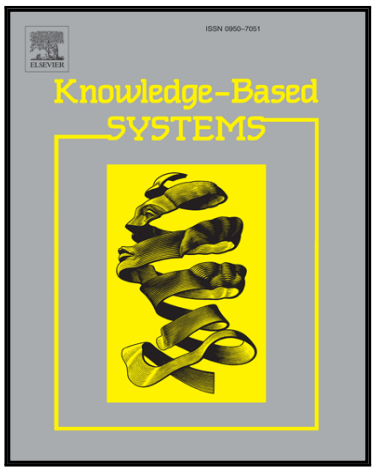


Mobi-SAGE-RS: A Sparse Additive Generative Model-based Mobile Application Recommender System

Hongzhi Yin, Weiqing Wang, Liang Chen, Xingzhong Du,
Quoc Viet Hung Nguyen, Zi Huang

PII: S0950-7051(18)30249-1
DOI: 10.1016/j.knosys.2018.05.028
Reference: KNOSYS 4348

To appear in: *Knowledge-Based Systems*

Received date: 21 November 2017
Revised date: 5 April 2018
Accepted date: 14 May 2018

Please cite this article as: Hongzhi Yin, Weiqing Wang, Liang Chen, Xingzhong Du, Quoc Viet Hung Nguyen, Zi Huang, Mobi-SAGE-RS: A Sparse Additive Generative Model-based Mobile Application Recommender System, *Knowledge-Based Systems* (2018), doi: 10.1016/j.knosys.2018.05.028

Mobi-SAGE-RS: A Sparse Additive Generative Model-based Mobile Application Recommender System

Hongzhi Yin[a,*], Weiqing Wang[a], Liang Chen[b], Xingzhong Du[a], Quoc Viet Hung Nguyen[c], Zi Huang[a]

[a]*Building 78, Staff House Road, The University of Queensland, Brisbane, QLD, 4072, Australia*
[b]*Sun Yat-sen University*
[c]*Griffith University*

Abstract

With the rapid prevalence of smart mobile devices and the dramatic proliferation of mobile applications (Apps), App recommendation becomes an emergent task that will benefit different stockholders of mobile App ecosystems. However, the extreme sparsity of user-App matrix and many newly emerging Apps create severe challenges, causing CF-based methods to degrade significantly in their recommendation performance. Besides, unlike traditional items, Apps have rights to access users' personal resources (e.g., location, message and contact) which may lead to security risk or privacy leak. Thus, users' choosing of Apps are influenced by not only their personal interests but also their privacy preferences. Moreover, user privacy preferences vary with App categories.

In light of the above challenges, we propose a mobile sparse additive generative model (Mobi-SAGE) to recommend Apps by considering both user interests and category-aware user privacy preferences in this paper. To overcome the challenges from data sparsity and cold start, Mobi-SAGE exploits both textual and

*Corresponding author
Email address: h.yin1@uq.edu.au (Hongzhi Yin)

visual content associated with Apps to learn multi-view topics for user interest modeling. We collected a large-scale and real-world dataset from 360 App store - the biggest Android App platform in China, and conducted extensive experiments on it. The experimental results demonstrate that our Mobi-SAGE consistently and significantly outperforms the other existing state-of-the-art methods, which implies the importance of exploiting category-aware user privacy preferences and the multi-modal App content data on personalized App recommendation.

Keywords: recommender system, mobile applications, user modeling, privacy, sparse additive generative model, cold start

1. Introduction

Recent years have witnessed the rocketing development and prevalence of smart mobile devices, such as smart phones. An important driver behind the wide adoption of smart mobile devices is the emergence of application ("App") stores, where the third party developers publish mobile Apps that users can download to augment their mobile devices' functionality. According to a recent report[1], as of November 2015, there were over 1.8 million Apps with over 60 billion cumulative downloads on Google Play (one of the largest App markets), and the number of Apps grew by around 80% between July 2013 and November 2015. With the dramatically increasing number of Apps, it is hard for users to explore the huge world of Apps and locate relevant Apps. Thus, effective personalized App recommender systems are in a urgent need. However, we discover that most mainstream App markets (e.g., Google Play and Apple App Store) currently do not provide the functionality of personalized App recommendation. In this paper, we focus on

[1] `http://www.statista.com/`

developing an effective mobile App recommender system. App recommendation is highly challenging for the following reasons:

Privacy Concern. Apps are very different from the items in traditional recommender systems such as movies, products and locations, as they have the privileges to access users' personal information such as messages, contacts and locations. Android and Iphone use permission systems to control the privileges of Apps. Apps can only access privacy and security-relevant resources if the user approves an appropriate permission request. For example, an Android application can only send text messages if it has the permission "SEND_SMS". As reported by NBC news[2], users have grown so concerned about privacy on their mobile phones. For instance, many users have avoided downloading some mobile Apps, and many others have removed Apps which may have acess to their personal data. Besides, different users might have varying privacy preferences, e.g., user u_a does not want the Apps to share contacts while user u_b tends to keep his/her personal locations (e.g., home locations or workplaces) hided from the third party Apps. Moreover, user privacy preferences tend to vary with App categories/functionalities[3]. For instances, users tend to permit navigation Apps to access their locations, while they might forbid entertainment Apps from doing that.

Cold Start. Cold start problem has been a major challenge for conventional recommender systems [1, 2, 3]. As App stores or platforms are expanding rapidly and the number of mobile Apps is increasing dramatically, cold start remains a critical problem in the domain of mobile App recommendation, which consists

[2] `http://www.nbcnews.com/`

[3] App categories and functionalities are equivalent, as categories in the App stores are defined according to the Apps' functionalities.

of the cold-start App problem and the cold-start user problem. Apps which have not received any rating or download are called cold-start Apps. Similarly, users who have not rated or downloaded any App are called cold-start users. There are thousands of new mobile Apps released daily on the App stores, and more and more users are beginning to use mobile devices. Therefore, collaborative filtering and matrix factorization methods that only use user-item interaction information are ineffective.

Conventional recommender systems [4, 5, 6, 7, 8, 9] essentially aim to learn each user's *interests* and each item's *functionalities*, given the interaction activities between the users and items. Based on the learned knowledge, given a user, the systems recommend the items whose functionalities are highly similar to the interests of this user. For instance, latent factor-based approaches [10, 6, 7, 11] model a user with a latent vector and an item with another latent vector; and an item is recommended to a user if the item's latent vector is similar to the user's latent vector. Such recommender systems have been successfully applied to recommend many different items such as movies (e.g., Netflix) [12], music [13], and points-of-interest [14, 15]. However, these approaches cannot address the above two problems.

Recently, a mobile App recommender system has been proposed by Zhu et al. in [16] in which recommendations are made by considering both Apps' popularity and the privacy risks. However, this system provides the same recommendations to all the users and thus it is not a personalized App recommender system. Liu et al. [17] extended the matrix factorization model for personalized App recommendation by integrating user privacy preferences. But, both [16] and [17] ignored that user privacy preferences are not always stable, and tend to vary with App

categories. Moreover, their methods cannot overcome the cold-start user issues. Lin et al. [18] integrated the follower information of the App's official Twitter account to address the cold-start App problem, but only a small portion of Apps in the real-life App markets have an official Twitter account. Besides, their method [18] cannot address the challenges from cold-start users and security or privacy risk.

In light of this, we propose a mobile sparse additive generative model (Mobi-SAGE) for mobile App recommendation in this paper, which jointly learns user interests and category-aware user privacy preferences in a unified way. Although an App's functionalities may match a user's personal interests, the user could still refuse to install it if it does not respect his/her privacy preferences w.r.t. the specific category of that App. Therefore, our Mobi-SAGE model considers both personal interests and category-aware personal privacy preferences. However, given a user, his/her rated or downloaded Apps with a specific category are extremely sparse. It is very difficult to directly learn the user-category-specific privacy preferences without overfitting. To combat the data sparsity issue, we decompose user-category-specific privacy preferences into two parts: user-specific privacy preferences and category-specific privacy preferences. The first part captures the user's stable privacy preferences, while the second part exploits the public's collective privacy preferences for a specific category of Apps (i.e., the wisdom of crowds) that captures the common patterns in permission requests for Apps with that specific category. Thus, an App with "unusual" permission requests (e.g., malware Apps) will enjoy low priority to be recommended.

To address the cold-start App problem, we exploit the wealthy of multi-modal content data associated with Apps (e.g., text descriptions and screenshot images)

to learn multi-view topics for user interest modeling. Meanwhile, we assume that each topic corresponds to an App category. Thus, both user interests and category-aware user privacy preferences can be learnt in a unified process. Inspired by [19], to leverage the collaborative filtering information, a topic is extended to generate App-IDs besides the multimodal content, to capture the App co-occurrence patterns. To our best knowledge, ideas for unifying the influence of user privacy preferences, collaborative filtering, content-based recommendation are unexplored and very challenging. To overcome the cold-start user problem, we integrate the general public's collective interests that can capture the trends of App categories by latent topics. Thus, our Mobi-SAGE can provide App recommendation to new mobile users by considering both the public's collective interests and their functionality-aware collective privacy preferences.

We summarize the key contributions as follows:

1. We are the *first* one exploiting both user interests and category-aware user privacy preferences in personalized App recommendation.
2. To the best of our knowledge, we are the first to exploit and integrate the public's collective interests and their collective privacy preferences to address the cold-start user problem and to explore the multi-modal content data of Apps to address the cold-start App problem.
3. We propose a probabilistic generative model Mobi-SAGE to jointly learn multi-view topics, user interests and category-aware user privacy preferences.
4. We collect a large-scale App dataset from 360 App store, and use it to comprehensively evaluate our approach and state-of-the-art App recommendation techniques. The experimental results show the superiority of our Mobi-

SAGE, especially in addressing the cold-start problem.

Note that we presented our preliminary study of mobile app recommendation in the prior work [20] as an abstract paper. In this article, we make significant revision and add substantial new materials compared with the work presented in [20]. Specifically, this article makes the following new contributions:

1 We provide the preliminaries in Section 3.1, a system overview in Section 3.2 and more technical details about Mobi-SAGE in Section 3.3, Section 3.4 and Section 3.6.

2 We impose sparse coding on user interest modeling and topic representation by introducing zero-mean Laplace distributions as prior distributions in Section 3.5.

3 We conduct a detailed analysis of time complexity of the proposed model in Section 3.6.

4 An efficient online recommendation scheme is designed to significantly speed up the online recommendation by separating the online computation from the offline computation to the maximum in Section 3.7. We also develop a recommendation scheme for cold-start users and cold-start apps in this section.

5 In Section 4, we have conducted much more comprehensive evaluations. First, we add the evaluation of the impact of different features in Mobi-SAGE by comparing with its four variants. Second, we also evaluate the recommendation effectiveness for cold-start users and apps. Third, the impact of tuning model parameters is also studied. Last, the efficiency of the newly designed online recommendation scheme is also evaluated.

6 We provide a more thorough analysis of Mobi-SAGE and also a more comprehensive review of the related work in Section 5.

The paper is organized as follows. In Section 2, we first present the preliminaries about the private resources and permissions, and then formulate the research problem in this paper. In Section 3, we detail our proposed model Mobi-SAGE. In Section 4, we first describe the experimental setup and then report the experimental results. Section 5 includes the related work analysis and then we conclude our paper in Section 6.

2. Preliminaries and Problem Formulation

In this section, we first present the private resources and permission system in mobile devices, and then formulate the problem of mobile App recommendation.

2.1. Private Resources and Permissions

We focus on Android Apps in this paper, although our approach is also applicable to other types of Apps (e.g., Iphone Apps). We follow the description of Android resources and permissions in [17]. Specifically, Android system is a permission-based framework to control the privileges of Apps. A permission consists of two elements: resources (e.g., Internet, contact, camera and location) and operations (e.g., read and write), and granting a permission to an App allows the App to operate the corresponding resource in the predefined way. Table 1 shows the permissions related to 10 critical resources. For example, given the permission "READ_CONTACTS", an App is authorized to read a user's contact data. Apps access users' private data by requesting the corresponding permissions.

Apps access users' private data for two main reasons. First, some Apps need to access users' certain types of private data to operate the functionality. For

example, Google Map has to acquire users' location data to provide the navigation functionality and thus requires the "ACCESS_FINE_LOCATION" permission. However, some other Apps might manipulate users' certain types of private data intentionally or unintentionally for non-functionality purpose (i.e., advertisements). According to a survey in [21], around 25% of Android Apps require the access to users' location just for advertisements. According to another survey in [22], 30% Android Apps request the permissions over the sensitive resources that are not used by themselves. Users might have different privacy concerns. Some users might have low privacy concerns and they feel fine to use the Apps at the cost of their privacy, while some other users might pay more attention to their privacy and tend to give up an App or transfer to another App that provides the same or similar functionality but request less private resources.

2.2. Problem Definition

In this section, we first introduce some key concepts and then formulate the problem.

Notation. Through this paper, all vectors are column vectors and are denoted by bold lower case letters (e.g., $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$). We use calligraphic letters to represent sets, e.g., $\mathcal{U}$ and $\mathcal{A}$ represent the user set and App set. For simplicity, we use their corresponding normal letters to denote their cardinalities (e.g., $A = |\mathcal{A}|$). For ease of presentation, Table 2 lists the notations.

Definition 1. ***(Mobile App)*** *A mobile App is a computer program designed to run on mobile devices such as smart phones.*

In our work, a mobile App has four attributes: identifier, description, images and permissions, as shown in Figure 1. We use a to represent an App identifier

Table 1: Privacy-Sensitive Permission Requests.

Resources	Privacy-Sensitive Permission Requests
Contact	READ_CONTACTS
	WRITE_CONTACTS
Message	READ_SMS
	WRITE_SMS
	SEND_SMS
	RECEIVE_SMS
	RECEIVE_MMS
	SEND_RESPONSE_VIA_MESSAGE
Location	ACCESS_FINE_LOCATION
	ACCESS_COARSE_LOCATION
Audio	RECORD_AUDIO
	MODIFY_AUDIO_SETTINGS
Camera	CAMERA
Phone_state	MODIFY_PHONE_STATE
	READ_PHONE_STATE
Phone_call	CALL_PHONE
	CALL_PRIVILEGED
Calendar	READ_CALENDAR
	WRITE_CALENDAR
Call_log	READ_CALL_LOG
	WRITE_CALL_LOG
Browser_history	READ_HISTORY_BOOKMARKS
	WRITE_HISTORY_BOOKMARKS

and $\mathcal{P}_a$ to denote a's permission request set. $\mathcal{T}_a$ and $\mathcal{V}_a$ are used to denote a's textual content and visual content, and we apply bag of words (BoW) to represent an App in both textual and visual spaces. Specifically, $\mathcal{T}_a$ is a collection of words (i.e., a textual document) extracted from a's description and name, and $\mathcal{V}_a$ is a collection of visual words (i.e., a visual document) extracted from the images associated with a. Specifically, for each image, we first adopt the Scale Invariant Feature Transform (SIFT) [23] method, which is a widely used computer vision technique, to produce a set of key points and each keypoint is denoted by a 128-dimensional vector. Subsequently, the keypoints extracted from all the images are grouped into k clusters by a fast k-means algorithm. These clusters form a "visual vocabulary" $\mathcal{V}$, where each cluster is considered as a "visual word" v ($v \in \mathcal{V}$). Finally, each key point is assigned to its closest cluster, and then this key point is represented by the visual word assigned to its closest cluster. As a result, each image is represented as a bag of visual words. There might be more than one image for each App. Thus, for an App a, we combine the visual words extracted from all its images to get a big visual document $\mathcal{V}_a$.

Definition 2. ***(User Profile)*** *For each user u, we create a user profile $\mathcal{D}_u = \{(a,\ t)\}$, which is a set of u's downloading records in the App store. t is the timestamp of u downloading a. Thus, the whole App downloading dataset $\mathcal{D}$ consists of all user profiles, i.e., $\mathcal{D} = \{\mathcal{D}_u : u \in \mathcal{U}\}$ where $\mathcal{U}$ is the set of users.*

Although rating is a good indicator to represent users' preferences, users' rating behaviors are extremely rare in the App markets. Even for the most popular Apps, the proportion of users who provide ratings are rather small. Thus, we collect the users' downloading behaviors in our dataset, which are implicit feedback.

Below we formally formulate our problem as:

Problem 1. ***(Mobile App Recommendation)*** *Given a user set $\mathcal{U}$, a mobile App set $\mathcal{A}$ and a user downloading history dataset $\mathcal{D}$, for each user $u \in \mathcal{U}$, our goal is to recommend top-k mobile Apps that u is most interested in. If user u or App a never appears in the historical dataset $\mathcal{D}$, this recommendation scenario is called out-of-matrix recommendation or cold start recommendation.*

3. The Mobi-SAGE Recommender System

In this section, we first present preliminaries about the SAGE model [24], and then describe our Mobi-SAGE model based on it.

3.1. Preliminaries about SAGE

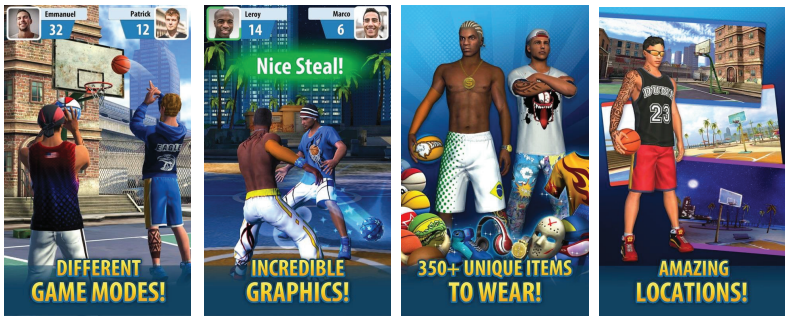
Our proposed model is built based on the Sparse Additive Generative Model (SAGE), which is an effective generative model proposed in [24]. The major feature that distinguishes SAGE from other generative models is that, if a variable is affected by several different factors, it can be generated by the mixture of these factors without learning any explicit indicator variables (i.e., mixture weights) in the log space. SAGE is well known for its robustness to the sparsity problem because of that it reduces the number of variables to learn [19, 25]. The data sparsity problem in Mobile App recommendation motivates us to build our model based on SAGE.

To provide a further illustration of SAGE, we compare a traditional probabilistic mixture generative model (e.g., LCA-LDA [26] and TCAM [27]) with it. Give a user u, we simply assume that u's choosing of Apps to download is influenced by her interests $\boldsymbol{\theta}_u^{user}$. To overcome the data sparsity issue, we introduce the general public's preferences $\boldsymbol{\theta}^0$ as the background model to smooth $\boldsymbol{\theta}_u^{user}$. Thus,

Name: Basketball Starts

Description: The world's best multiplayer Basketball game on mobile, from the creators of multiple smash-hit online sports games!

Images:



Permissions: READ_PHONE_STATE; MODIFY_AUDIO_SETTINGS; ACCESS_FINE_LOCATION …

Figure 1: An Example App.

the likelihood that user u will prefer App a in the traditional probabilistic mixture model is computed as follows.

$$P(a|\boldsymbol{\theta}_u^{user}, \boldsymbol{\theta}^0) = \lambda_u P(a|\boldsymbol{\theta}_u^{user}) + (1 - \lambda_u) P(a|\boldsymbol{\theta}^0) \tag{1}$$

where the two factors, $\boldsymbol{\theta}_u^{user}$ and $\boldsymbol{\theta}^0$, are combined through a linear combination, and λ_u is the personalized mixture weight that can be inferred for each user through a "switching" variable. Note that, in LCA-LDA and TCAM, we need to learn a switching variable for each user. However, in recommender system, this learning can be inaccurate as the data for each user is sparse. As we mentioned before, SAGE combats this problem by avoiding the learning of switching variables. Specifically, SAGE combines personal interests with general public's preferences by simple addition in log space as Equation 2. Besides, SAGE models the *difference* in log-frequencies from a background model. More specifically, we chooses $\boldsymbol{\theta}^0$ to denote the (background) log frequency of a in the whole App

downloading dataset while the component $\boldsymbol{\theta}_u^{user}$ is used to model the deviation in log-frequency from a constant background model. This increases predictive accuracy and robustness to limited training data.

$$P(v|\boldsymbol{\theta}_u^{user}, \boldsymbol{\theta}^0) = P(a|\boldsymbol{\theta}_u^{user} + \boldsymbol{\theta}^0) = \frac{exp(\theta_{u,a}^{user} + \theta_a^0)}{\sum_{a'} exp(\theta_{u,a'}^{user} + \theta_{a'}^0)} \quad (2)$$

3.2. System Overview

To make accurate App recommendation, we propose a sparse additive generative model based recommender system Mobi-SAGE-RS to solve both the privacy concern and cold start problem. As shown in Figure 2, Mobi-SAGE-RS consists of two main parts: offline modeling and online recommendation. The offline model, Mobi-SAGE, is a generative model jointly trained over the textual words, visual information, App-IDs and permission requests in the users' downloading profiles. It is designed to model user preferences to Apps by simultaneously considering the following two factors in a unified manner. 1) *Personal Interests and Privacy Preferences*. Mobi-SAGE models user interest by mining multimodal content information including textual information and image information to solve the data sparsity problem and make the recommendation for cold-start items. In addition, Mobi-SAGE also mines users' personal privacy preferences to provide privacy-aware recommendation. 2) *Public Interests and Privacy Preferences*. Mobi-SAGE models the public interests and privacy preferences to solve the cold-start users problem.

Given a querying user u, the online recommendation part computes a ranking score for each App a by automatically combining u's preference and the public preference over a, which are learned offline by Mobi-SAGE. To speed up the process of online recommendation, we adopt a scalable query processing technique

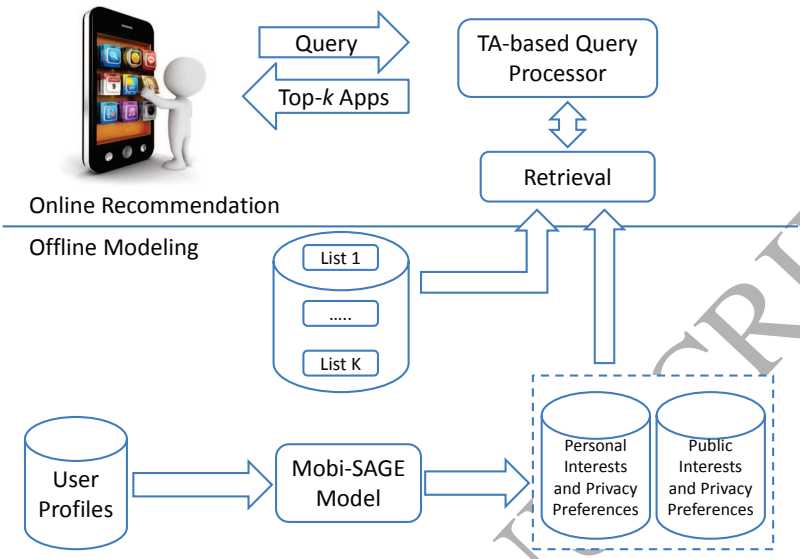


Figure 2: the Architecture of Mobi-SAGE-RS

for top-k recommendations: TA-based algorithm [26, 27]. Specifically, we pre-compute K sorted lists of Apps, where $K = |\mathcal{K}|$ is the number of topics, according to the latent topics $\mathcal{K}$ learned by offline model Mobi-SAGE. In each list, Apps are sorted based on their generative probabilities with respect to the corresponding topic. At query time, we access Apps from the K sorted lists and compute top-k Apps by running the TA algorithm.

3.3. Model Description

To model mobile users' downloading behaviors on App stores, we propose a mobile sparse additive generative model (Mobi-SAGE). Figure 3 is the graphical description of Mobi-SAGE. Before a detailed introduction to Mobi-SAGE, we first list the notations used in this model in Table 2. The shaded circles in Figure 3 model the observed variables (i.e., the input data), which are users' downloading profiles in Mobi-SAGE. Mobi-SAGE is a generative model jointly over the textual

Table 2: Notations used in this paper.

SYMBOL	DESCRIPTION
u, a, p	user u, App a and permission p
w, v	textual word w and visual word v
$\mathcal{U}$, $\mathcal{A}$, $\mathcal{P}$	the sets of users, Apps and permissions
$\mathcal{W}$, $\mathcal{V}$	the textual and visual vocabularies, respectively
$\mathcal{T}_a$	the textual document associated with App a
$\mathcal{V}_a$	the visual document associated with App a
$\mathcal{D}_u$	the profile of user u
$\mathcal{K}$	the set of topics
$\mathcal{P}_a$	the permission request set of App a
$\boldsymbol{\theta}_u^{user}$	the interests of user u
$\boldsymbol{\theta}^0$	the interests of the general public
$\boldsymbol{\vartheta}_u^{user}$	the intrinsic privacy preferences of user u
$\boldsymbol{\vartheta}_z^0$	the privacy preferences of the general public for Apps with topic z
$\boldsymbol{\phi}_z^{topic}$	the textual word vector specific to topic z
$\boldsymbol{\psi}_z^{topic}$	the visual word vector specific to topic z
$\boldsymbol{\varphi}_z^{topic}$	the App-ID vector specific to topic z
$\boldsymbol{\phi}^0$, $\boldsymbol{\psi}^0$, $\boldsymbol{\varphi}^0$	the textual-word, visual-word, App-ID vectors of the background, respectively

words, visual words, App-IDs and permission requests in the users' downloading profiles. It *discovers multi-view topics* and *learns user interests* and *topic-aware user privacy preferences* in a unified way. Below, we will describe each component in Mobi-SAGE.

User Interest Modeling. Intuitively, a user chooses an App by matching his/her personal interests with the content of that App. Inspired by the early work on user interest modeling [28, 29, 26], Mobi-SAGE also uses latent topics, defined as z, to characterize users' personal interests. Specifically, we learn a topic-based vector representation for each user according to his/her downloaded Apps and their associated multi-modal content, denoted as $\boldsymbol{\theta}_u^{user}$. Besides, to alleviate the data sparsity and address the cold-start user problem, we also introduce a topic-based background vector $\boldsymbol{\theta}^0$ to capture the general public's interests (i.e., the common interests among all users). Moreover, adopting the background model $\boldsymbol{\theta}^0$ also makes the learned users' personal interests $\boldsymbol{\theta}^{user}$ more discriminative.

Multi-View Topic Modeling. The quality of the discovered topics are very important for modeling users' interests. To integrate the advantages of both collaborative filtering-based and content-based recommendation methods, a topic z in Mobi-SAGE is designed to be responsible for clustering semantically and visually similar Apps together and capturing the App co-occurrence patterns at the same time. Specifically, a topic z in Mobi-SAGE is associated with three vectors: a textual-word vector $\boldsymbol{\phi}_z^{topic}$, a visual-word vector $\boldsymbol{\psi}_z^{topic}$ and an App-ID vector $\boldsymbol{\varphi}_z^{topic}$. In other words, each topic z in our model is responsible for simultaneously generating textual words, visual words and App-IDs. This design enables $\boldsymbol{\phi}_z^{topic}$, $\boldsymbol{\psi}_z^{topic}$ and $\boldsymbol{\varphi}_z^{topic}$ to be mutually influenced and enhanced during the topic discovery process by associating them. $\boldsymbol{\phi}_z^{topic}$ and $\boldsymbol{\psi}_z^{topic}$ are in charge of grouping semantically and visually similar Apps together like the idea of content-based recommendation methods while $\boldsymbol{\varphi}_z^{topic}$ can capture the App co-occurrence patterns to link relevant Apps together, similar to item-based collaborative filtering recommendation methods. In this way, we are able to provide a multi-view in-

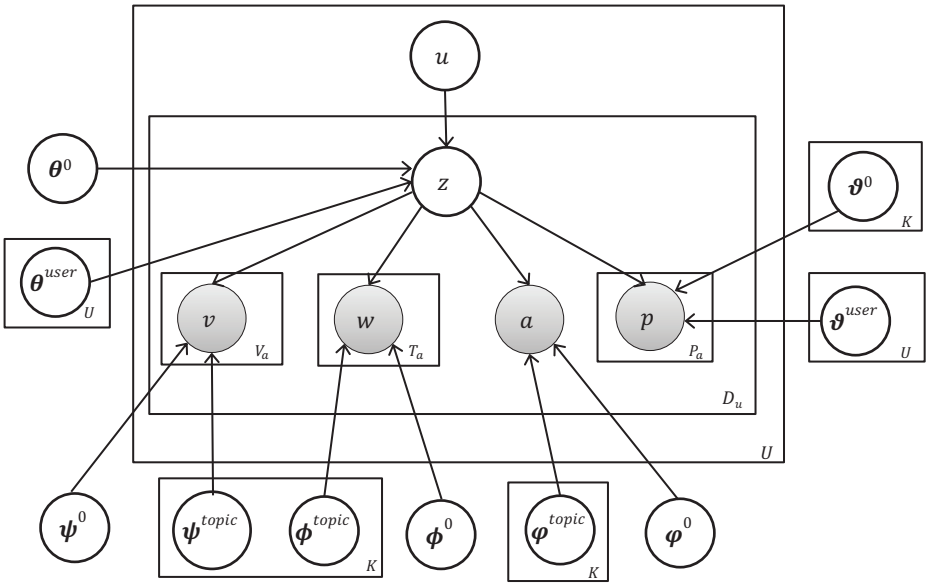


Figure 3: The Graphical Representation of Mobi-SAGE.

terpretation for the topics. By integrating and exploiting multi-modal contents associated with Apps, our Mobi-SAGE can effectively address the cold-start App problem. We also introduce three background models for textual words, visual words and App-IDs, respectively: ϕ^0, ψ^0 and φ^0. We introduce the background models in modeling topics with the aim making the learned topics more discriminative as the background models ϕ^0, ψ^0 and φ^0 would assign high probabilities to non-discriminative and non-informative textual words, visual words and Apps.

For each App a in the user profile $\mathcal{D}_u$, $\mathcal{T}_a$ and $\mathcal{V}_a$ are a bag of textual words and a bag of visual words describing App a in the semantic and visual spaces, respectively. We associate $\mathcal{T}_a$ and $\mathcal{V}_a$ with a latent variable z to indicate the topic of App a. Note that in the traditional topic models [30] such as LDA, a document contains a mixture of topics, and each word has a hidden topic label. This is

reasonable for long documents with multiple themes. However, each App in the App stores (e.g., Google Play) belong to only one category. We therefore assign a single topic z to the textual document $\mathcal{T}_a$ and visual document $\mathcal{V}_a$. By applying this constraint to our Mobi-SAGE model, we aim to build the potential one-to-one correspondence between the discovered latent topics and the categories defined by the App stores. As categories in the App stores are defined according to the Apps' functionalities, a discovered topic is expected to effectively capture and describe the functionalities that the same type of Apps have.

Topic-Aware User Privacy Preference Modeling. Being different from the traditional items or products, the user's decision making for mobile Apps is also influenced by his/her privacy preferences. Moreover, user privacy preferences are not always stable and tend to vary with App categories. Therefore, we need to model topic-aware user privacy preferences in Mobi-SAGE. One alternative is to directly learn a user-category-specific vector $\boldsymbol{\vartheta}_{u,z}$ to capture u's privacy preferences under topic z. Considering that a user's downloaded Apps under a specific category are extremely sparse, and it is very difficult to learn $\boldsymbol{\vartheta}_{u,z}$ without overfitting. Therefore, to combat the data sparsity issue, we take advantage of the additivity in SAGE to decompose the user-category-specific privacy preferences $\boldsymbol{\vartheta}_{u,z}$ into two parts: user-specific privacy preferences $\boldsymbol{\vartheta}_u^{user}$ and category-specific privacy preferences $\boldsymbol{\vartheta}_z^0$ (i.e., $\boldsymbol{\vartheta}_{u,z} = \boldsymbol{\vartheta}_u^{user} + \boldsymbol{\vartheta}_z^0$). The first component $\boldsymbol{\vartheta}_u^{user}$ is used to capture the intrinsic privacy preferences of user u, while the second component $\boldsymbol{\vartheta}_z^0$ exploits the public's collective privacy preferences for a specific category of Apps (i.e., the wisdom of crowds) that captures the common patterns in permission requests for Apps with that specific category. Thus, an App with "unusual" permission requests (e.g., malwares) will enjoy low priority to be recommended.

3.4. Generative Process of Mobi-SAGE

Given the user profile $\mathcal{D}_u$ for a user u, the generative process of the Mobi-SAGE model for a downloaded App a in $\mathcal{D}_u$ is as follows.

- Draw a topic $z \sim P(z|\boldsymbol{\theta}^0, \boldsymbol{\theta}_u^{user})$
- For each textual word w in $\mathcal{T}_a$, draw $w \sim P(w|\boldsymbol{\phi}_z^0, \boldsymbol{\phi}_z^{topic})$
- For each visual word v in $\mathcal{V}_a$, draw $v \sim P(v|\boldsymbol{\psi}_z^0, \boldsymbol{\psi}_z^{topic})$
- Draw an App-ID $a \sim P(a|\boldsymbol{\varphi}_z^0, \boldsymbol{\varphi}_z^{topic})$
- For each permission request p in $\mathcal{P}_a$, draw $p \sim P(p|\boldsymbol{\vartheta}_z^0, \boldsymbol{\vartheta}_u^{user})$

For each downloaded App in $\mathcal{D}_u$, Mobi-SAGE first chooses the topic this App is about. To generate the topic index z, we utilize a multinomial model as follows:

$$P(z|\boldsymbol{\theta}^0, \boldsymbol{\theta}_u^{user}) = P(z|\boldsymbol{\theta}^0 + \boldsymbol{\theta}_u^{user}) \tag{3}$$

Here $\boldsymbol{\theta}^0$ and $\boldsymbol{\theta}_u^{user}$ are topic-based vectors that represent u's personal interests and the general public's interests. Once a topic index z is determined, the App a and its associated textual and visual words will be generated with Equations (4,5,6), respectively.

$$P(a|\boldsymbol{\varphi}_z^0, \boldsymbol{\varphi}_z^{topic}) = P(a|\boldsymbol{\varphi}_z^0 + \boldsymbol{\varphi}_z^{topic}) \tag{4}$$

$$P(w|\boldsymbol{\phi}_z^0, \boldsymbol{\phi}_z^{topic}) = P(w|\boldsymbol{\phi}_z^0 + \boldsymbol{\phi}_z^{topic}) \tag{5}$$

$$P(v|\boldsymbol{\psi}_z^0, \boldsymbol{\psi}_z^{topic}) = P(v|\boldsymbol{\psi}_z^0 + \boldsymbol{\psi}_z^{topic}) \tag{6}$$

Similarly, based on the sampled topic z, the permissions of App a are generated as follows:

$$P(p|\boldsymbol{\vartheta}_z^0, \boldsymbol{\vartheta}_u^{user}) = P(p|\boldsymbol{\vartheta}_z^0 + \boldsymbol{\vartheta}_u^{user}) \tag{7}$$

The above generative process applies to all user profiles in the dataset. The graphical representation of the generation process is shown in Figure 3. We introduce the following shorthands to simplify our notation:

$$
\begin{aligned}
&P(z|\boldsymbol{\theta}^0 + \boldsymbol{\theta}_u^{user}) = \zeta_{u,z},\ P(a|\boldsymbol{\varphi}_z^0 + \boldsymbol{\varphi}_z^{topic}) = \beta_{z,a} \\
&P(w|\boldsymbol{\phi}_z^0 + \boldsymbol{\phi}_z^{topic}) = \gamma_{z,w},\ P(v|\boldsymbol{\psi}_z^0 + \boldsymbol{\psi}_z^{topic}) = \xi_{z,v} \\
&P(p|\boldsymbol{\vartheta}_z^0 + \boldsymbol{\vartheta}_u^{user}) = \eta_{u,z,p}
\end{aligned}
$$

where the above probability distributions are computed as in Equation 8, from which we can see that each user and each topic in our model are endowed with a model of the deviation in log-frequency from a background distribution.

$$
\begin{aligned}
\zeta_{u,z} = \frac{exp(\theta_z^0 + \theta_{u,z}^{user})}{\sum_{z'} exp(\theta_{z'}^0 + \theta_{u,z'}^{user})}, \beta_{z,a} = \frac{exp(\varphi_a^0 + \varphi_{z,a}^{topic})}{\sum_{a'} exp(\varphi_{a'}^0 + \varphi_{z,a'}^{topic})}, \gamma_{z,w} = \frac{exp(\phi_w^0 + \phi_{z,w}^{topic})}{\sum_{w'} exp(\phi_{w'}^0 + \phi_{z,w'}^{topic})}, \\
\xi_{z,v} = \frac{exp(\psi_v^0 + \psi_{z,v}^{topic})}{\sum_{v'} exp(\psi_{v'}^0 + \psi_{z,v'}^{topic})}, \eta_{u,z,p} = \frac{exp(\vartheta_{z,p}^0 + \vartheta_{u,p}^{user})}{\sum_{p'} exp(\vartheta_{z,p'}^0 + \vartheta_{u,p'}^{user})}
\end{aligned}
\tag{8}
$$

$$
\begin{aligned}
P(\boldsymbol{z},\ \boldsymbol{a},\ \boldsymbol{w},\ \boldsymbol{v},\ \boldsymbol{p}|\ominus,\ \boldsymbol{u}) &= P(\boldsymbol{z}|\boldsymbol{u}, \boldsymbol{\theta}^0, \boldsymbol{\theta}^{user}) P(\boldsymbol{w}|\boldsymbol{z}, \boldsymbol{\phi}^0, \boldsymbol{\phi}^{topic}) P(\boldsymbol{v}|\boldsymbol{z}, \boldsymbol{\psi}^0, \boldsymbol{\psi}^{topic}) \\
&\quad P(\boldsymbol{a}|\boldsymbol{z}, \boldsymbol{\varphi}^0, \boldsymbol{\varphi}^{topic}) P(\boldsymbol{p}|\boldsymbol{u}, \boldsymbol{z}, \boldsymbol{\vartheta}^0, \boldsymbol{\vartheta}^{user}) \\
&= \prod_{u \in \mathcal{U}} \prod_{a \in \mathcal{D}_u} \{\zeta_{u,z_{u,a}} \beta_{z_{u,a},a} \prod_{w \in \mathcal{T}_a} \gamma_{z_{u,a},w} \prod_{v \in \mathcal{V}_a} \xi_{z_{u,a},v} \prod_{p \in \mathcal{P}_a} \eta_{u,z_{u,a},p}\} \\
&= \prod_{u \in \mathcal{U}} \prod_{z \in \mathcal{K}} \{\zeta_{u,z}^{n(u,z)} \prod_{p \in \mathcal{P}} \eta_{u,z,p}^{n(u,z,p)}\} \prod_{z \in \mathcal{K}} \{\prod_{a \in \mathcal{A}} \beta_{z,a}^{n(z,a)} \prod_{w \in \mathcal{W}} \gamma_{z,w}^{n(z,w)} \prod_{v \in \mathcal{V}} \xi_{z,v}^{n(z,v)}\}
\end{aligned}
\tag{9}
$$

3.5. Sparse Modeling

Sparsity of the representations is desirable in user interest modeling and topic representation. For users, very often it makes intuitive sense to assume that each user has a few salient topical interests rather than letting every topic make a non-zero contribution; this is important in practice for large scale user modeling en-

deavors such as those undertaken in Facebook or Amazon, where it needs to learn hundreds or thousands of topics for hundreds of millions of users - without an explicit sparcification procedure, it would be extremely challenging to nail down the interests of a user. Likewise, a word has only a few salient topical senses, and the discovered topics are expected to be discriminative, rather than obtaining redundant words in different topics. For instance, the word "car" might be more prevalent in the "transportation" and the "game" topics. However, we do not expect it to be prevalent in a large number of topics beyond what a background language model would indicate. In order to achieve the above goals, we need to impose sparsity-inducing prior distributions over certain parts of our model. More specifically, for $\boldsymbol{\theta}_u^{user}$, $\boldsymbol{\phi}_z^{topic}$, $\boldsymbol{\psi}_z^{topic}$, $\boldsymbol{\varphi}_z^{topic}$, $\boldsymbol{\vartheta}_z^0$ and $\boldsymbol{\vartheta}_u^{user}$, we impose zero-mean Laplace distributions as their prior distributions. Actually, a zero-mean Laplace prior has the same effect as placing an $L1$ regularizer on these model parameters, resulting in a sparse solution to the model. Therefore, we employ $L1$ regularization in the following model optimization.

3.6. Model Inference

The goal function of our model is given in Equation 9. The goal of model inference is to learn parameters that maximize the marginal log-likelihood of the observed variables **a**, **w**, **v** and **p**. Similar to the idea of many latent models, we introduce the latent random variable **z** in performing the marginalization. As Equation 9 is hard to be maximized directly, we adopt Gibbs EM (i.e., expectation maximization) algorithm [31], which is a mixture between a Monte Carlo sampler and EM, to learn the parameters maximizing Equation 9. In Equation 9, $\ominus$ is the set of all the parameters, and $z_{u,a}$ represents the topic of App a in $\mathcal{D}_u$. In E-step, we sample topic assignments by fixing all other parameters using Gibbs sampling. In

M-step, we optimize model parameters $\ominus$ by fixing all topic assignments. These two steps are iterated until convergence.

More specifically, in the E step, for each downloaded App (a, $\mathcal{T}_a$, $\mathcal{V}_a$, $\mathcal{P}_a$) in $\mathcal{D}_u$, we sample latent topic z according to the following posterior probability:

$$\begin{aligned} & P(z_{u,a} = z|\mathbf{z}_{\neg},\ \mathbf{a},\ \mathbf{w},\ \mathbf{v},\ \mathbf{p}, \ominus) \\ \propto\ & \zeta_{u,z} \times \beta_{z,a} \times \prod_{w \in \mathcal{T}_a} \gamma_{z,w} \times \prod_{v \in \mathcal{V}_a} \xi_{z,v} \times \prod_{p \in \mathcal{P}_a} \eta_{u,z,p} \end{aligned} \tag{10}$$

where $\mathbf{z}_{\neg}$ represents topic assignments for all downloading records except the current one, we assume all other variables are fixed.

In the M-step, the parameters $\ominus$ are optimized to maximize the log likelihood of the goal function by fixing all topic assignments. We enforce the inferred parameters (e.g., $\boldsymbol{\theta}_u^{user}$, $\boldsymbol{\phi}_z^{topic}$, $\boldsymbol{\psi}_z^{topic}$, $\boldsymbol{\varphi}_z^{topic}$, ϑ_z^0 and ϑ_u^{user}) to be L1-regularized, inspired by the fact that L1-regularization has the nice property of enforcing the inferred representations to be sparse. PSSG (Projected Scaled Sub-Gradient) [32], a popular gradient descent learning algorithm designed to optimize the problems with L1 regularization on parameters, is adopted in this step to update $\ominus$. Another advantage of PSSG is being scalable, because it adopts quasi-Newton strategy with line search, which is robust to common functions. More specifically, the limited-memory BFGS [33] updates for the quasi-Newton method are adopted in our model. The gradients of $\boldsymbol{\theta}_z^0$ and $\boldsymbol{\theta}_{u,z}^{user}$ are provided as follows.

$$\frac{\partial L}{\partial \theta_z^0} = \sum_{u \in \mathcal{U}} n(u,z) - \sum_{u \in \mathcal{U}} (n(u) \times \zeta_{u,z}) \tag{11}$$

$$\frac{\partial L}{\partial \theta_{u,z}^{user}} = n(u,z) - n(u) \times \zeta_{u,z} \tag{12}$$

where $n(u,z)$ represents the number of Apps assigned with topic z in $\mathcal{D}_u$, and $n(u)$ denotes the total number of Apps in $\mathcal{D}_u$, i.e., $n(u) = |\mathcal{D}_u| = D_u$.

Similarly, the gradients of model parameters $\boldsymbol{\phi}_w^0$, $\boldsymbol{\phi}_{z,w}^{topic}$, $\boldsymbol{\psi}_v^0$, $\boldsymbol{\psi}_{z,v}^{topic}$,$\boldsymbol{\varphi}_a^0$ and $\boldsymbol{\varphi}_{z,a}^{topic}$ are computed as follows:

$$\frac{\partial L}{\partial \phi_w^0} = n(w) - \sum_{z \in \mathcal{K}} n_T(z) \times \gamma_{z,w} \tag{13}$$

$$\frac{\partial L}{\partial \phi_{z,w}^{topic}} = n(z,w) - n_T(z) \times \gamma_{z,w} \tag{14}$$

$$\frac{\partial L}{\partial \psi_v^0} = n(v) - \sum_{z \in \mathcal{K}} n_V(z) \times \xi_{z,v} \tag{15}$$

$$\frac{\partial L}{\partial \psi_{z,v}^{topic}} = n(z,v) - n_V(z) \times \xi_{z,v} \tag{16}$$

$$\frac{\partial L}{\partial \varphi_a^0} = n(a) - \sum_{z \in \mathcal{K}} n_A(z) \times \beta_{z,a} \tag{17}$$

$$\frac{\partial L}{\partial \varphi_{z,a}^{topic}} = n(z,a) - n_A(z) \times \beta_{z,a} \tag{18}$$

where $n(z,w)$ and $n(z,v)$ are the numbers of times that textual word w and visual word v are assigned to topic z, respectively, and $n_T(z)$ and $n_V(z)$ are the total numbers of textual and visual words assigned to topic z, respectively, i.e., $n_T(z) = \sum_{w' \in \mathcal{W}} n(z,w')$ and $n_V(z) = \sum_{v' \in \mathcal{V}} n(z,v')$. $n(z,a)$ is the number of times that App a is assigned to topic z, and $n_A(z)$ is the total number of Apps assigned to topic z, i.e., $n_A(z) = \sum_{a' \in \mathcal{A}} n(z,a')$. These gradients have an intuitive interpretation as the difference of the true counts and their expected counts.

The gradients of model parameters $\vartheta_{z,p}^0$ and $\vartheta_{u,p}^{user}$ are computed as follows:

$$\frac{\partial L}{\partial \vartheta_{z,p}^0} = \sum_{u \in \mathcal{U}} n(u,z,p) - \sum_{u \in \mathcal{U}} n_P(u,z) \times \eta_{u,z,p} \tag{19}$$

$$\frac{\partial L}{\partial \vartheta_{u,p}^{user}} = \sum_{z \in \mathcal{K}} n(u,z,p) - \sum_{z \in \mathcal{K}} n_P(u,z) \times \eta_{u,z,p} \tag{20}$$

where $n(u,z,p)$ is the number of times that permission request p is assigned to topic z in the $\mathcal{D}_u$, and $n_P(u,z)$ is the total number of permission requests assigned

to topic z in $\mathcal{D}_u$, i.e., $n_P(u,z) = \sum_{p \in \mathcal{P}} n(u,z,p)$. Note that $n_P(u,z)$ and $n(u,z)$ are not equal, as an App may have multiple permission requests.

Time Complexity Analysis. There are two main components in it: Gibbs sampling in the E-step and gradient descent learning in the M-step. We assume that the inference algorithm needs I iterations to reach convergence. For each iteration, its time complexity is analyzed as follows. In the E-step, it needs to go through all user downloading records, and for each rated or downloaded App it requires $O(K)$ operations to compute the posterior probability distribution for sampling topic z. K is the number of latent topics that is also equal to the number of categories defined in the App markets. Thus, the time complexity in this step is $O(K \times D)$ where D=$\sum_u D_u$ is the total number of downloading records in the dataset. In the M-step, we use the gradient descent learning algorithm PSSG to update the model parameters $\ominus = \{\boldsymbol{\theta}^0, \boldsymbol{\theta}^{user}, \boldsymbol{\vartheta}^0, \boldsymbol{\vartheta}^{user}, \boldsymbol{\phi}^0, \boldsymbol{\phi}^{topic}, \boldsymbol{\psi}^0, \boldsymbol{\psi}^{topic}, \boldsymbol{\varphi}^0, \boldsymbol{\varphi}^{topic}\}$ based on the topic assignments sampled in the E-step. We assume that the gradient descent algorithm needs J iterations to converge, and in each iteration the time complexity to compute the gradients for the model parameters $\ominus$ is $O(K \times D)$. Thus, the time complexity in the M-step is $O(J \times K \times D)$. The total time complexity for each EM iteration is $O((J+1) \times K \times D)$=$O(J \times K \times D)$, which indicates that the computational time is linear with respect to the number of downloading records in our dataset. Mobi-SAGE achieves fast convergence rates in M-step and needs only a few iterations (i.e., $J < 20$) to converge in our dataset.

3.7. Efficient Recommendation

Once we have trained the model Mobi-SAGE, given a target user u, we compute the probability for each unrated App a, as in Equation 21, and then select the

top-k ones with highest probabilities as recommendations.

$$\begin{aligned}
&P(a, \mathcal{T}_a, \mathcal{V}_a, \mathcal{P}_a | u, \ominus) \\
=&\sum_{z \in \mathcal{K}} P(a, z, \mathcal{T}_a, \mathcal{V}_a, \mathcal{P}_a | u, \ominus) \\
=&\sum_{z \in \mathcal{K}} P(z|\boldsymbol{\theta}^0, \boldsymbol{\theta}_u^{user}) P(a|\boldsymbol{\varphi}^0, \boldsymbol{\varphi}_z^{topic}) P(\mathcal{T}_a|\boldsymbol{\phi}^0, \boldsymbol{\phi}_z^{topic}) \\
&\times P(\mathcal{V}_a|\boldsymbol{\psi}^0, \boldsymbol{\psi}_z^{topic}) P(\mathcal{P}_a|\boldsymbol{\vartheta}_z^0, \boldsymbol{\vartheta}_u^{user}) \\
=&\sum_{z \in \mathcal{K}} \zeta_{u,z} \beta_{z,a} \Big(\prod_{w \in \mathcal{T}_a} \gamma_{z,w} \Big)^{\frac{1}{T_a}} \Big(\prod_{v \in \mathcal{V}_a} \xi_{z,v} \Big)^{\frac{1}{V_a}} \Big(\prod_{p \in \mathcal{P}_a} \eta_{u,z,p} \Big)^{\frac{1}{P_a}}
\end{aligned} \tag{21}$$

where we adopt geometric mean for the probability of topic z generating $\mathcal{T}_a$, $\mathcal{V}_a$ and $\mathcal{P}_a$, considering that different Apps may have different numbers of textual words, visual words and permission requests. The above equation can be rewritten as follows:

$$\begin{aligned}
S(u, a) &= \sum_{z \in \mathcal{K}} W(u, z) F(a, z) \\
W(u, z) &= \zeta_{u,z} \times \Big(\prod_{p \in \mathcal{P}_a} \eta_{u,z,p} \Big)^{\frac{1}{P_a}} \\
F(a, z) &= \beta_{z,a} \Big(\prod_{w \in \mathcal{T}_a} \gamma_{z,w} \Big)^{\frac{1}{T_a}} \Big(\prod_{v \in \mathcal{V}_a} \xi_{z,v} \Big)^{\frac{1}{V_a}}
\end{aligned} \tag{22}$$

where $S(u,\ a)$ represents the ranking score of App a for user u. Each latent topic z can be seen as a functionality, and $W(u, z)$ represents the preference weight of user u on functionality z, and $F(a, z)$ represents the score of App a with respect to functionality z. This ranking framework separates the offline computation from the online computation. Since $F(a, z)$ is independent of the querying user u, it is computed offline. Although the query weight $W(u, z)$ is computed online, its main time-consuming components (i.e., $\zeta_{u,z}$ and $\eta_{u,z,p}$) are also computed offline, and the online computation is just a simple assembling process. This design enables maximum precomputation for the problem considered, and in turn minimizes the online recommendation time.

The straightforward method of generating top-k recommendations needs to compute the ranking scores for all Apps according to Equation 22 and select the k ones with highest ranking scores, which is, however, computationally inefficient, especially when the number of available Apps becomes large. To improve the online recommendation efficiency based on the observation of user interest sparsity that a user u only prefers a small number of topics (say 5-10 latent dimensions) and her/his weights on most dimensions are extremely small, we adopt the TA-based query processing technique developed in [26, 27]. Since $W(u, z)$ is non-negative, the proposed ranking function in Equation 22 is monotonically increasing given a user u, which meets the requirement of the TA-based query processing technique. This technology has the nice property of finding top-k results correctly by examining the minimum number of items without scanning all ones, which enables our recommender system scalable to large-scale App market datasets.

Addressing Cold-start Problem. As for cold-start users, our Mobi-SAGE model can still make App recommendations according to the "wisdom of crowds", i.e., general public' interests $\boldsymbol{\theta^0}$ that capture the global trends of App categories and the public's privacy preferences $\boldsymbol{\vartheta_z^0}$ that represent the common patterns in permission requests by Apps with functionality z. In this scenario, $\zeta_{u,z}$ and $\eta_{u,z,p}$ in $W(u, z)$ of Equation22 are degenerated to:

$$\zeta_{u,z} = \frac{exp(\theta_z^0)}{\sum_{z'} exp(\theta_{z'}^0)}, \ \ \eta_{u,z,p} = \frac{exp(\vartheta_{z,p}^0)}{\sum_{p'} exp(\vartheta_{z,p'}^0)}$$

Compared with the popularity-based recommendation method, the newly launched Apps with popular functionalities can enjoy an opportunity to be recommended by our approach.

As for cold-start Apps, our Mobi-SAGE model can still accurately recommend

Table 3: Characteristics of Different Methods.

Methods \ Features	Textual Content	Visual Content	User Security or Privacy Preferences	Category-Aware User Privacy Preferences	Personal Interests
Mobi-SAGE	•	•	•	•	•
LIBFM	•	•	•		•
SPAR			•		
PBPR			•		•
BPR					•
Mobi-SAGE-V1		•	•	•	•
Mobi-SAGE-V2	•	•	•		•
Mobi-SAGE-V3	•		•	•	•
Mobi-SAGE-V4	•	•			•

them to the right users according to their textual and visual contents and permission requests. Specifically, $F(a, z)$ in Equation 22 is reformulated as follows.

$$F(a,z) = \Big(\prod_{w \in \mathcal{T}_a} \gamma_{z,w} \Big)^{\frac{1}{T_a}} \Big(\prod_{v \in \mathcal{V}_a} \xi_{z,v} \Big)^{\frac{1}{V_a}}$$

4. Experiments

In this section, we first describe the setup of experiments and then demonstrate the evaluation results.

4.1. Dataset

Our data comes from a leading Android app store in China, called 360 Mobile Assistant[4]. The 360 platform has over 275 million active users, growing at a rapid rate every month. We randomly sampled 100000 users and collected their App download records in the recent one year. Totally, there are 50217 Apps and 10,203,755 download records in our dataset. For each App, we collected its name,

[4] http://zhushou.360.cn/

description, screenshots and permissions. The user-App download matrix has a sparsity as high as 99.80%. Different from the Google Play dataset collected by [17], our dataset is an implicit feedback dataset.

4.2. Comparison Approaches

We compare our proposed Mobi-SAGE model with the following two state-of-the-art App recommender models and two powerful conventional latent factor models. The characteristics of all the recommendation methods are listed in Table3.

SPAR. SPAR [16] is the first security-aware mobile App recommendation approach. It first computed a security score for each App by a regularization approach, and then used a modern portfolio theory based method to rank Apps by striking a balance between the Apps' popularity and their security scores. This method does not consider users' preferences, thus it is a non-personalized recommendation approach.

LIBFM. LIBFM [34] is a state-of-the-art feature based factor model library. Based on it, we build a factorization model by incorporating all side information of Apps including textual, visual and permissions with the collaborative filtering.

BPR. BPR [5] is the state-of-the-art of personalized ranking model for implicit feedback datasets. Compared with the point-wise matrix factorization methods that focus on the prediction of user ratings, BPR is a pairwise matrix factorization method that directly optimizes the ranking of the feedback and is more suitable for top-k recommendation.

PBPR. Based on BPR, we implement a stronger baseline, the privacy-aware BPR, to integrate user privacy preferences with user interests, following the method developed in [17]. Note that we do not directly compare with the App recom-

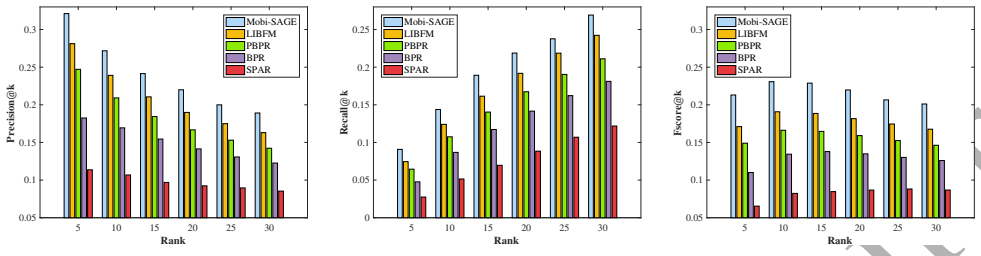


Figure 4: Recommendation Effectiveness on 360 Mobile App Dataset.

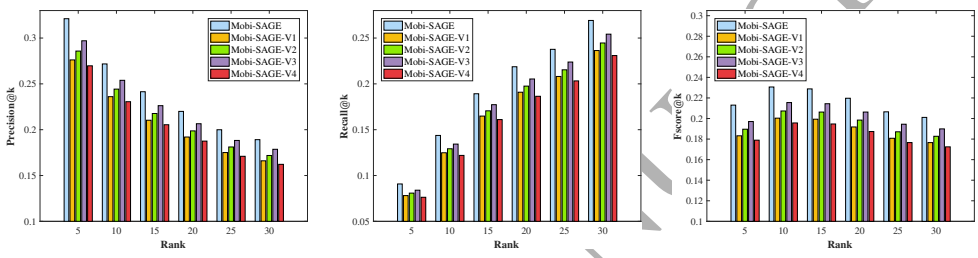


Figure 5: Impact of Different Features on Recommendation Effectiveness.

mender model developed by Liu et al. [17], since they adopted a Poisson Factor model that was designed for user rating dataset while our dataset only contains implicit feedback.

To further validate the benefits brought by integrating textual content, visual content, user privacy preferences and category-aware user privacy preferences, respectively, we design four variants of Mobi-SAGE, as shown in Table 3. Compared with Mobi-SAGE, **Mobi-SAGE-V1** and **Mobi-SAGE-V3** ignore the textual content and visual content, respectively; **Mobi-SAGE-V2** ignores that user privacy preferences depend on App categories; **Mobi-SAGE-V4** does not consider the factor of user privacy preferences at all.

4.3. Evaluation methods

Dataset Splitting. Given a user profile $\mathcal{D}_u$, namely a collection of downloading records, we first rank the records according to their downloading timestamps. Then, we use the 70-th and 80-th percentiles as the cut-off points so that the first 70% records are used for training, and the last 20% are marked off as the testing data. The remaining 10% are used as the validation data to tune the model hyper-parameters such as the numbers of topics or the dimension of latent factors. According to the above dividing strategies, we split the dataset $\mathcal{D}$ into the training set $\mathcal{D}^{training}$, the validation set $\mathcal{D}^{validation}$ and the test set $\mathcal{D}^{test}$.

Evaluation Metrics. In real mobile App recommendation scenarios, our task is to recommend a top-ranked list of Apps to the target user. Thus, in this section, we evaluate the effectiveness of proposed recommender models from the aspect of personalized ranking. More specifically, each target user in the test set is presented with k Apps having the highest ranking scores computed by Equation 21, but have not been rated or downloaded by the target user in the training set. Then, the performance of different recommendation methods are evaluated based on whether the recommended Apps will be downloaded by the target user in the test set. Given a target user u in the test set, if a recommended App is in the test set $\mathcal{D}_u^{test}$, we assume that u will download it and we call this case as a "hit", otherwise it's a "miss". To make the experimental results comparable, we use multiple well-known metrics in information retrieval to measure the ranked results. We first use Precision@k and Recall@k to assess the quality of the top-k recommended Apps as follows:

$$Precision@k = \frac{\#hits}{k}, \;\; Recall@k = \frac{\#hits}{D_u^{test}}$$

where $\#hits$ is the number of "hit" Apps in the top-k recommended ones and

$D_u^{test} = |\mathcal{D}_u^{test}|$. The precision and recall for the entire dataset are computed by averaging the precision and recall over all the users, respectively. Besides, F-measure is employed to balance between precision and recall. We consider the F_β metric, and it is defined as:

$$F_\beta = (1+\beta^2)\frac{Precision \times Recall}{\beta^2 \cdot Precision + Recall}.$$

The F_β metric with $\beta < 1$ indicates more emphasis on precision than recall. In our experiment, we use F_β metric with $\beta = 0.5$ since we are more concerned about "Precision" in the recommendation scenario, following [17].

4.4. Recommendation Effectiveness

In this part, we present the effectiveness of the recommendation methods with well-tuned parameters. Figure 4 reports the comparison results between our proposed model Mobi-SAGE and other competitor methods in terms of Precision@k, Recall@k and Fscore@k. All differences between our model Mobi-SAGE and the other comparison methods are statistically significant ($p < 0.01$).

Clearly, our proposed Mobi-SAGE model significantly outperforms other competitor models consistently in all the three metrics, and the relative improvements, in terms of Fscore@10, are 20.9%, 38.8%, 71.5% and 180% compared with LIBFM, PBPR, BPR and SPAR, respectively. Several observations are also made from the results: (1) PBPR achieves higher recommendation accuracy than BPR, showing the benefit brought by considering user privacy preferences. (2) Both Mobi-SAGE and LIBFM outperform PBPR. It might be because Mobi-SAGE and LIBFM leverage visual and textual contents of Apps to alleviate the issue of data sparsity. (3) Our Mobi-SAGE beats LIBFM due to that LIBFM cannot model and capture the subtle dependence between App functionalities and user

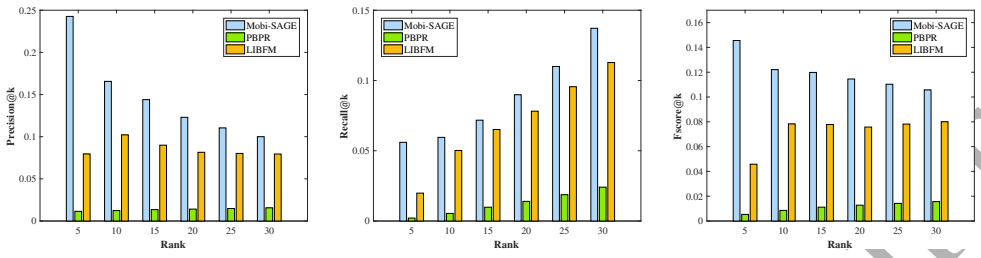


Figure 6: Recommendation Effectiveness for Cold-Start Apps.

privacy preferences. Another possible reason is that LIBFM is a general-purpose factor model that utilizes auxiliary information in a feature-engineering way and treats each feature equally, while our Mobi-SAGE is able to treat each App feature in a more reasonable way and thus improve the recommendation quality. (4) SPAR drops behind all other models, as it is a non-personalized recommendation method and tends to recommend the most popular Apps for each user, ignoring the uniqueness of user interests and tastes [35].

4.5. Impact of Different Factors

Impact of Features. In this subsection, we first carry out an ablation study showing the benefits of each feature proposed in Mobi-SAGE, i.e., App textual feature (V1), relation between use privacy preferences and App categories (V2), App visual feature (V3) and user privacy preferences (V4). We compare Mobi-SAGE with its four variant versions proposed in Section 4.2, and the comparison results are shown in Figure 5. From the results, we first observe that Mobi-SAGE consistently outperforms the four variant versions in mobile App recommendation, indicating the benefits brought by each factor, respectively. For instance, the performance gap between Mobi-SAGE and Mobi-SAGE-V2 validates the benefit of exploiting and integrating the dependence of user privacy preferences on App

functionalities or categories. Second, we find that the contribution of each feature to improving App recommendation is different. Specifically, according to the importance of the four features in the App recommendation, they can be ranked as follows: $V4 > V1 > V2 > V3$. Obviously, leveraging user privacy preferences from App permission requests is most important to improve App recommendation. Another observation is that the textual features of Apps are more powerful than the visual features. In this experiment, we only study one type of visual features, i.e., the visual words extracted by SIFT, and we will explore more advanced visual features - CNN-based visual words [36] in our future work.

Impact of Model Parameters. There is one important hyper-parameter in our Mobi-SAGE model: the number of topics (i.e., K). According to our previous experiences in topic models [26, 27], tuning the number of topics is critical to the model performance. Thus, we also study the impact of the parameter K (i.e., the number of topics), and present the results in Table 4. Due to space constraints, we have only shown the experimental results for the top-10 recommendation, and similar observations are also made for other k values. We tested the performance of Mobi-SAGE by varying the number of topics from 40 to 140. From the results, we observe that the performance first improves quickly with the increase of the number of topics and then the increment becomes small. The number of the topics represents the model complexity. Thus, when K is too small, the model has limited ability to describe the data. However, when K exceeds a threshold (e.g., $K = 100$), the model is complex enough to handle the App market data. At this point, it is less helpful to improve the model performance by increasing K, but will increase both model training and online recommendation time cost. As a result, we choose the number of topics $K = 100$ as the best trade-off between

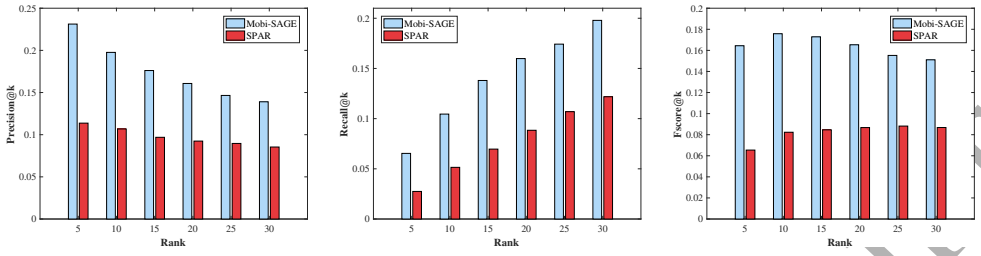


Figure 7: Recommendation Effectiveness for Cold-Start Users.

effectiveness and efficiency.

Table 4: The Effectiveness of Top-10 Recommendation.

	K=40	K=60	K=80	K=100	K=120	K=140
P@10	0.239	0.250	0.263	0.271	0.271	0.272
R@10	0.126	0.132	0.139	0.144	0.144	0.144
F@10	0.203	0.212	0.224	0.231	0.231	0.232

4.6. Test for Cold-Start Recommendation

We further conduct experiments to study the effectiveness of different recommendation algorithms handling the cold start problem. We first test the recommendation effectiveness for cold-start Apps on the 360 mobile App dataset and present the results in Figure 6. As BPR is a pure collaborative filtering method and SPAR is a popularity-based method, they cannot apply to cold-start App recommendation, and we do not compare with them. To evaluate the performance of our Mobi-SAEG model, we first choose those Apps that are downloaded by less than 5 users as the cold-start Apps. Then, we select users who have downloaded at least one cold-start Apps as test users. For each test user, we mark off her/his download records associated with cold-start Apps as the test data, and the

remaining download records as the training data.

From the results, we have the following observations: 1) our proposed Mobi-SAGE performs best consistently in recommending cold-start Apps; and 2) compared with the results in Figure 4, the recommendation effectiveness of all methods decreases, to different degrees, for cold-start Apps, e.g., the recommendation effectiveness of PBPR drops drastically while our Mobi-SAGE decreases slightly; 3) both Mobi-SAGE and LIBFM perform significantly better than PBPR, which demonstrates that the recommendation methods that incorporate multiple types of auxiliary information perform significant better than the one that only leverages a single type of features. This is because cold-start Apps lack interaction information which is the essential foundation of the BPR-based methods. The superior performance of Mobi-SAGE models in recommending cold-start Apps shows that exploiting and integrating user privacy preferences, textual and visual features can effectively alleviate cold-start problem.

We also test the performance of our Mobi-SAGE in recommending Apps to cold-start users. We first randomly select 10000 users as the test users. For each test user, we remove their download records as the test data. As LIBFM, PBPR and BPR cannot apply to cold-start users, we only compare Mobi-SAGE with SPAR in this experiment. Figure 7 reports the performance of Mobi-SAGE and SPAR for cold start users on the 360 mobile App dataset. By comparing Figure 7 and Figure 4, we observe that 1) the performance of SPAR does not change in these two figures, as it is a non-personalized recommendation method; and 2) although the recommendation effectiveness of our Mobi-SAGE decreases, to some extent, for cold start users, it still outperforms SPAR significantly. Compared with the popularity-based SPAR, our Mobi-SAGE captures the global trends of

App categories by latent topics rather than the popularity of specific Apps, thus the newly launched Apps with few downloads but popular functionalities can still be recommended to the new users by our approach.

4.7. Recommendation Efficiency

This experiment is to evaluate the online recommendation efficiency. For the online recommendation of Mobi-SAGE, we adopt two methods. The first one is called Mobi-SAGE-TA which adopts the TA-based query processing technique [26] to produce online recommendation. The second is called Mobi-SAGE-BF which uses a naive brute-force algorithm to produce top-k recommendations by computing a score for each App. As the latent vectors learnt in LIBFM, PBPR and BPR can be negative, their ranking functions are not monotonic. The TA-based query processing technique cannot be applied to them, and these three methods use brute-force algorithm to produce top-k recommendations. All the online recommendation methods were implemented in Java (JDK 1.8) and run on a Windows Server with 200G RAM.

Table 5 presents the average online efficiency of five different methods over all users in $\mathcal{D}^{test}$ on the 360 App dataset. We show the performance where k is set to 5, 10, 15 and 20. For example, on average Mobi-SAGE-TA finds the top-10 recommendations from about 50,000 Apps in 28.45 ms. From the results, we observe that 1) Mobi-SAGE-TA outperforms Mobi-SAGE-BF significantly, justifying the benefits brought by the TA-based query processing technique, and it only needs to access around 12% of all the Apps on average for top-10 recommendations; 2) the time cost of Mobi-SAGE-TA increases with the increasing number of recommendations (i.e., k), but it is still much lower than all comparison methods; 3) BPR is faster than other brute-force based recommendation methods, because it does not

consider any auxiliary information of Apps in computing the ranking score; and 4) Mobi-SAGE-BF is more time-consuming than LIBFM, as Mobi-SAGE-BF involves more multiplication operations than LIBFM when computing the ranking score.

Table 5: Recommendation Efficiency on 360 App Dataset.

Methods	Online Recommendation Time Cost (ms)			
	k=5	k=10	k=15	k=20
Mobi-SAGE-TA	10.91	28.45	37.65	46.28
Mobi-SAGE-BF	195.34	195.75	195.89	195.94
LIBFM	180.31	180.33	180.42	180.48
PBPR	159.43	159.48	159.51	159.57
BPR	143.12	143.26	143.38	143.42

5. Related Work

In this section, we group related studies into two categories, and survey the literature of each category in detail.

5.1. Application Markets Mining and Analysis

App markets contain a wealth of multi-modal data associated with Apps, which is potentially useful for different App ecosystem stakeholder. Recently, there are emerging studies on mining App markets data to facilitate various applications, such as App review mining and analysis [37], automatic App tagging [38], detecting similar Apps [39], App search [40], and so on. Although the nature of the App data used in the above studies is similar to ours, the techniques and research goals are very different. Our work aims to use the knowledge discovered from

App market data to automatically and accurately recommend Apps to the right mobile users.

5.2. Mobile Application Recommendation

5.2.1. Privacy-Insensitive Recommendation

More recently, mobile App recommendation has begun to attract attention from both data mining and information retrieval communities, but most of existing works focused on recommending the most relevant Apps to a user without considering the influence of user privacy preferences.

AppJoy proposed in [41] measured how the applications are actually used, and the usage scores were then used by a collaborative filter algorithm to make personalized recommendation, inspired by the fact that whether a user has installed an application is only a weak indicator of whether she actually likes that application. AppAware [42] allowed its users to see what applications are being installed right now or around their position by other people, thus introducing a new way of promoting the mobile applications. Viljanac et al. [43] focused on context-based recommendations of mobile apps. [43] demonstrated how the constantly changing context and the constantly changing preferences influence the recommendations for users. LSA captured not only the needs of the users, but also the expectations of the developers and the online market [44]. Torres-Carazo et al. made an initial exploration in taking the people with special needs, like the disabled users, into consideration during the mobile App recommendation in [45]. Souza et al. presented a study of open source recommender systems and their usefulness for SoliMobile Project, which aimed to design, build and implement a package of innovative services focused on the individual in unstable situation (unemployment, homeless, etc.) in [46]. [47] aimed to identify and analyze mobile application

recommender systems developed for the health area.

Lin et al. [18] proposed to integrate side information from Twitter to address cold-start App problem. Specifically, the followers of the App's official Twitter account are collected and treated as "virtual words", then a latent Dirichlet allocation model is employed to generate latent groups. At recommendation time, a target user seeking recommendations is mapped to these latent groups. By using the transitive relationship of latent groups to Apps, they estimated the probability of the user liking the app. However, only a small portion of Apps in the real-life App markets have an official Twitter account, and most of these Apps tend to be popular ones rather than cold-start ones. Yin et al. [48] considered a trade-off between satisfaction and temptation for App recommendation with a special focus on the case that a user like to replace an old App with a new one. An interesting dataset is collected via an App from mobile users to study the trade-off between satisfaction and temptation, and the analyzed results reveal the users' decision process of choosing a new App after comparing it with those already obtained ones, which is then used to facilitate the recommendation algorithm. Lin et al. [49] distinguished the same Apps with different versions and incorporated features distilled from version descriptions into App recommendation. Specifically, they first used a topic model to construct a representation of an App's version as a set of latent topics discovered from version metadata and textual descriptions. They then discriminated the topics based on genre information and weighted them on a per-user basis to generate a version-sensitive ranked list of Apps.

5.2.2. Privacy-Sensitive Recommendation

To the best of our knowledge, there are only three works [17, 20, 16] that considered the privacy-sensitive permission requests or security.

Specifically, Zhu et al. [16] proposed a mobile App recommender system by considering both the App's popularity and security risks. But, they provided an identical global ranking of Apps to each user, and thus their work is not personalized App recommendation. Liu et al. [17] extended the probabilistic matrix factorization model for personalized App recommendation by considering both user interests and user privacy preferences. Compared with our work, they ignored the dependence between user privacy preferences and the App functionalities, i.e., they assumed that the user has the same privacy preferences toward all Apps even with different functionalities. Moreover, they only used the users' rating information, and did not explore or exploit the potential of the rich multi-modal data associated with Apps to improve App recommendation, especially for the cold-start Apps and users. In their experiments, they removed unpopular users and Apps (i.e., Apps with less than 5 users and users with less than 10 Apss) to avoid data sparsity and cold start problem. Although they reported the good performance of their approach on the carefully chosen testing data, their experimental setup is quite different from the real-life App recommendation scenario. [20] is an abstract paper which presents our preliminary study of mobile app recommendation. In this article, we make significant revision and add substantial new materials compared with the work presented in [20]. For example, we impose sparse coding on user interest modeling and topic representation by introducing zero-mean Laplace distributions as prior distributions.

6. Conclusions

In this paper, we proposed a mobile sparse additive generative model (MobiSAGE) to recommend Apps by considering both user interests and functionality-

aware user privacy preferences. To alleviate the data sparsity issue, we exploited both textual and visual content associated with Apps to learn multi-view topics for user interest modeling. Besides, we also leveraged the "wisdom of crowd" including both the general public's interests and functionality-aware privacy preferences to produce recommendations for cold-start users. To evaluate the performance of our proposed recommender model Mobi-SAGE, extensive experiments were conducted on a large App dataset. The experimental results revealed that our Mobi-SAGE consistently and significantly outperforms the state-of-the-art approaches, which implies the importance of exploiting functionality-aware user privacy preferences and the multi-modal App content data on personalized App recommendation.

7. Acknowledgement

The work described in this paper is partially supported by ARC Discovery Early Career Researcher Award(DE160100308), ARC Discovery Project (DP170103954) and the National Natural Science Foundation of China (61572335).

[1] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, KBS 46 (2013) 109–132.

[2] A. L. V. Pereira, E. R. Hruschka, Simultaneous co-clustering and learning to address the cold start problem in recommender systems, KBS 82 (2015) 11–19.

[3] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma, X. Wang, Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems, KBS 138 (2017) 202–207.

[4] M. Y. H. Al-Shamri, User profiling approaches for demographic recommender systems, KBS 100 (2016) 175–187.

[5] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, Bpr: Bayesian personalized ranking from implicit feedback, in: UAI, 2009, pp. 452–461.

[6] R. Salakhutdinov, A. Mnih, Bayesian probabilistic matrix factorization using markov chain monte carlo, in: ICML, 2008, pp. 880–887.

[7] A. Mnih, R. Salakhutdinov, Probabilistic matrix factorization, in: NIPS, 2007, pp. 1257–1264.

[8] L. Qiu, S. Gao, W. Cheng, J. Guo, Aspect-based latent factor model by integrating ratings and reviews for recommender system, KBS 110 (2016) 233–243.

[9] H. Yin, B. Cui, L. Chen, Z. Hu, X. Zhou, Dynamic user modeling in social media systems, ACM Trans. Inf. Syst. 33 (3) (2015) 10:1–10:44.

[10] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37.

[11] Q. Gu, J. Zhou, C. H. Ding, Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs., in: SDM, 2010, pp. 199–210.

[12] R. M. Bell, Y. Koren, Lessons from the netflix prize challenge, SIGKDD Explor. Newsl. 9 (2) (2007) 75–79.

[13] N. Aizenberg, Y. Koren, O. Somekh, Build your own music recommender by modeling internet radio streams, in: WWW, 2012, pp. 1–10.

[14] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, Y. Rui, Geomf: Joint geographical modeling and matrix factorization for point-of-interest recommendation, in: KDD, 2014, pp. 831–840.

[15] B. Liu, Y. Fu, Z. Yao, H. Xiong, Learning geographical preferences for point-of-interest recommendation, in: KDD, 2013, pp. 1043–1051.

[16] H. Zhu, H. Xiong, Y. Ge, E. Chen, Mobile app recommendations with security and privacy awareness, in: KDD, 2014, pp. 951–960.

[17] B. Liu, D. Kong, L. Cen, N. Z. Gong, H. Jin, H. Xiong, Personalized mobile app recommendation: Reconciling app functionality and user privacy preference, in: WSDM, 2015, pp. 315–324.

[18] J. Lin, K. Sugiyama, M.-Y. Kan, T.-S. Chua, Addressing cold-start in app recommendation: Latent user models constructed from twitter followers, in: SIGIR, 2013, pp. 283–292.

[19] W. Wang, H. Yin, L. Chen, Y. Sun, S. Sadiq, X. Zhou, Geo-sage: A geographical sparse additive generative model for spatial item recommendation, in: KDD, 2015, pp. 1255–1264.

[20] H. Yin, L. Chen, W. Wang, X. Du, N. Q. V. Hung, X. Zhou, Mobi-sage: A sparse additive generative model for mobile app recommendation, in: ICDE, 2017, pp. 75–78.

[21] S. Shekhar, M. Dietz, D. S. Wallach, Adsplit: Separating smartphone advertising from applications, in: Security, 2012, pp. 28–28.

[22] A. P. Felt, E. Chin, S. Hanna, D. Song, D. Wagner, Android permissions demystified, in: CCS, 2011, pp. 627–638.

[23] D. G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vision 60 (2) (2004) 91–110.

[24] J. Eisenstein, A. Ahmed, E. P. Xing, Sparse additive generative models of text, in: ICML, 2011, pp. 1041–1048.

[25] W. Wang, H. Yin, S. W. Sadiq, L. Chen, M. Xie, X. Zhou, SPORE: A sequential personalized spatial item recommender system, in: ICDE, 2016, pp. 954–965.

[26] H. Yin, Y. Sun, B. Cui, Z. Hu, L. Chen, Lcars: A location-content-aware recommender system, in: KDD, 2013, pp. 221–229.

[27] H. Yin, B. Cui, L. Chen, Z. Hu, Z. Huang, A temporal context-aware model for user behavior modeling in social media systems, in: SIGMOD, 2014, pp. 1543–1554.

[28] B. Hu, M. Ester, Spatial topic modeling in online social media for location recommendation, in: RecSys, 2013, pp. 25–32.

[29] B. Liu, H. Xiong, Point-of-interest recommendation in location based social networks with topic and location awareness, in: SDM, 2013, pp. 396–404.

[30] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[31] H. M. Wallach, Topic modeling: Beyond bag-of-words, in: ICML, 2006, pp. 977–984.

[32] M. Schmidt, A. Niculescu-Mizil, K. Murphy, Learning graphical model structure using l1-regularization paths, in: AAAI, 2007, pp. 1278–1283.

[33] D. C. Liu, J. Nocedal, On the limited memory bfgs method for large scale optimization, Math. Program. 45 (3) (1989) 503–528.

[34] S. Rendle, Factorization machines with libFM, ACM Trans. Intell. Syst. Technol. 3 (3) (2012) 57:1–57:22.

[35] Y.-A. de Montjoye, L. Radaelli, V. K. Singh, A. ". Pentland, Unique in the shopping mall: On the reidentifiability of credit card metadata, Science 347 (6221) (2015) 536–539.

[36] L. Zheng, Y. Yang, Q. Tian, SIFT meets CNN: A decade survey of instance retrieval, CoRR abs/1608.01807.

[37] N. Chen, J. Lin, S. C. H. Hoi, X. Xiao, B. Zhang, Ar-miner: Mining informative reviews for developers from mobile app marketplace, in: ICSE, 2014, pp. 767–778.

[38] N. Chen, S. C. Hoi, S. Li, X. Xiao, Mobile app tagging, in: WSDM, 2016, pp. 63–72.

[39] N. Chen, S. C. Hoi, S. Li, X. Xiao, Simapp: A framework for detecting similar mobile applications by online kernel learning, in: WSDM, 2015, pp. 305–314.

[40] J. Zhuo, Z. Huang, Y. Liu, Z. Kang, X. Cao, M. Li, L. Jin, Semantic matching in app search, in: WSDM, 2015, pp. 209–210.

[41] B. Yan, G. Chen, Appjoy: personalized mobile application discovery, in: MobiSys, 2011, pp. 113–126.

[42] A. Girardello, F. Michahelles, Appaware: which mobile applications are hot?, in: HCI, 2010, pp. 431–434.

[43] V. Viljanac, Recommender system for mobile applications.

[44] X. Xia, X. Wang, X. Zhou, B. Liu, Evolving mobile app recommender systems: An incremental multi-objective approach, in: J. J. J. H. Park, I. Stojmenovic, M. Choi, F. Xhafa (Eds.), Future Information Technology, Springer Berlin Heidelberg, 2014, pp. 21–27.

[45] M. I. Torres-Carazo, M. J. Rodríguez-Fórtiz, M. V. Hurtado, J. Samos, V. Espín, Architecture of a mobile app recommender system for people with special needs, in: UCAMI, 2014, pp. 288–291.

[46] R. G. D. Souza, R. Chiky, Z. Kazi-Aoul, Open source recommendation systems for mobile application, in: PRSAT, 2010, pp. 55–58.

[47] L. R. Ferretto, C. R. Cervi, A. C. B. de Marchi, Recommender systems in mobile apps for health a systematic review, in: CISTI, 2017.

[48] P. Yin, P. Luo, W.-C. Lee, M. Wang, App recommendation: A contest between satisfaction and temptation, in: WSDM, 2013, pp. 395–404.

[49] J. Lin, K. Sugiyama, M.-Y. Kan, T.-S. Chua, New and improved: Modeling versions to improve app recommendation, in: SIGIR, 2014, pp. 647–656.