (NMF) [Lin 2007], have been widely-accepted to interpret low-dimensional representations of documents [Cai et al. 2008; Chang et al. 2009]. The approach of collaborative topic modeling (CTR) [Wang and Blei 2011] integrates the merits of traditional collaborative filtering with probabilistic topic modeling to recommend scientific articles. The method of hidden factors as topics (HFT) [McAuley and Leskovec 2013] combines ratings with review texts for product recommendations, which works by aligning hidden factors in product ratings with hidden topics in product reviews. The model of TopicMF [Bao et al. 2014] recommends products by jointly considering user ratings and unstructured reviews. As an improvement of HFT, the technique of RMR (ratings meet reviews) [Ling et al. 2014] combines content-based filtering with collaborative filtering seamlessly, which exploits the information in both ratings and reviews. Most previous efforts [Nie et al. 2014; He et al. 2014a; He et al. 2015] enhance recommendation performance by considering semantic information in review texts or item descriptions. However, using textual content to distinguish the differences among items is rarely considered. In this paper, to model the differences among Apps on different platforms, we utilized a topic model to learn semantic information of reviews and App descriptions on different platforms, which contain rich information of platform-aware ratings.

2.3. Context-Aware Recommender Systems

Context information has been proven to be useful for providing recommendations [Palmisano et al. 2008], and relevant context-aware recommendation approaches have been proposed. According to the survey of [Adomavicius and Tuzhilin 2011], context-aware recommendation methods can be classified as pre-filtering [Adomavicius et al. 2005], where context drives data selection; post-filtering [Panniello et al. 2009], where context is used to filter recommendations after traditional method; and context modeling, where context is integrated directly into the model. However, the work on pre-filtering and post-filtering require supervision and fine-tuning in all steps of recommendation. Recent context modeling methods use all the context and user-item rating matrix simultaneously and build models based on matrix factorization methods. Multiverse recommendation [Karatzoglou et al. 2010] presents a collaborative filtering method based on tensor factorization that allows for a flexible and generic integration of contextual information by modeling the data as a user-item-context N-dimensional tensor instead of the traditional 2D user-item matrix. Factorization machines (FM) [Rendle et al. 2011] are applied to model contextual information and to provide context-aware predictions. Contextual operating tensor (COT) [Liu et al. 2015] is proposed, which represents the common semantic effects of contexts as a contextual operating tensor and represents a context as a latent vector. More recently, [Hsieh et al. 2016] has presented an Immersive Recommendation system, which incorporates users' digital traces from different contexts into recommendations. However, using external textual content to model the influence of context has been seldom considered. In this paper, we introduced the textual content of reviews and App descriptions to capture the influence of platform in App recommendation, which is different from traditional context-aware recommender systems.

2.4. Cross-Domain Recommender Systems

According to the survey of [Fernández-Tobías et al. 2012], cross-domain recommender systems can be categorized into two categories. One of them is adaptive models which exploit information from a source domain to make recommendations in a target domain. The work of [Zhang et al. 2011; Hu et al. 2013; Jamali and Lakshmanan 2013; Li and Lin 2014; Jiang et al. 2015] focus on user attributes, domain-specific factors, context-dependent entity type, matching of users and items, and social networks, respectively. The other category is collective models which are built with data from several domains and potentially can make joint recommendations for such domains. Collective matrix factorization (CMF) [Singh and Gordon 2008] factorizes several matrices and shares parameters among factors simultaneously when an entity participates in multiple relations. And it has been applied in attribute-aware relation prediction [Lippert et al. 2008] and social rating networks [Yang et al. 2011]. Collective models are suitable for our App recommendation task, since the recommendation performance of all platforms should be improved in our work. However, we observe that both adaptive models and collective models are either content-based or collaborative filtering-based, and hybrid cross-domain recommendation approaches have been barely investigated. In our task, we used numerical ratings and textual content simultaneously to capture common features and distinguish specific features of Apps on all platforms.

2.5. Mobile App Recommendation

In order to deal with the increasing number of Apps, some work on mobile App recommendation are emerging. Some of these work focus on constructing context-aware recommender systems on the platform of mobile device, which contains rich context-ware information (*e.g.*, location, social network, and activity) [Zheng et al. 2010; Karatzoglou et al. 2012; Zhu et al. 2012; Liu et al. 2013]. Personalized recommendation on the platform of mobile device has attracted a lot of attentions [Liu et al. 2011; Costa-Montenegro et al. 2012; Böhmer et al. 2013; Lin et al. 2014b]. The similarity of mobile Apps has been investigated by using graph [bhandari2013serendipitous] or kernel function [Chen et al. 2015; 2016]. At the same time, some specific features of mobile Apps are utilized to improve mobile App recommendation results. Ranking [Yankov et al. 2013] and popularity [Zhu et al. 2015] are well studied, since they are useful for understanding user experiences and learning the process of adoption of mobile Apps. The troublesome problems of data sparsity [Shi and Ali 2012] and cold-start [Lin et al. 2013] could be relieved or solved by using some specific features of mobile Apps (*e.g.*, neighbourhood among Apps and official twitter accounts). Some in-depth studies on privacy and security awareness [Zhu et al. 2014; Liu et al. 2015] have been given to avoid privacy invasion and other security concern. In the work of [Yin et al. 2013], App recommendation can be regarded as a result of the contest between satisfaction and temptation. Different from traditional items (*e.g.*, books, movies, and music) in recommender systems, Apps change and evolve, which is reflected by an increment in their version numbers. Relevant work [Lin et al. 2014a] has be presented, which incorporates features distilled from version descriptions into app recommendation. To achieve both memorization and generalization for recommender systems, a Wide & Deep learning framework [Cheng et al. 2016] has been proposed, which jointly trains a linear model and a nonlinear neural network model. It is reported that the method has been deployed to production for Google Play. However, previous work focus on mobile App recommendation and few of them try to make use of the data from other platforms (*e.g.*, tablet and computer). In this paper, we captured App features on different platforms and fused the data on different platforms together to improve the recommendation results on all platforms.

3. CROSS-PLATFORM APP RECOMMENDATION

In this section, we present our proposed STAR method. We begin by introducing the problem formulation, followed by elaborating the two core components of our design

— the cross-platform matrix factorization model (CPM) and the platform-aware App modeling with texts (PAT). At last, we give the inference algorithm of STAR and discuss the new-user and new-App cold-start issues.

3.1. Problem Formulation

Let I, J, and S denote the number of users, Apps, and platforms, respectively. Each input rating instance is then represented as a quadruple (i, j, s, r_{ijs}), meaning that user i has rated App j on the s^{th} platform with the score r_{ijs}. Mathematically, we can represent the data as a three-order tensor $R \in \mathbb{R}^{I\times J\times S}$, where the rating matrix of the platform s corresponds to the slice $R_{\cdot,\cdot,s}$. It is worth pointing out that it is common that an App has no ratings in a platform (new-item cold-start) and a user has no ratings in a platform (new-user cold-start), which yields certain slices of the tensor are just a zero matrix. This poses challenges to the traditional tensor factorization methods like Tucker Decomposition and High-Order SVD [Rendle 2011].

In addition, each App is accompanied by some textual content such as the developer's description and users' reviews, and we use d_{js} to denote the accompanying text of the App j on the s^{th} platform. Let the set of observed rating instances (*i.e.,* the non-zero entries of tensor R) be $\mathcal{R}$, which is an incomplete set of tensor entries as most ratings are unknown (*i.e.,* $|\mathcal{R}| \ll IJS$). The cross-platform recommendation problem is then formulated as predicting the unknown ratings in the tensor R, where the predicted scores of unknown ratings can be used to rank items for a user.

3.2. Cross-Platform Matrix Factorization Model (CPM)

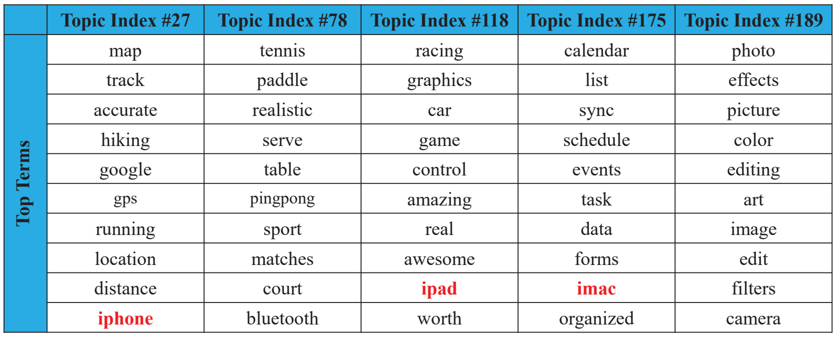
As a latent factor model, matrix factorization (MF) maps both users and items to a joint K-dimensional latent space, such that user-item interactions are estimated as the inner products in that space. While the standard MF model [Koren 2010] is designed for modeling user–item ratings of a single (homogeneous) domain, it is a suboptimal choice for modeling ratings of multiple (correlated) domains, since a direct application of the MF model will treat the data of different domains independently. Inspired by the generalized collective matrix factorization method [Singh and Gordon 2008], we apply the similar idea that models a user's cross-domain behaviours with a shared set of latent parameters. Mathematically, we model each rating entry as:

$$\hat{r}_{ijs} = \mu + \mathbf{b}_u(i) + \mathbf{b}_v(js) + \mathbf{u}_i^T \mathbf{v}_{js}, \tag{1}$$

where μ is the global bias, which can be set as the average score of all ratings; $\mathbf{b}_u(i)$ denotes the bias of the user i, and $\mathbf{b}_v(js)$ denotes the bias of the App j on the s^{th} platform. Latent vectors $\mathbf{u}_i$ and $\mathbf{v}_{js}$ are the key parameters of the CPM model, denoting the representation of the user i and App j on the s^{th} platform, respectively.

Note that we have purposefully designed our model to share a user's representation across platforms while differentiate an App's representation of different platforms. The consideration is that a user's interest in consuming Apps should remain largely unchanged for different platforms, while an App's properties can be changed dramatically since developers usually design the App to adapt the platform's properties. The design of shared user representations leads to the side effect of alleviating the new-user cold-start issue — even if a user has no ratings in one platform, his/her preference can be learnt from the data of other platforms. As we will show later in the experiments, although simple, this shared representation of a user is an effective way to model the user's cross-platform behaviours and boost the prediction's accuracy significantly.

Nevertheless, the platform-specific modeling of Apps in Eqn. (1) assumes that the representations of an App on different platforms are independent with each other. This is counterintuitive, since the same App developed by a company usually shares



	Topic Index #27	Topic Index #78	Topic Index #118	Topic Index #175	Topic Index #189
Top Terms	map	tennis	racing	calendar	photo
	track	paddle	graphics	list	effects
	accurate	realistic	car	sync	picture
	hiking	serve	game	schedule	color
	google	table	control	events	editing
	gps	pingpong	amazing	task	art
	running	sport	real	data	image
	location	matches	awesome	forms	edit
	distance	court	**ipad**	**imac**	filters
	iphone	bluetooth	worth	organized	camera

Fig. 2. An example of five topics extracted by LDA, where each topic is represented by the top 10 words ranked by their probabilities. The highlighted terms (in bold) reveal the platform information of the topic.

the basic functionalities, regardless of the released platform (*e.g.*, *YouTube* is mainly for watching videos, while *WhatsApp* is mainly for people's communication). One way to get around the independence is to enforce an App's representations on different platforms to be similar with each other, such as via the pair-wise regularizer [He et al. 2014b]. However, it is non-trivial to control the level of similarity for different Apps and platforms. In what follows, we approach the problem by additionally modeling the surrounding texts for platform-aware App representation learning.

3.3. Platform-Aware App Modeling with Texts (PAT)

As we have mentioned in the introduction section, most App platforms have rich textual information available. The two most universal types are the reviews (written by users) and App description (written by developers). We find that both user reviews and App description are well-suited to complement with the ratings for recommendation — reviews justify the ratings by discussing the App's properties from the perspective of users, while the App description describes the App's functionalities from the perspective of developers. By properly modeling the rich evidence sources in the textual content, we believe a better representation for an App can be learnt, especially in differentiating its common functionality and platform-specific properties.

Nevertheless, the inherent noises in user-generated reviews and the variability of natural language pose great challenges for understanding the semantics in texts. One viable solution is first extracting the aspects (*i.e.,* nouns and noun phrases that describe Apps' properties) from the texts, and then feeding into the recommendation model [He et al. 2015]. However, this way requires a quality aspect extraction tool; while the accuracy of aspect extraction tools is usually domain dependent, to our knowledge, there does not exist such a tool specially developed for the App domain. Another solution is projecting the text into the latent topic space, where each topic can be explained by a few top words [McAuley and Leskovec 2013]. Fig. 2 shows the top words of five selected topics, which are learnt by LDA on our dataset. As can be seen, the learnt topics are rather coherent and explainable — they not only reflect the properties of Apps, but also reveal some information relevant to the platform. As such, to lessen the dependency on the domain knowledge for identifying aspects, we resort to the topic modeling approach to model the texts. Specifically, we apply LDA on the textual content, obtaining a topic distribution $\boldsymbol{\theta}_{js}$ as the semantic abstraction for the App's text d_{js}.

To capture our intuition that the properties of an App are composed of its functionalities (shared across platforms) and platform-aware specificities, we separate the

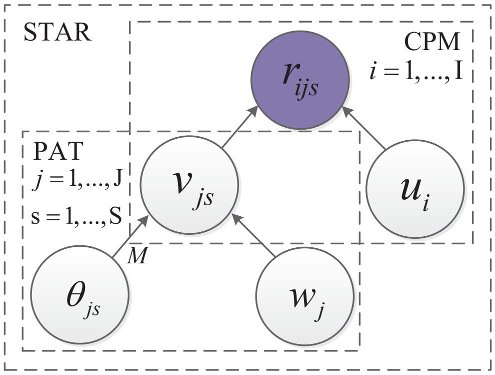


Fig. 3. Graphical representation for the STAR model.

modeling for an App into two parts:

$$\mathbf{v}_{js} = \mathbf{w}_j + \mathbf{M}\boldsymbol{\theta}_{js}, \tag{2}$$

where $\mathbf{w}_j$ is the latent vector for the App j that encodes its shared functionalities, and M is the projection matrix that projects the non-negative topic space into the latent factor space, and we use the term $\mathrm{M}\boldsymbol{\theta}_{js}$ to encode the App properties that are dependent with the platform. Let the number of latent factors be K and the number of topics be T, then the matrix M is of dimension $K \times T$. It is worth pointing out that the linear transformation that bridges the discrepancy between the topic space and latent factor space is very effective to translate the semantics in texts into the vectorized representation of Apps, and yields significant improvement over the CPM model. While other choices of non-linear transformations do exist, such as the exponent function used by [McAuley and Leskovec 2013] and neural network-based functions [Dziugaite and Roy 2015], we leave this exploration for future work.

3.4. The STAR Recommendation Method and Learning Algorithm

Our final STAR recommendation model combines the designs of CPM that models users' cross-platform ratings and PAT that enhances the App representation with textual data, illustrated with the graph representation in Fig. 3. Note that we have intentionally ignored the bias terms in the figure to better highlight our key designs. A rating is generated by the interaction between the user's and the App's representation on the target platform, which is further decomposed into the App's common features and platform-specific features. Formally, we give the prediction model as follows:

$$\hat{r}_{ijs} = \mu + \mathbf{b}_u(i) + \mathbf{b}_w(j) + \mathbf{u}_i^T(\mathbf{w}_j + \mathbf{M}\boldsymbol{\theta}_{js}), \tag{3}$$

where μ, $\mathbf{b}_u(i)$, $\mathbf{u}_i$, $\mathbf{w}_j$, M and $\boldsymbol{\theta}_{js}$ have the same meanings as presented in Eqn. (1) and Eqn. (2); $\mathbf{b}_v(js)$ is replaced by $\mathbf{b}_w(j)$, which denotes the bias of the j-th App.

To infer model parameters for a personalized recommendation model, we need to design an objective function to optimize. There are two forms of objective functions that have been widely adopted in the literature: 1) point-wise loss [Koren 2008] that performs regression on known ratings, and 2) pair-wise loss [Rendle et al. 2009] that maximizes the margin between the known and unknown ratings. While the second choice is mainly designed for learning from implicit feedback (where users' explicit preference on items are unknown), we opt for the first choice which has been shown to be very effective in modeling users' explicit ratings and also yields strong

Top-K ranking performance [Koren 2008]. The regression-based objective function is formulated as follows:

$$\mathcal{L}(\Theta) = \frac{1}{2} \sum_{(i,j,s)\in\mathcal{R}} (r_{ijs} - \hat{r}_{ijs})^2 + \lambda||\Theta||^2, \tag{4}$$

where $\Theta = \{\mathbf{U}, \mathbf{W}, \mathbf{M}, \mathbf{b}_u, \mathbf{b}_w\}$, denoting all the five groups of parameters to learn, and $\mathcal{R}$ is the set of observed rating instances for training. To combat overfitting, we use the standard L_2 regularization of all the parameters, weighted by the hyperparameter λ (distinct for each group of parameters in our implementation, *i.e.,* $\lambda_u, \lambda_w, \lambda_m, \lambda_b$ for $\mathbf{U}, \mathbf{W}, \mathbf{M}, \mathbf{b}_u$ and $\mathbf{b}_w$, respectively).

To minimize the objective function, a standard solution is applying squared loss on model parameters [Koren et al. 2009]. Since the prediction model is in a linear form, another solution is coordinate descent [He et al. 2016], also dubbed as alternating least squares (ALS) for optimizing the squared loss. While the ALS solution is more difficult to obtain (requiring an exact optimization solution for each parameter in each update), we resort to the stochastic gradient descent (SGD) algorithm, which is much easier to derive. Specifically, it randomly samples a training instance, and then performs a gradient descent step for all related parameters regarding the loss of the training instance. Let $l(i, j, s)$ be the local loss for the instance (i, j, s), then we give the derivative with respect to each group of parameters as follows:

$$\begin{aligned}
\frac{\partial l(i,j,s)}{\partial \mathbf{u}_i} &= - e_{ijs}\left(\mathbf{w}_j + \mathbf{M}\boldsymbol{\theta}_{js}\right) + \lambda_u \mathbf{u}_i, \\
\frac{\partial l(i,j,s)}{\partial \mathbf{w}_j} &= - e_{ijs}\mathbf{u}_i + \lambda_w \mathbf{w}_j, \\
\frac{\partial l(i,j,s)}{\partial \mathbf{M}} &= - e_{ijs}\mathbf{u}_i \boldsymbol{\theta}_{js}^T + \lambda_m \mathbf{M}, \\
\frac{\partial l(i,j,s)}{\partial \mathbf{b}_u\left(i\right)} &= - e_{ijs} + \lambda_b \mathbf{b}_u\left(i\right), \\
\frac{\partial l(i,j,s)}{\partial \mathbf{b}_w\left(j\right)} &= - e_{ijs} + \lambda_b \mathbf{b}_w\left(j\right),
\end{aligned} \tag{5}$$

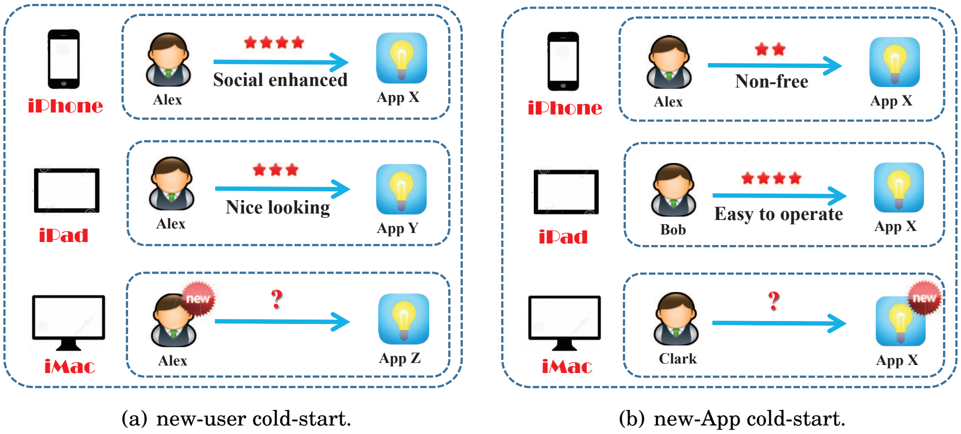
where e_{ijs} denotes the prediction deviation, computed as $e_{ijs} = r_{ijs} - \hat{r}_{ijs}$. In each gradient step, a parameter takes an update towards the negative gradient, which is rescaled by a step size (also termed as “learning rate”). As a constant learning rate may result in fluctuations in the later iterations (close to the local minimum), we employ an adaptive strategy for the learning rate. We monitor the training loss of each iteration; when the loss increases, we punish the learning rate by a ratio of 0.5. Through this way, the SGD algorithm can steadily converge to a local minimum.

3.5. New-User and New-App Cold-Start Problems

We concern ourselves with the cold-start scenario where a user or an App is new to a platform (*i.e.,* has no rating history), while the user or App does have historical data on other platforms[2]. Fig. 4 shows two illustrative examples of the cold-start scenarios.

Since the data from multiple platforms have been jointly modelled in STAR, the users (and Apps) of the cold-start platform are in the same latent space with the users (and Apps) of other non-cold-start platforms. This is a key advantage of our

[2] Note that as long as the user or App has historical data on at least one platform, our solution for alleviating cold-start will work. However, similar to other collaborative filtering methods, our method will also fail for the pure cold-start problems where a user or App does not have any rating history.



(a) new-user cold-start.

(b) new-App cold-start.

Fig. 4. Illustration of the new-user and new-App cold-start problems. The question mark "?" stands for the ratings that we wish to predict, and the label "new" means the user or App is new to the platform and has no rating history on it.

joint modeling approach and forms the basis for addressing the cold-start problems. Specifically, for the new-user cold-start situation, we can directly use the latent vector $\mathbf{u}_i$ learnt from other platforms as the user's representation for the cold-start platform. In the case of the new-App cold-start scenario, when the textual description is available, we can follow the same procedure (as in Eqn. (2)) to obtain the App's representation. When the textual information is absent, we can use the shared latent vector $\mathbf{w}_j$ as the App's representation on the cold-start platform. In this case, STAR will forgo incorporating the platform-aware features, and the prediction on the item will be solely based on its shared properties on other platforms.

4. DATA PREPARATION AND STATISTICS

In this section, we detail how we prepare data for empirical evaluation. In addition, we perform some basic statistical analysis to better understand the data.

4.1. Data Collection

The data sources used for evaluating our solution include both numerical ratings and textual content:

1. Numerical ratings and their affiliated textual reviews.. We first crawled all Apps' names and IDs from the iTunes Store[3] and Mac App Store[4] in November 2015. In total, we obtained $2,076,240$ iPhone and iPad Apps, and $49,298$ iMac Apps. We then retrieved the reviews for all Apps. A review entry contained a numerical rating (of scale 1–5), reviewing timestamp, title, and content.

2. Apps' textual descriptions.. For each App, we also crawled its textual description from the iTunes Store and Mac App Store. An App's description contained the App's title, textual description, genre, developer's ID, and version change logs.

4.2. Data Processing

4.2.1. Datasets construction. We constructed two datasets — one is of two platforms and the other is of three platforms — to evaluate our method. The first dataset is

[3] https://itunes.apple.com/us/genre/ios/id36?mt=8
[4] https://itunes.apple.com/us/genre/mac/id39?mt=12

Table I. Some statistics of the iPhone-iPad dataset.

	Amount	Min.#ratings	Max.#ratings	Avg.#ratings
User	$112,031$	2	219	2.86
App-iPhone	$2,704$	1	$15,946$	62.31
App-iPad	$2,704$	1	$11,846$	56.23

Table II. Some statistics of the iPhone-iPad-iMac dataset.

	Amount	Min.#ratings	Max.#ratings	Avg.#ratings
User	$121,905$	2	21	2.21
App-iPhone	201	1	$64,482$	$1,117.37$
App-iPad	201	1	$5,572$	206.62
App-iMac	201	1	439	13.97

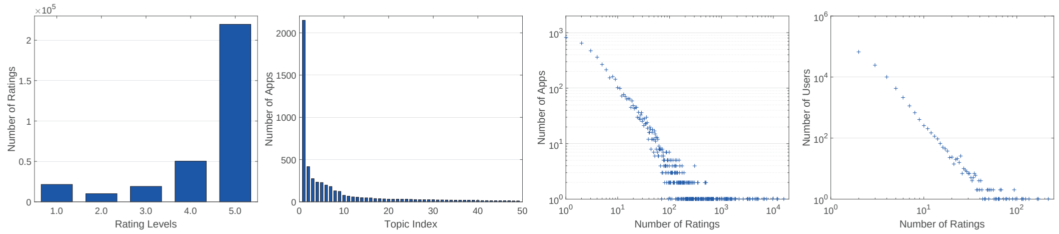
constructed based on the platforms of iPhone and iPad. The name of an App on iPad usually contains the word of “HD”, which represents high definition. So we first chose the Apps whose names contain “HD”, and then found their corresponding versions on iPhone. We found that $3,800$ pairs of Apps exist on both of these platforms. Since most users use the same Apple account to download Apps on both iPhone and iPad, we could identify the same users by matching user IDs on the two different platforms. We further processed the dataset by retaining users who rated at least once on both of these platforms. Ultimately, we obtained $112,024$ users, $2,704$ pairs of Apps, and $320,535$ ratings ($168,489$ ratings on iPhone, and $152,046$ ratings on iPad). The user-App ratings matrix has a sparsity of 99.95%. The time span of ratings ranges from September 13th, 2008 to October 24th, 2015. Detailed statistics of the dataset are provided in Table I.

The second dataset is constructed based on the platforms of iPhone, iPad, and iMac. Since the Apps in iTunes Store and Mac App Store share the same name, they can be easily linked. We used the same method mentioned in the first dataset to link the Apps on the platforms of iPhone and iPad. Finally, we obtained 260 triples of Apps that are existed on all the three platforms. We selected users who had at least two ratings in the 260 triples, and finally obtained $121,905$ users ($102,789$ users rated on one platform, $18,960$ users rated on two platforms, and 156 users rated on there platforms), 201 triples of Apps, and $268,929$ ratings ($224,591$ ratings on iPhone, $41,530$ ratings on iPad, and $2,808$ ratings on iMac). The user-App ratings matrix has a sparsity of 99.63%. The time span of ratings ranges from January 27th, 2008 to November 16th, 2015. Detailed statistics of the dataset are provided in Table II.

4.2.2. Topic modeling on textual content. We first integrated all reviews of an App and treated it as a document. We also treated the description of an App as a document. Before running the LDA method [Blei et al. 2003] to extract topics, we performed a modest filtering on texts to reduce possible noises. Specifically, we first removed the non-English words and stop words, and normalized verbs and adjectives with the help of the porter stemmer [Zhu et al. 2014]. To run LDA, we set the priors for topic–word and document–topic distributions as $50/T$ (where T denotes the number of topics) and 0.1, respectively, as suggested by [Heinrich 2008]. Fig. 2 shows an example of topics extracted by LDA from our datasets.

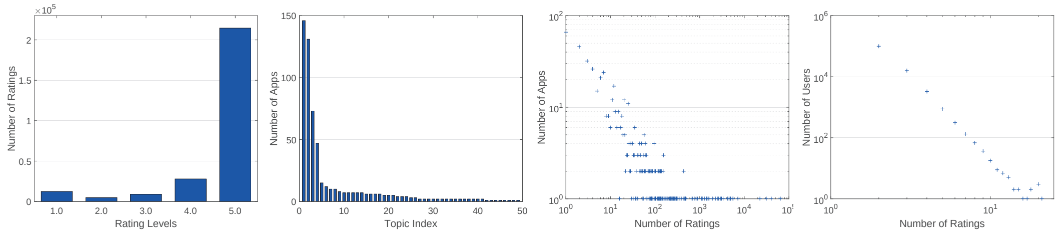
4.3. Statistical Analysis

We performed some statistical analysis to better understand the datasets. Since the statistics of the iPhone-iPad-iMac dataset show very similar trends with the iPhone-iPad dataset, we focus on discussing Fig. 5 for short. From Fig. 5(a), we can see the rating distribution is strongly biased towards the high-score ratings (over 80% of the ratings are higher than 3). It might be because that users tend to rate Apps they



(a) Rating level distribution (b) Topic distribution (c) App distribution (d) User distribution

Fig. 5. Data distributions of the iPhone-iPad dataset.



(a) Rating level distribution (b) Topic distribution (c) App distribution (d) User distribution

Fig. 6. Data distributions of the iPhone-iPad-iMac dataset.

like and avoid the ones they dislike. This similar *not missing at random* (NMAR) phenomenon has been previously observed in other domains, such as music [Marlin et al. 2007]. Fig. 5(b) shows the topic distribution over Apps. For each App, we select its prominent topics and show the distribution of the top 50 most popular topics in the figure. As can be seen, it exhibits a highly skewed distribution of topics, where the top 10 topics take over 80% of Apps. We hypothesize this may be caused by the focused categories of the Apps, where most Apps on our datasets are about games and entertainments. Lastly, we show the App distribution and user distribution with respect to the number of ratings in Fig. 5(c) and 5(d) (log–log plot), respectively. As we can see, both users and Apps show a long-tail distribution — most users and Apps only have very few ratings, and only a small proportion of users and Apps have many ratings. This is consistent with most recommendation benchmarks, such as the Netflix [Koren 2008] and Yelp [He et al. 2015] datasets, and highlights the sparsity challenge faced by personalized recommender systems.

5. EXPERIMENTS

In this section, we conducted extensive experiments on our collected datasets aiming to answer the following five research questions:

— **RQ1**. How does our designed STAR approach perform as compared with other state-of-the-art recommendation algorithms?
— **RQ2**. How does STAR perform in handling the new-user and new-App cold-start problems?
— **RQ3**. Do users exhibit distinct preferences for different platforms of an App? Is STAR able to target the exact platform of an App that the user has rated?
— **RQ4**. How do the common features and specific features of Apps contribute to the overall effectiveness of STAR?

— **RQ5**. In addition to rating prediction that is prevalent to a recommendation algorithm, how does STAR perform in the more practical top-N recommendation?

To empirically answer these questions, we carried out experiments on the iPhone-iPad and iPhone-iPad-iMac datasets, respectively. It is worth emphasizing that the experimental results regarding **RQ1**, **RQ4** and **RQ5** reported in this paper were based on five-fold cross-validation. For the rest two questions, due to their intrinsic requirements, it is hard to perform five-fold cross-validation. Instead, we repeated the experiments five times and reported their average results.

5.1. Experimental Settings

5.1.1. Evaluation Metrics. Rating prediction has become the *de facto* to evaluate an explicit feedback-based recommender systems. To measure the prediction performance, we adopted the commonly used error metric, mean absolute error (MAE), which is calculated as,

$$MAE = \frac{1}{|\mathcal{T}|} \sum_{(i,j,s)\in\mathcal{T}} \left|\hat{r}_{ijs} - r_{ijs}\right|, \tag{6}$$

where $|\mathcal{T}|$ is the number of ratings in the testing set $\mathcal{T}$, $\hat{r}_{ijs}$ is the predicted rating of the user i on the App j over the s^{th} platform, and r_{ijs} is the ground truth. Besides, we employed the root mean squared error (RMSE) as another error metric, defined as,

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(i,j,s)\in\mathcal{T}} \left(\hat{r}_{ijs} - r_{ijs}\right)^2}. \tag{7}$$

From the formal definition, we can see that the smaller MAE and RMSE values normally indicate better performance for rating predictions. MAE assigns equal weight to the data, whereas RMSE emphasizes the extremes.

On the other hand, the top-N recommendation task is of more practical value, as most online recommendation systems only show a short list of items to users. To evaluate the performance of top-N recommendation, we resorted to the widely used accuracy metric, recall, which is defined as,

$$Recall@M = \frac{n_u}{N_u}, \tag{8}$$

where n_u and N_u are the number of Apps the user likes in the top M ranked list and the total number of Apps the user likes, respectively. We calculate the recall for every user in the testing set, and report the average score. In addition, we utilized the normalized discounted cumulative gain (NDCG) as our another ranking-aware metric, which is based on discounted cumulative gain (DCG),

$$DCG@M = \sum_{i=1}^{M} \frac{2^{rel_i} - 1}{\log_2(i+1)}, \tag{9}$$

where rel_i is the graded relevance of the result at position i. IDCG is the ideal DCG for a given set of Apps, and then the NDCG is computed as,

$$NDCG@M = \frac{DCG@M}{IDCG@M}. \tag{10}$$

We can see that large values of Recall and NDCG indicate better performance for top-N recommendation. Recall measures the proportion of relevant items that the algorithm successfully identified in the top-N ranked list, while NDCG accounts for the position of correct hits, which assigns higher importance to results at top ranks, scoring successively lower ranks with marginal fractional utility.

5.1.2. Baseline Methods. Here we briefly introduce the baseline algorithms which we intend to compare with. The baseline algorithms were chosen from collaborative filtering method in Section 2.1, semantics enhanced recommendation approach in Section 2.2, context-aware recommender systems technique in Section 2.3, and cross-domain recommender systems method in Section 2.4, respectively.

— **SVD++** [Koren 2008]. As a representative MF method, SVD++ considers the implicit information (*e.g.*, clicks and purchase records), which has been proven to be superior to traditional MF methods. In our experiments, we regarded the historical ratings as implicit feedbacks and incorporated them in the SVD++ approach. We trained the model on different platforms separately. Hence, the same user on different platforms was considered as different users. SVD++ was used in the experiment of overall performance comparisons.

— **RMR** [Ling et al. 2014]. As a typical semantics enhanced technique, RMR has been proven to be superior to CTR and HFT. The document setting in RMR is the same as that in our STAR model, which integrates all user reviews of an App into one document. We trained the model on different platforms at the same time, and treated the same user on different platforms as one user. Therefore, RMR was used in the experiment of overall performance comparisons and new-user cold-start problem. Since RMR is based on HFT, we used the code released by the authors of HFT[5].

— **CTR** [Wang and Blei 2011]. Similar to RMR, CTR is also one of the semantics enhanced methods, and it is capable of handling out-of-matrix cold-start problem, which is the same as our new-App cold-start problem. We treated the same user on different platform as one user. Since RMR cannot handle new-App cold-start problem, we employed CTR as a semantics enhanced baseline in our experiment to handle this scenario.

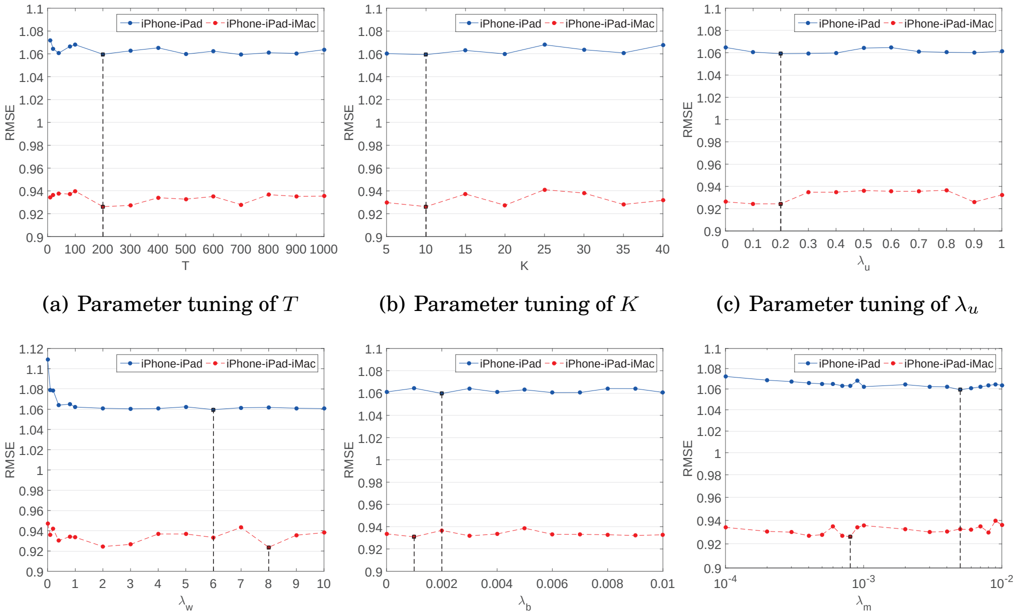
— **FM** [Rendle et al. 2011]. As a context-aware recommendation algorithm, FM outperforms other context-aware competitors in prediction quality and runtime. We utilized the released codes to implement this method[6]. In our experiment, the platform was considered as a context and the same user on different platforms was seen as one user. FM was used in the experiment of overall performance comparisons, new-user cold-start problem, and new-App cold-start problem.

— **CMF** [Singh and Gordon 2008]. CMF is one of the most reputable approaches in cross-domain recommender systems. It shares parameters among factors on different matrices simultaneously when a user or item participates in multiple relations. In our experiment, user participates in multiple platforms and can be associated by identifying the same user ID on different platforms. CMF was used in the experiment of overall performance comparisons, new-user cold-start problem, and new-App cold-start problem. It is worth noting that the App on different platforms was seen as the same App in the new-App cold-start problem.

— **WMF** [Hu et al. 2008]. WMF is a collaborative filtering method for implicit feedback datasets that is tailored for top-N recommendation. It transforms user's implicit feedback into two magnitudes — preference and confidence. Preference measures whether the user likes or dislikes the item. In our experiment, a rating of 3 and above on the 5-point Likert scale was translated to "like", and a rating of 2 and

[5] http://cseweb.ucsd.edu/~jmcauley/

[6] http://www.libfm.org/



(a) Parameter tuning of T (b) Parameter tuning of K (c) Parameter tuning of λ_u

(d) Parameter tuning of λ_w (e) Parameter tuning of λ_b (f) Parameter tuning of λ_m

Fig. 7. Parameter tuning in terms of RMSE over two datasets.

below was translated to "dislike", which are consistent with the experiment setting in [Lin et al. 2014a]. Confidence measures the level of users' preferences on the item, which is reflected by the numerical rating (the implementation is a monotone function with the rating). WMF regards all missing data as negative samples and utilizes alternative least square solution (ALS) to solve this model, which is time-consuming and memory-consuming for large-scale data. Thus, we resort to sample various numbers of negative samples for each user instead of using full missing data, and then obtain the optimal results to approximate the result of WMF. Before diving into the algorithm, we rescaled the absolute ratings using their average cumulative proportion [Hsieh et al. 2016] to alleviate the rating bias.

— **Popular**. Items are ranked by their popularity judged by the number of ratings. It is a non-personalized method that benchmarks the performance of other personalized methods . While it seems simple and heuristic, it can be a competitive baseline for the top-N task, as users tend to consume popular items [Cremonesi et al. 2010].

5.2. Parameter Tuning and Convergence Analysis

For each method mentioned in Section 5.1.2, the involved parameters were carefully tuned, and the procedures to tune the parameters are analogous to ensure fair comparison. In addition, the parameters with the best performance were used to report the final comparison results. Take the tuning procedure of our proposed STAR model as an example.

There are six important parameters in our STAR model: 1) the number of topics T in topic modeling; 2) the dimensionality of latent factors K in matrix factorization; and 3) the parameters λ_u, λ_w, λ_b, and λ_m that balance the terms in our proposed model. In particular, in one of the five fold, we selected the optimal parameters by grid search with a small but adaptive step size on the training set. Here we revealed

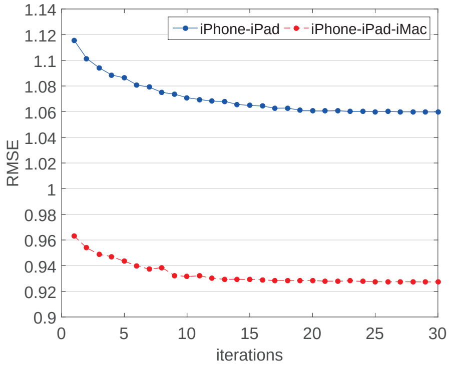


Fig. 8. Convergence analysis in terms of RMSE over two datasets.

the tuning results in the overall performance comparisons, and observed that when $T = 200$, $K = 10$, $\lambda_u = 0.2$, $\lambda_w = 6$, $\lambda_b = 0.002$, and $\lambda_m = 0.005$ for iPhone-iPad dataset, our model achieved the best performance regarding RMSE. Regarding iPhone-iPad-iMac dataset, the optimal parameter setting is $T = 200$, $K = 10$, $\lambda_u = 0.2$, $\lambda_w = 8$, $\lambda_b = 0.001$, and $\lambda_m = 0.0008$. We then investigated the sensitivity of our STAR to these parameters by varying one and fixing the others. The parameter tuning results for the iPhone-iPad-iMac dataset is almost the same as that of the iPhone-iPad dataset. To save the space, we only illustrated the tuning results of the iPhone-iPad dataset here.

We first fixed $K = 10$, $\lambda_u = 0.2$, $\lambda_w = 6$, $\lambda_b = 0.002$, $\lambda_m = 0.005$, and varied T. As shown in Fig. 7(a), the value of RMSE changes in a small range, when varying T from 100 to 1000, and reaches its minimum value when $T = 200$. The slight change demonstrates that our model is non-sensitive to the parameter T.

We then fixed $T = 200$, $\lambda_u = 0.2$, $\lambda_w = 6$, $\lambda_b = 0.002$, $\lambda_m = 0.005$, and varied K. As shown in Fig. 7(b), RMSE decreases first and then increases along the increasing of K. It reaches its minimum when $K = 10$. Our finding is different from traditional latent factor methods, which are inclined to use more factors [Koren et al. 2009]. This is mainly because latent factors in matrix factorization and topic distributions on textual content are mutually interrelated in our model. And users might only mention a few of all possible factors in textual content, which leads to limited factors are considered in matrix factorization.

Using the similar method to adjust other parameters, we can see from Fig. 7(c), Fig. 7(d), Fig. 7(e), and Fig. 7(f) that the value of RMSE changes in a small range, when varying λ_u, λ_w, λ_b, and λ_m in range of [0, 1], [1, 10], [0, 0.01], and [0.001, 0.01], respectively.

At last, we recorded the value of RMSE along with each iteration using the aforementioned optimal parameter setting. The initial value of learning rate was set to 0.01. If the value of loss function increases, we divided the learning rate by 2. After five times division, we finished the iteration process. Fig. 8 shows the convergence study with the increasing number of iterations. It shows that our algorithm can converge within 20 iterations.

5.3. Overall Performance Comparisons (RQ1)

To demonstrate the overall effectiveness of our proposed STAR model, as discussed in Section 5.1.2, we compared the STAR with several state-of-the-art recommendation approaches: 1) SVD++; 2) RMR; 3) FM; and 4) CMF.

Table III. Performance comparison of various methods on the iPhone-iPad dataset regarding RMSE and MAE.

Methods	MAE	RMSE	p-value (MAE)	p-value (RMSE)
SVD++	0.7823 ± 0.003	1.1218 ± 0.004	$2.44e$-10	$1.02e$-10
RMR	0.7679 ± 0.002	1.0716 ± 0.003	$5.83e$-09	$7.16e$-08
FM	0.8002 ± 0.003	1.0887 ± 0.003	$3.05e$-11	$2.06e$-10
CMF	0.7701 ± 0.003	1.0977 ± 0.003	$2.96e$-09	$7.21e$-10
STAR	$\mathbf{0.7560 \pm 0.003}$	$\mathbf{1.0595 \pm 0.003}$	–	–

Table IV. Performance comparison of various methods on the iPhone-iPad-iMac dataset regarding RMSE and MAE..

Methods	MAE	RMSE	p-value (MAE)	p-value (RMSE)
SVD++	0.6040 ± 0.002	0.9389 ± 0.002	$2.88e$-09	$5.79e$-11
RMR	0.5985 ± 0.003	0.9333 ± 0.004	$1.77e$-08	$7.61e$-07
FM	0.6117 ± 0.003	0.9352 ± 0.003	$5.55e$-10	$2.19e$-08
CMF	0.6035 ± 0.003	0.9371 ± 0.003	$3.30e$-09	$6.60e$-10
STAR	$\mathbf{0.5889 \pm 0.002}$	$\mathbf{0.9261 \pm 0.003}$	–	–

Results on the iPhone-iPad dataset are shown in Table III. We have the following observations: 1) Our STAR achieves the MAE of 0.7560 and the RMSE of 1.0595, respectively. It shows substantial improvements over SVD++, RMR, FM, CMF of 3.36%, 1.54%, 5.52%, 1.83% in MAE, and 5.55%, 1.13%, 2.68%, 3.48% in RMSE, respectively. We also conducted the paired two-sample t-test based on the five-fold cross-validation results. All the p-values between our model and each of the baselines are much smaller than 0.05, which indicates that the improvements are statistically significant. This is mainly because STAR not only utilizes the dual-heterogeneous data, but also captures common features and distinguishes specific features of Apps on different platforms. 2) The experimental results of CMF are superior to SVD++ in both MAE and RMSE. Because in the construction of the dataset, we only selected those users who have rated two platforms. CMF merges the users across different platforms by identifying the same user IDs, and it thus would increase the density of the data and improve the performance. And 3) FM and CMF achieve suboptimal performance, as compared with RMR, since they do not leverage the extra textual content, which justifies the importance and necessity of incorporating the textual information.

Results on the iPhone-iPad-iMac dataset are summarized in Table IV. We can observe that: 1) Our STAR model obtains the MAE of 0.5889 and the RMSE of 0.9261, which gains improvements over SVD++, RMR, FM, CMF at 2.50%, 1.60%, 3.73%, 2.42% in MAE, and 1.36%, 0.77%, 0.97%, 1.17% in RMSE, respectively. The results of p-values confirm the statistically significant improvements. 2) The improvement of CMF over SVD++ is not sufficiently obvious. The reason is that we selected the users who have at least two ratings, while only 15.68% of the users have rating records at least on two platforms. Thus the performance of CMF is not very significant.

5.4. Handling Cold-Start Problems (RQ2)

5.4.1. New-User Cold-Start Problem. As illustrated in Section 3.5 and Fig. 4, the new-user cold-start problem refers to the existing user appearing on a new platform. We conducted the experiment on the iPhone-iPad-iMac dataset. There exist $19,116$ users who have ratings on at least two platforms, which is 15.68% of the whole $121,905$ users. For each of these users, we first randomly selected one platform from those whereby he/she has rating records, and removed the ratings on this platform. Thereafter, we used the remaining ratings for training, and the removed ratings for testing. We repeated this experimental settings five times and reported the average results. Since

Table V. Performance comparison of various methods in handling new-user cold-start problem.

Methods	MAE	RMSE	p-value (MAE)	p-value (RMSE)
RMR	0.7003 ± 0.003	1.0592 ± 0.002	$2.67e$-09	$3.97e$-10
FM	0.7223 ± 0.003	1.0690 ± 0.003	$7.67e$-11	$1.05e$-10
CMF	0.7041 ± 0.004	1.0783 ± 0.003	$1.10e$-09	$4.04e$-11
STAR	$\mathbf{0.6849 \pm 0.003}$	$\mathbf{1.0344 \pm 0.003}$	–	–

Table VI. Performance comparison of various methods in handling new-App cold-start problem.

Methods	MAE	RMSE	p-value (MAE)	p-value (RMSE)
CTR	0.8746 ± 0.003	1.2310 ± 0.003	$6.76e$-10	$6.98e$-10
FM	0.9017 ± 0.003	1.2452 ± 0.004	$2.64e$-11	$1.76e$-11
CMF	0.8736 ± 0.002	1.2471 ± 0.002	$8.17e$-10	$2.48e$-11
STAR	$\mathbf{0.8529 \pm 0.002}$	$\mathbf{1.2063 \pm 0.003}$	–	–

SVD++ regards the same user on different platforms as different users, it cannot handle new-user cold-start problem. We compared our STAR method with these three approaches: 1) RMR; 2) FM; and 3) CMF.

Results are displayed in Table V. From this table, we observed that STAR achieves the MAE of 0.6849 and the RMSE of 1.0344, which gains improvements over RMR, FM, CMF at 2.20%, 5.18%, 2.73% in MAE, and 2.34%, 3.24%, 4.07% in RMSE, respectively. The paired two-sample t-test also support the conclusion of significant improvements. These experimental results reflect that user interests are almost invariant across different platforms, and it is thus reasonable to leverage a user's latent factor on other platforms to help the rating prediction on a new platform.

5.4.2. New-App Cold-Start Problem. The new-App cold-start problem refers to such cases that developers release an existing App on a new platform and the App has no ratings on the new platform. We carried out experiments on the iPhone-iPad-iMac dataset. The textual content used here is App descriptions. Each App exists on these three platforms. For each App, we first randomly selected its one platform and removed the ratings on this platform. The remaining ratings of each App were used for training, and the removed ratings were used for testing. We repeated this experimental settings five times and reported the average results. Both RMR and CTR are semantics enhanced techniques, while CTR is a method to handle out-of-matrix cold-start problem which is the same in our scenario. So we used CTR to solve our new-App cold-start problem. FM treats the platform as a context and the context information of new App can be obtained from other Apps. CMF regards the App on all platforms as a single one and there are enough information about the App on the known platforms. In brief, we compared our STAR method with: 1) CTR; 2) FM; and 3) CMF.

The comparison results are summarized in Table VI. The results show that: 1) Our STAR obtains the MAE of 0.8529 and the RMSE of 1.2063, which gains improvements over CTR, FM, CMF at 2.48%, 5.41%, 2.37% in MAE, and 2.01%, 3.12%, 3.27% in RMSE, respectively. The improvements are statistically significant. 2) CTR outperforms FM and CMF, since textual content provides sufficient information about the App in the new-App cold-start scenario.

5.5. User Preference on App-Platform (RQ3)

In our datasets, we only know user rated an App on a specific platform. However, we are not sure whether the user likes the App on the platform he/she has rated as compared with other platforms, or it is just the first platform that the user encounters

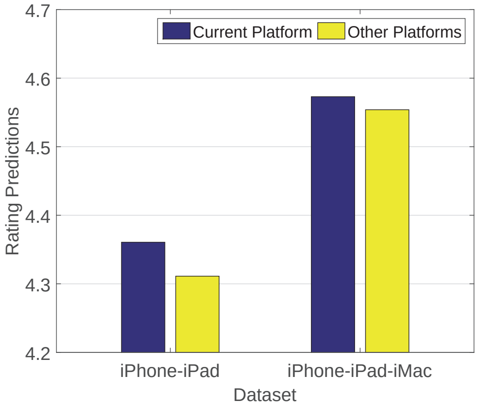


Fig. 9. Comparison of current platform and other platforms.

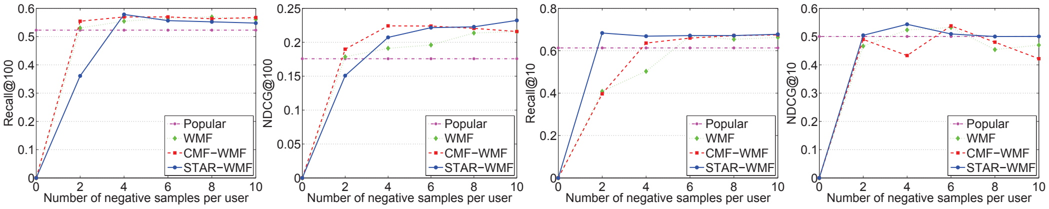
the App. To answer this question, we explored the prediction of user's rating of an App on the current platform (*i.e.*, the platform that was rated on by the target user), but also the prediction of the ratings on other platforms. If the rating prediction of the current platform is statistically higher than other platforms, it demonstrates the current platform is favoured by the users.

For each App that each target user has consumed, we predicted the ratings for both current platform and other platforms and compute the average results. Fig. 9 shows the average score of rating prediction results of our STAR model for both current platform and other platforms of all users on the two datasets. We have the following observations: 1) Our method favors the current platform better, which indicates that our STAR model effectively towards the platform of an App that maximizes its chances of being acquired by the target user. In other words, our model is able to recommend platform-aware Apps. 2) The gap between rating predictions of the current platform and other platforms on the iPhone-iPad dataset is larger than that of iPhone-iPad-iMac dataset. This is due to the dataset construction strategy. In particular, in the dataset construction process, we selected users who rated on two platforms on the iPhone-iPad dataset; in contrast, we selected users who rated at least twice on the iPhone-iPad-iMac dataset (according to our statistics, only 15.68% of the users rated on two or more platforms). Obviously, users' preference on the platform on the iPhone-iPad dataset is easier to be captured compared with that of the iPhone-iPad-iMac dataset.

5.6. Justification of Common Features and Specific Features (RQ4)

In the overall performance comparison, STAR outperforms CMF in both datasets, which demonstrates the importance of capturing common features and specific features of Apps across different platforms. To further understand the influence of common features and specific features of App latent factors, we performed experiments by removing common features w_j and specific features $\mathrm{M}\boldsymbol{\theta}_{d_{js}}$ in Eqn. (4), separately. For convenience, we used cSTAR to represent “STAR with common features only”, and sSTAR to represent “STAR with specific feature only”.

Table VII and Table VIII show the results of importance comparisons of common features and specific features. We have the following observations: 1) STAR outperforms cSTAR, sSTAR of 2.71%, 3.60% in MAE, and 2.12%, 4.05% in RMSE on the iPhone-iPad dataset, respectively. At the same time, STAR's improvements over cSTAR, sSTAR on the iPhone-iPad-iMac dataset are 3.33%, 6.51% in MAE, and 1.10%, 2.22% in RMSE. The results illustrate that the combination of common features and specific features



(a) iPhone-iPad (b) iPhone-iPad (c) iPhone-iPad-iMac (d) iPhone-iPad-Mac

Fig. 10. Top-N results of various numbers of negative samples for each user on the iPhone-iPad and iPhone-iPad-iMac datasets.

Table VII. Importance comparisons of common features and specific features on the iPhone-iPad dataset.

Methods	MAE	RMSE	p-value (MAE)	p-value (RMSE)
cSTAR	0.7771 ± 0.004	1.0824 ± 0.003	$7.57e$-10	$8.72e$-09
sSTAR	0.7842 ± 0.003	1.1042 ± 0.002	$2.37e$-10	$5.39e$-13
STAR	$\mathbf{0.7560 \pm 0.003}$	$\mathbf{1.0595 \pm 0.003}$	–	–

Table VIII. Importance comparisons of common features and specific features on the iPhone-iPad-iMac dataset.

Methods	MAE	RMSE	p-value (MAE)	p-value (RMSE)
cSTAR	0.6092 ± 0.003	0.9364 ± 0.003	$8.83e$-10	$2.13e$-07
sSTAR	0.6299 ± 0.003	0.9471 ± 0.002	$5.31e$-11	$1.24e$-09
STAR	$\mathbf{0.5889 \pm 0.002}$	$\mathbf{0.9261 \pm 0.003}$	–	–

significantly improved the performance. 2) cSTAR outperforms sSTAR of 0.91% in MAE and 1.97% on the iPhone-iPad dataset, 3.29% in MAE and 1.13% in RMSE on the iPhone-iPad-iMac dataset. Obviously, common features have greater impacts than specific features in our framework.

5.7. Evaluation of Top-N Recommendation (RQ5)

The optimization for recommender systems has long been divided into rating prediction and top-N recommendation, which leads to two branches of evaluation metrics — error-based (*e.g.*, MAE and RMSE) and accuracy-based (*e.g.*, recall and NDCG). According to the conclusion drawn from [Cremonesi et al. 2010], there is no monotonic relation between error metrics and accuracy metrics. It means that even though a method achieves a lower error rate for rating prediction, it does not necessarily mean it will outperform other algorithms for the top-N recommendation. One of the key insights is in the modeling of missing data, which is crucial for a model to obtain good performance in ranking unconsumed items for users [He et al. 2016]. Since rating prediction models [Hu et al. 2008; Koren 2010] account for observed entries only and forgo the missing data, they may be suboptimal for the top-N task. As such, we resorted to make some adjustments upon STAR to apply it to the top-N task. WMF [Hu et al. 2008] is a top-N recommendation algorithm using only implicit feedback datasets, which is based on collaborative filtering. We incorporated the core part of STAR (Eqn. (2)) into WMF, and denoted it as STAR-WMF. Meanwhile, CMF is also suitable for being embedded into WMF, which just ties the same user on different platforms together, and we denoted it as CMF-WMF. In addtion, non-personalized algorithm, Popular, is added into the performance comparison, which ranks items by the number of ratings an item has. As discussed in Section 5.1.2, WMF is time-consuming and

Table IX. Evaluation of top-N recommendation on the iPhone-iPad dataset.

Methods	Recall@100	NDCG@100	p-value (Recall)	p-value (NDCG)
Popular	0.5227 ± 0.000	0.1757 ± 0.000	$1.58e$-11	$1.48e$-11
WMF	0.5695 ± 0.003	0.2158 ± 0.002	$1.02e$-10	$2.12e$-09
CMF-WMF	0.5696 ± 0.002	0.2241 ± 0.003	$2.74e$-08	$3.66e$-08
STAR-WMF	$\mathbf{0.5782 \pm 0.002}$	$\mathbf{0.2321 \pm 0.003}$	–	–

Table X. Evaluation of top-N recommendation on the iPhone-iPad-iMac dataset.

Methods	Recall@10	NDCG@10	p-value (Recall)	p-value (NDCG)
Popular	0.6133 ± 0.000	0.5005 ± 0.000	$6.21e$-12	$4.42e$-11
WMF	0.6686 ± 0.002	0.5321 ± 0.003	$3.12e$-09	$9.19e$-09
CMF-WMF	0.6745 ± 0.003	0.5379 ± 0.004	$2.38e$-08	$1.63e$-07
STAR-WMF	$\mathbf{0.6834 \pm 0.003}$	$\mathbf{0.5434 \pm 0.003}$	–	–

memory-consuming for large-scale data, which is caused by accounting for all missing data. Instead, we sampled various numbers of missing data as negative samples for each user to approximate the result of WMF. And we found the performance of each method becomes stable when the number of negative samples increases. Fig. 10 reveals the performance of algorithms with various numbers of negative samples for each user on the iPhone-iPad and iPhone-iPad-iMac datasets. We selected the best results for each algorithm on each dataset to report the final comparisons.

Final results on the iPhone-iPad and iPhone-iPad-iMac datasets are shown in Table IX and Table X, respectively. From these tables, we can observe that: 1) STAR-WMF achieves the Recall@100 of 0.5782 and the NDCG@100 of 0.2321 on the iPhone-iPad dataset, which gains improvements over Popular, WMF, CMF-WMF at 10.61%, 1.53%, 1.50% in Recall@100, and 32.10%, 7.55%, 3.57% in NDCG@100. Meanwhile, STAR-WMF outperforms Popular, WMF, CMF-WMF of 11.43%, 2.21%, 1.32% in Recall@10, and 8.57%, 2.12%, 1.02% in NDCG@10 on the iPhone-iPad-iMac dataset. The results show that STAR can also have a great performance in the top-N task when it is embedded into a top-N recommendation algorithm. 2) Personalized algorithms (WMF, CMF-WMF, and STAR-WMF) outperforms non-personalized algorithm (Popular), since the ranking result heavily depends on individual user's preferences.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel cross-platform App recommendation framework via jointly modeling numerical ratings and textual content. Particularly, it improves the performance of the rating prediction by capturing common features and distinguishing specific features of Apps across multiple platforms. It is capable of alleviating the data sparsity problem via forcing common feature sharing across platforms, which is particularly beneficial to less popular platforms. Meanwhile, it is able to solve the new-user and new-App cold-start problems to a certain extent. To validate the effectiveness of our proposed approach, we constructed two benchmark datasets. Extensive experiments on the two datasets demonstrated the efficacy of our STAR method. We also performed micro-analysis to show how our method targets at particular platforms of Apps, how the common features and specific features of Apps affect the results, and how does our framework perform in the top-N recommendation.

In the future, we plan to extend our work in the following three aspects: 1) Modeling users implicit feedback on different platforms. Since implicit feedback (*e.g.*, browsing history, purchasing history) are richer and easier to collect than explicit ratings, it will be more beneficial to model users' implicit feedback. 2) Training topic models and latent factor methods in an unified framework, which may further improve the performance. 3) Realizing the App recommendation task in an online manner. Users' personal interests change over time, so do the topics of reviews' content of Apps. It would be helpful to utilize users' reviews to capture the dynamic changes of both users' preferences and Apps' features as has been studied in [Zhang et al. 2014; He et al. 2015].

REFERENCES

Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. 2005. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems* 23, 1 (2005), 103–145.

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

Yang Bao, Hui Fang, and Jie Zhang. 2014. TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2–8.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

Matthias Böhmer, Lyubomir Ganev, and Antonio Krüger. 2013. Appfunnel: A framework for usage-centric evaluation of recommender systems that suggest mobile applications. In *Proceedings of the International Conference on Intelligent User Interfaces*. ACM, 267–276.

John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 43–52.

Deng Cai, Qiaozhu Mei, Jiawei Han, and Chengxiang Zhai. 2008. Modeling hidden topics on document manifold. In *Proceedings of the ACM Conference on Information and Knowledge Management*. ACM, 911–920.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of the Advances in Neural Information Processing Systems Conference*. 288–296.

Ning Chen, Steven CH Hoi, Shaohua Li, and Xiaokui Xiao. 2015. Simapp: a framework for detecting similar mobile applications by online kernel learning. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. ACM, 305–314.

Ning Chen, Steven CH Hoi, Shaohua Li, and Xiaokui Xiao. 2016. Mobile App Tagging. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. ACM, 63–72.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & Deep Learning for Recommender Systems. *arXiv preprint arXiv:1606.07792* (2016).

Enrique Costa-Montenegro, Ana Belén Barragáns-Martínez, and Marta Rey-López. 2012. Which App? A recommender system of applications in markets: Implementation of the service for monitoring users' interaction. *Expert Systems with Applications* 39, 10 (2012), 9367–9375.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, 39–46.

Gintare Karolina Dziugaite and Daniel M. Roy. 2015. Neural Network Matrix Factorization. *CoRR* abs/1511.06443 (2015). http://arxiv.org/abs/1511.06443

Ignacio Fernández-Tobías, Iván Cantador, Marius Kaminskas, and Francesco Ricci. 2012. Cross-domain recommender systems: A survey of the state of the art. In *Spanish Conference on Information Retrieval*.

Yong Ge, Hui Xiong, Alexander Tuzhilin, and Qi Liu. 2014. Cost-aware collaborative filtering for travel tour recommendations. *ACM Transactions on Information Systems* 32, 1 (2014), 4.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware Explainable Recommendation by Modeling Aspects. In *Proceedings of the ACM International on Conference on Information and Knowledge Management*. ACM, 1661–1670.

Xiangnan He, Ming Gao, Min-Yen Kan, Yiqun Liu, and Kazunari Sugiyama. 2014a. Predicting the popularity of web 2.0 items based on user comments. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 233–242.

Xiangnan He, Min-Yen Kan, Peichu Xie, and Xiao Chen. 2014b. Comment-based Multi-view Clustering of Web 2.0 Items. In *Proceedings of the International Conference on World Wide Web*. ACM, 771–782.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

Gregor Heinrich. 2008. Parameter estimation for text analysis. *Technical Report, University of Leipzig* (2008).

Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 230–237.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 50–57.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning* 42, 1-2 (2001), 177–196.

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

Thomas Hofmann and Jan Puzicha. 1999. Latent class models for collaborative filtering. In *Proceedings of International Joint Conference on Artificial Intelligence*. AAAI Press, 688–693.

Cheng-Kang Hsieh, Longqi Yang, Honghao Wei, Mor Naaman, and Deborah Estrin. 2016. Immersive Recommendation: News and Event Recommendations Using Personal Digital Traces. In *Proceedings of the International Conference on World Wide Web*. ACM, 51–62.

Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. 2013. Personalized recommendation via cross-domain triadic factorization. In *Proceedings of the International Conference on World Wide Web*. ACM, 595–606.

Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 263–272.

Mohsen Jamali and Laks Lakshmanan. 2013. HeteroMF: recommendation in heterogeneous information networks using context dependent factor models. In *Proceedings of the International Conference on World Wide Web*. ACM, 643–654.

Meng Jiang, Peng Cui, Xumin Chen, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2015. Social recommendation with cross-domain transferable knowledge. *IEEE Transactions on Knowledge and Data Engineering* 27, 11 (2015), 3084–3097.

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

Alexandros Karatzoglou, Linas Baltrunas, Karen Church, and Matthias Böhmer. 2012. Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In *Proceedings of the ACM international Conference on Information and Knowledge Management*. ACM, 2527–2530.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 426–434.

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

Yehuda Koren and Robert Bell. 2011. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, 145–186.

Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

Chung-Yi Li and Shou-De Lin. 2014. Matching users and items across domains to improve the recommendation quality. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 801–810.

Chen Lin, Runquan Xie, Xinjun Guan, Lei Li, and Tao Li. 2014b. Personalized news recommendation via implicit social experts. *Information Sciences* 254 (2014), 1–18.

Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural Computation* 19, 10 (2007), 2756–2779.

Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. 2013. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 283–292.

Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. 2014a. New and improved: modeling versions to improve app recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 647–656.

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

Christoph Lippert, Stefan Hagen Weber, Yi Huang, Volker Tresp, Matthias Schubert, and Hans-Peter Kriegel. 2008. Relation prediction in multi-relational domains using matrix factorization. In *Proceedings of the NIPS Workshop: Structured Input-Structured Output*. Citeseer.

Bin Liu, Deguang Kong, Lei Cen, Neil Zhenqiang Gong, Hongxia Jin, and Hui Xiong. 2015. Personalized mobile app recommendation: Reconciling app functionality and user privacy preference. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. ACM, 315–324.

Duen-Ren Liu, Pei-Yun Tsai, and Po-Huan Chiu. 2011. Personalized recommendation of popular blog articles for mobile applications. *Information Sciences* 181, 9 (2011), 1552–1572.

Qi Liu, Haiping Ma, Enhong Chen, and Hui Xiong. 2013. A survey of context-aware mobile recommendations. *International Journal of Information Technology & Decision Making* 12, 01 (2013), 139–172.

Qiang Liu, Shu Wu, and Liang Wang. 2015. COT: Contextual Operating Tensor for Context-Aware Recommender Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 203–209.

Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. 2007. Collaborative filtering and the missing at random assumption. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 267–276.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, 165–172.

Andriy Mnih and Ruslan Salakhutdinov. 2007. Probabilistic matrix factorization. In *Proceedings of the Advances in Neural Information Processing Systems Conference*. 1257–1264.

Liqiang Nie, Yi-Liang Zhao, Xiangyu Wang, Jialie Shen, and Tat-Seng Chua. 2014. Learning to recommend descriptive tags for questions in social forums. *ACM Transactions on Information Systems* 32, 1 (2014), 5.

Cosimo Palmisano, Alexander Tuzhilin, and Michele Gorgoglione. 2008. Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering* 20, 11 (2008), 1535–1549.

Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. 2009. Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the ACM conference on Recommender systems*. ACM, 265–268.

Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. 2011. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Transactions on Information Systems* 29, 4 (2011), 20.

Steffen Rendle. 2011. *Context-aware ranking with factorization models*. Springer.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 452–461.

Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 635–644.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the International Conference on World Wide Web*. ACM, 285–295.

Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 253–260.

Kent Shi and Kamal Ali. 2012. GetJar mobile application recommendations with very sparse datasets. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 204–212.

Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 650–658.

Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2008. Matrix factorization and neighbor based algorithms for the netflix prize problem. In *Proceedings of the ACM conference on Recommender Systems*. ACM, 267–274.

Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 448–456.

Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the International Conference on World Wide Web*. ACM, 537–546.

Dragomir Yankov, Pavel Berkhin, and Rajen Subba. 2013. Interoperability ranking for mobile applications. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 857–860.

Peifeng Yin, Ping Luo, Wang-Chien Lee, and Min Wang. 2013. App recommendation: a contest between satisfaction and temptation. In *Proceedings of the ACM International Conference on Web Search and Data Mining*. ACM, 395–404.

Liang Zhang, Deepak Agarwal, and Bee-Chung Chen. 2011. Generalizing matrix factorization through flexible regression priors. In *Proceedings of the ACM Conference on Recommender Systems*. ACM, 13–20.

Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 83–92.

Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 236–241.

Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 315–324.

Hengshu Zhu, Enhong Chen, Kuifei Yu, Huanhuan Cao, Hui Xiong, and Jilei Tian. 2012. Mining personal context-aware preferences for mobile users. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 1212–1217.

Hengshu Zhu, Chuanren Liu, Yong Ge, Hui Xiong, and Enhong Chen. 2015. Popularity modeling for mobile Apps: A sequential approach. *IEEE Transactions on Cybernetics* 45, 7 (2015), 1303–1314.

Hengshu Zhu, Hui Xiong, Yong Ge, and Enhong Chen. 2014. Mobile app recommendations with security and privacy awareness. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 951–960.