Local Variational Feature-based Similarity Models for Recommending Top-N New Items

YIFAN CHEN, University of Amsterdam, The Netherlands
YANG WANG*, Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, Hefei University of Technology, China
XIANG ZHAO, National University of Defense Technology, China
HONGZHI YIN, University of Queensland, Australia
ILYA MARKOV, University of Amsterdam, The Netherlands
MAARTEN DE RIJKE, University of Amsterdam, The Netherlands

The top-N recommendation problem has been studied extensively. Item-based collaborative filtering (ICF) recommendation algorithms show promising results for the problem. They predict a user's preferences by estimating similarities between a target and user rated items. Top-N recommendation remains a challenging task in scenarios where there is a lack of preference history for new items. Feature-based Similarity Models (FSMs) address this particular problem by extending ICF by estimating similarity functions of item features. The quality of the estimated similarity function determines the accuracy of the recommendation. However, existing FSMs only estimate *global* similarity functions, i.e., they estimate using preference information across all users. Moreover, the estimated similarity functions are *linear*; hence, they may fail to capture the complex structure underlying item features.

In this paper, we propose to improve FSMs by estimating local similarity functions, where each function is estimated for a subset of like-minded users. To capture global preference patterns, we extend the global similarity function from linear to non-linear, based on the effectiveness of variational autoencoders. We propose a Bayesian generative model, called the Local Variational Feature-based Similarity Model, to encapsulate local and global similarity functions. We present a variational Expectation Minimization algorithm for efficient approximate inference. Extensive experiments on a large number of real-world datasets demonstrate the effectiveness of our proposed model.

CCS Concepts: • **Information systems → Recommender systems**; **Personalization**; • **Computing methodologies** → *Learning latent representations.*

*Corresponding author.

This research was partially supported by Ahold Delhaize, the Association of Universities in the Netherlands (VSNU), the China Scholarship Council, the National Natural Science Foundation of China, with no 61806035 and 61872446, the Natural Science Foundation of Hunan under grant No. 2019JJ20024. the Netherlands Institute for Sound and Vision, and the Netherlands Organisation for Scientific Research (NWO), under project nr CI-14-25. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.
Authors' addresses: Yifan Chen, University of Amsterdam, Amsterdam, The Netherlands, y.chen4@uva.nl; Yang Wang, Key Laboratory of Knowledge Engineering with Big Data, Ministry of Education, Hefei University of Technology, Hefei, China, yangwang@hfut.edu.cn; Xiang Zhao, National University of Defense Technology, Changsha, China, xiangzhao@nudt.edu.cn; Hongzhi Yin, University of Queensland, Brisbane, Australia, db.hongzhi@gmail.com; Ilya Markov, University of Amsterdam, Amsterdam, The Netherlands, i.markov@uva.nl; Maarten de Rijke, University of Amsterdam, Amsterdam, The Netherlands, derijke@uva.nl.

1046-8188/2019/1-ART1 $15.00
https://doi.org/10.1145/3372154

Additional Key Words and Phrases: Top-N recommendation, item cold-start, item feature, deep generative model

ACM Reference Format:
Yifan Chen, Yang Wang, Xiang Zhao, Hongzhi Yin, Ilya Markov, and Maarten de Rijke. 2019. Local Variational Feature-based Similarity Models for Recommending Top-N New Items. *ACM Transactions on Information Systems* 1, 1, Article 1 (January 2019), 33 pages. https://doi.org/10.1145/3372154

1 INTRODUCTION

Top-N recommendation systems expose users to a limited number of items that reflect the most relevant items a user has not yet rated. This helps users cope with large volumes of information. Existing methods for this task broadly fall into two categories: latent space methods and neighborhood-based methods. Latent space methods [21] learn a low-rank factorization of the user-item matrix into user and item factor matrices, representing both the users and the items in a common latent space. Neighborhood-based methods [23] (user-based or item-based) focus on identifying similar users/items, where item-based neighborhood methods demonstrate better top-N recommendation performance than user-based ones [18, 19, 39, 51]. Item-based neighborhood methods can be further categorized into two classes: memory-based [23, 59] and model-based [39, 51]. Memory-based methods compute similarities between items based on statistical measures, such as Pearson coefficient and cosine similarity. However, recommendations based on such heuristic-based approaches are usually inferior. Compared to memory-based methods, model-based methods, often known as similarity models, achieve state-of-the-art performance on the top-N recommendation task by learning similarities from data [39, 51].

It remains a challenging task to recommend top-N *cold-start* items, that is, recommending N items to users from a set of *new* items. The problem of recommending top-N new items is significant because new items are continuously observed: new products are introduced, new books and articles are written, and news stories break. Conventional similarity models cannot generate a recommendation in a cold-start setting [1, 5, 6, 29, 60]. The cold-start problem strongly impacts recommendation performance and the user experience, hence it attracts much attention from the research community [17, 68]. Feature-based Similarity Models (FSMs) address the problem by extending similarity models to utilize auxiliary information associated with items, i.e., item features, where item similarity is calculated using item features. FSMs have demonstrated their effectiveness for recommending top-N new items [27, 63].

Existing FSMs have the following limitations:

- *They estimate global similarity functions only.* Existing FSMs exploit information across all users to estimate the similarity function, thus assuming that items have the same similarities for all users. In many real-world applications, item similarities should be better identified within subsets of users [18, 64], especially when a large number of users are involved. In fact, there could be a pair of items that are extremely similar for a specific subset of users, while they have low similarity for another subset of users. Existing FSMs fail to capture item similarities w.r.t. a specific aspect that is only of interest to a subset of like-minded users.
- *The estimated similarity functions are linear.* Linear similarity functions fail to capture the complex structure underlying item features. Item similarities measured by linear functions can also be inaccurate, especially when item features are sparse.

To overcome these limitations of existing FSMs, we propose to model local aspects of items that are of interest for a subset of users and extend the linear similarity function to a non-linear one. Specifically, we first identify user subsets via clustering, where users within the subset share similar preferences. For each user subset, we estimate a local similarity function. Motivated by the success of deep learning in the context of collaborative filtering [44, 74], we also estimate a global similarity function that encodes item features into deep representations to measure item similarity in a latent space. Local similarity functions capture specific aspects of items and the global similarity function encodes more abstract properties of items. The combination of local and global similarity functions captures feature-based item similarities from different perspectives.

One challenging task is how to combine deep learning with item collaborative filtering and user clustering: (1) deep learning requires the inputs to be i.i.d. [74]; therefore, it is difficult for deep models to capture implicit relationships among items, which is crucial for item collaborative filtering; and (2) deep learning is rarely applied to clustering problems; typically, deep learning-based methods are used for dimensionality reduction, followed by classical clustering techniques applied to the resulting low-dimensional space [80].

We address the challenge of combining deep learning with item collaborative filtering and user clustering by introducing a Bayesian generative model [40, 70]. We propose a *Local Variational Feature-based Similarity Model* (LVSM) that integrates deep learning with user clustering and collaborative filtering for top-N cold-start item recommendation. Inference for LVSM is challenging due to the complex entanglement of variables and the non-linear structure within the deep network. Therefore, we conduct variational inference. Existing deep learning for collaborative filtering methods introduce offset variables on top of latent item representations, which can facilitate variational inference [45, 74]. However, for new items, the offset cannot be inferred due to the absence of ratings. In order to recommend new items, they simply ignore the offset, which brings bias between the rated items and new items. Unlike these methods, LVSM assumes that the generation of user ratings depends directly on the latent item representations. However, this also brings an extra difficulty for inference. We derive the Evidence Lower Bound (ELBO) with approximations, based on which ELBO can be efficiently optimized through a variational Expectation Minimization (EM) procedure.

The contributions of our paper can be summarized as follows:

- We propose a deep generative model, LVSM, to address the item cold-start top-N recommendation problem. The model can capture local aspects of items and measure global item similarity based on deep representations extracted from item features.
- To address the difficulty of optimizing LVSM, we perform variational inference and derive the ELBO. Given this approximation, LVSM can be optimized efficiently.
- We conduct comprehensive experiments to demonstrate the effectiveness of LVSM, yielding important insights into how it generates robust recommendations with a large fraction of cold-start items and sparse item features.

The remainder of the paper is organized as follows. We introduce preliminaries in Section 2. We review related work in Section 3. We propose our model, LVSM, in Section 4 and then conduct variational inference in Section 5. Section 6 and Section 7 describe our experimental setup and results. We conclude the paper in Section 8.

2 PROBLEM DEFINITION

In this work, we consider the cold-start top-N recommendation problem, i.e., the problem of recommending items that have neither been seen nor rated by users. The problem is defined as follows: given a set of new items (rating information for these items from users is entirely missing) and their features (characteristics such as genre, product categories, keywords, etc.), recommend each user with the top-N items selected from the new items. We assume general contents as item features, which can be textual but not necessarily to be.

To recommend new items, standard cold-start recommender systems work as follows [27, 63]:

Table 1. Notation used in this paper. The first section of the table summarizes the notation regarding sets and numbers. The second section contains our notation for variables. The third section lists our notation for parameters.

	Notation	Description
Sets and numbers	$\mathcal{U}$	Set of users
	$\mathcal{I}$	Set of items
	$\mathcal{R}_u^+$	Set of items rated by user i
	$\mathcal{R}_{u-i}^+$	Set of items rated by user u excluding item i
	m	Number of users, i.e., $\lvert\mathcal{U}\rvert$
	n	Number of items, i.e., $\lvert\mathcal{I}\rvert$
	d	Number of item features
	c	Number of user groups
	n_u	Number of items rated by user i, i.e., $\left\lvert\mathcal{R}_u^+\right\rvert$
	n_{u-i}	Number of items rated by user i excluding item j, i.e., $\left\lvert\mathcal{R}_{u-i}^+\right\rvert$
Variables	$Y \in \mathbb{R}^{m\times n}$	User rating matrix
	$X \in \mathbb{R}^{n\times d}$	Item feature matrix
	$V \in \mathbb{R}^{n\times h}$	Latent item representation matrix
	$\boldsymbol{x}_i \in \mathbb{R}^d$	Feature vector of item i
	$\boldsymbol{v}_i \in \mathbb{R}^h$	Latent representation of item i
	$\boldsymbol{h}_i^{inf}$	Hidden variables of item i in the inference network
	$\boldsymbol{h}_i^{gen}$	Hidden variables of item i in the generation network
	y_{ui}	Rating of user u for item i
	s_{ij}	Similarity between item i and item j
	z_u	Indicator of the group for user u
Parameters	Θ	Parameters of the generative model
	Φ	Parameters of the inference model
	θ	Parameters of generation network
	ρ	Parameters of the inference network
	Ω	Parameters of feature weights
	$\boldsymbol{\omega}_k \in \mathbb{R}^d$	Parameters of feature weights for k-th user group
	$\pi \in \mathbb{R}^{m\times c}$	Variational parameters of Z

1. for a given user, predict her preference scores for all new items; the preference scores are predicted using some models;
2. for this user, the new items are sorted using the predicted scores in non-increasing order; the N items at the top of the sorted list are recommended to her; and
3. repeat 1. and 2. for each user in the system.

Next, we introduce the relevant notation. We write m, n, d for the number of users, items and item features, respectively. We refer to Y as the preference matrix; y_{ui} represents the rating of user u to item i. In many scenarios, user ratings are in the form of implicit feedback, such as purchase history, watching habits, browsing activity, etc. Following the common setting for implicit feedback [24], we assume that user ratings are binarized [36, 53, 71]. We refer to the item feature matrix as X. Then, $\boldsymbol{x}_i$ represents the feature vector for item i and x_{ij} represents the j-th feature of item i. We assume numerical values for item features; in this way, we are able to handle various multimedia features [45]. The notation used to describe LVSM as well as other models is summarized in Table 1.

3 RELATED WORK

The idea of estimating multiple local models together with a global model has previously been found to be effective for many recommendation tasks, including rating prediction in both general [41] and cold-start settings [64] and top-N recommendation [18, 19]. The broader message of this paper is that we extend the effectiveness of the idea to top-N recommendation in a cold-start setting. LVSM is specifically designed for recommending top-N new items. Besides reviewing models specifically designed for this problem, we review related work concerning a broader scope, e.g., methods designed for cold-start recommendation.

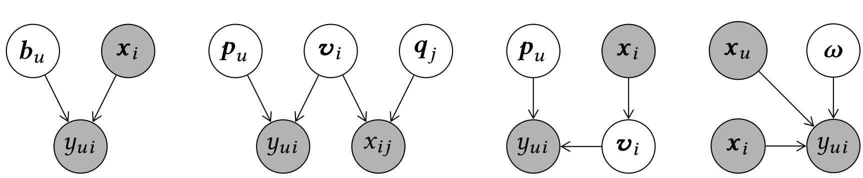
To recommend items to new users, side information associating with users are utilized [28], e.g., contextual information [47], profiling [61], social networking [10, 85], social media [91, 92] and etc. However, these additional information are not always available due to privacy issue. When confronted with this challenge task that user side information is not available, interview-based recommenders are studied, where a small number of items are selected as questions, and a new user is required to answer these questions [20, 46, 64]. Similar to interview-based methods, active learning methods have also been applied to tackle user cold-start recommendation [26].

We review cold-start item recommendations in detail. Although they are originally designed for rating prediction over new items, they can also provide a top-N recommendation from new items. Naively, new items may be recommended to users based on their popularity [54] or based on a random selection [46]. The accuracy of these methods is low as they cannot provide personalized recommendations. Alternative methods have been proposed to warm-up cold-start items by forcing several representative users to rate them [20, 46]. In recent years, there has been an increase in interest in utilizing other rich sources associated with items along with the rating matrix to increase the accuracy of the recommendation [3, 28, 77, 84], and in dealing with cold-start challenges. Although many other hybrid methods [50, 56, 66] also utilize item features, they are specifically designed to address the data sparsity problem and fail to cope with cold-start item problems, which is the main focus of this paper.

Next, we discuss work that utilizes item features, namely so-called feature-based methods.

3.1 Feature-based methods

Based on how the rating of user u for new item i, i.e., y_{ui}, is generated, different models have been proposed. Here, we review four common methods, respectively *User Modeling* (UM), *Latent Factor Model* (LFM), *Item Feature Mapping* (IFM) and *Feature-based Similarity Model* (FSM). We describe each category of models and depict them as probabilistic graphical models in Fig. 1.



(a) UM (b) LFM (c) IFM (d) FSM

Fig. 1. Overview of existing feature-based methods represented as probabilistic graphical models. $\boldsymbol{b}_u$: parameters associated with user u. $\boldsymbol{p}_u, \boldsymbol{v}_i, \boldsymbol{q}_j$: latent factors associated with user u, item i and feature j. $\boldsymbol{x}_i, \boldsymbol{x}_j$: feature vector for item i, j. $\boldsymbol{x}_u$: feature vector for user u, defined as $\boldsymbol{x}_u = \sum_{j \in \mathcal{R}^+_{u-i}} \boldsymbol{x}_j$. $\boldsymbol{\omega}$: parameters for similarity function.

User Modeling. One of the earliest approaches for identifying which of the new items may be relevant to a user is user modeling [4, 22, 30, 88]. These methods learn to generate personalized recommendations by formulating the task as a classification or regression problem. While they provide personalized recommendations, they are generally regarded as content-based filtering methods, which fail to take advantage of collaborative filtering. Later, factorization machines [12, 14, 58] have been proposed to capture feature interactions. Factorization machines can utilize item features and can be categorized as UM in the scenario of item cold-start recommendation.

Latent Factor Model. LFMs provide a better way to utilize item features that does take recent advances in matrix factorization methods into account [1, 52, 57, 60, 65, 94]. Rating and item feature matrices are simultaneously decomposed, sharing latent item factors. However, LFM requires a large parameter space, especially when item features are high-dimensional. LFM also shares item factors across different contexts [37]. This is problematic as item factors that are cold-start in the context of user ratings will be learned mainly based on data from the context of item features that are not cold-start, and therefore the item factors are not properly learned in an item cold-start setting.

Item Feature Mapping. An alternative type of feature-based model is IFM. To recommend new items, several authors [29, 72, 86] propose to form a regression model by utilizing item features. Unlike UM, they first learn a mapping function to project item features into a common latent space as user factor. Wang and Blei [72] propose an IFM based on topic modeling. A recent trend is to extract deep latent item factors for collaborative filtering [32, 69, 76]. Auto-encoders have recently been studied to learn item representations from content [45, 74]. Item representations are used as regularizations for item factors.

Feature-based Similarity Model. FSMs have been shown to achieve state-of-the-art performance for recommending top-N new items [2, 9, 23, 27, 39, 51, 63]. FSMs learn similarity functions, measuring item similarities based on item features. The similarity functions are estimated across all users, exploiting the effectiveness of item collaborative filtering. Existing FSMs estimate linear or bilinear similarity functions [27, 63]. As LVSM, our proposed method, follows the general framework of FSMs, we now discuss FSMs in more detail.

3.2 Feature-based similarity models

FSMs attempt to predict a rating score y_{ui} of user i for a new item j by defining:

$$\tilde{y}_{ui} = \sum_{j \in \mathcal{R}^+_{u-i}} \mathrm{sim}(i, j), \tag{1}$$

where $\mathcal{R}^+_{u-i}$ is the set of items rated by user i excluding item j; $\mathrm{sim}(i, j)$ is a similarity function that measures the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. When $\mathrm{sim}(\cdot)$ is linear or bilinear, Eq. (1) can be rewritten as:

$$\tilde{y}_{ui} = \mathrm{sim}(u, i). \tag{2}$$

Here, $\mathrm{sim}(u, i)$ measures the similarity of $\boldsymbol{x}_u$ and $\boldsymbol{x}_i$, where $\boldsymbol{x}_u = \sum_{j \in \mathcal{R}^+_{u-i}} \boldsymbol{x}_j$. There are several definitions for the similarity function $\mathrm{sim}(\cdot)$. One of the most intuitive ones is to calculate the dot product [23]:

$$\mathrm{sim}(u, i) = \boldsymbol{x}_u^T \boldsymbol{x}_i. \tag{3}$$

The similarity function defined in Eq. (3) has several drawbacks:

(1) *learning free*: the similarity function is predefined; it does not utilize historical preferences in order to estimate a similarity function that better predicts the observed preferences;
(2) *equal weights*: the features are treated equally when measuring item similarity; and

(3) *non-collaborative*: the rating score that is computed for a new item w.r.t. user u relies entirely on the set of items previously liked by u, and as such it does not use information from other users.

To overcome these drawbacks, Personalized Feature Weighting (PFW) [9] has been proposed; it introduces personalized weights $\boldsymbol{\omega}_u$ for item features:

$$\text{sim}(u, i) = \boldsymbol{\omega}_u^T (\boldsymbol{x}_u \circ \boldsymbol{x}_i), \tag{4}$$

where $\circ$ is the element-wise product between vectors. PFW introduces learning parameters to the model and weighs features to provide personalized recommendations. However, PFW also fails to take advantage of collaborative filtering as $\boldsymbol{\omega}_u$ is optimized separately for each user. Later, User-specific Feature-based Similarity Model (UFSM) [27] has been introduced, which defines $\text{sim}(i, j)$ as:

$$\text{sim}(u, i) = \sum_{k=1}^{c} \pi_{uk} \boldsymbol{\omega}_k^T (\boldsymbol{x}_u \circ \boldsymbol{x}_i). \tag{5}$$

Eq. (5) defines c global similarity functions $(\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_c)$ and user-specific contributions of each global similarity function $(\pi_{u1}, \ldots, \pi_{uc})$. UFSM exploits item collaborative filtering by estimating $\{\boldsymbol{\omega}_k\}$ across all users. However, UFSM fails to take into consideration interactions among features. UFSM considers item features independently. Hence, the similarity measured this way could be inaccurate especially when features are high-dimensional and sparse, where two items might share few common features. To capture feature interactions, a Feature-based factorized Bilinear Similarity Model (FBSM) [63] has been proposed, where $\text{sim}(u, i)$ is defined as:

$$\text{sim}(u, i) = \boldsymbol{x}_u^T D \boldsymbol{x}_i + \boldsymbol{x}_u^T F F^T \boldsymbol{x}_i, \tag{6}$$

where D and FF^T approximate the diagonal and off-diagonal of the feature interaction matrix, respectively. While UFSM and FBSM demonstrate superior performance for item cold-start top-N recommendations, the linearity of both models has restricted their expressiveness. Both methods estimate similarity functions from information across all users, rather than subsets of like-minded users, thus failing to capture local aspects.

3.3 Local collaborative filtering

Clustering has been widely studied for collaborative filtering [8, 31, 42, 75, 81, 82, 90]. Previous methods cluster users or items based on user ratings into subgroups and then train a local model separately for each cluster. The results from all subgroups are aggregated to produce recommendations.

Christakopoulou and Karypis [19] propose local latent factor models, where the assignments of users to subsets are constantly updated. Wang et al. [75] introduce a probabilistic model to cluster items as topics. Wu et al. [79] propose a mixture model to infer memberships of users or items to subgroups. Lee et al. [43] describe an iterative way to estimate a latent factors, where, first, latent factors representing the anchor points are estimated and, then, based on similarities of the observed entries to the anchor points, the latent factors are re-estimated. Christakopoulou and Karypis [18] explore subsets of users to learn user-specific local item similarity models, which are combined with a global similarity model.

Unlike these methods, LVSM addresses the problem of recommending new items by combining user clustering with deep learning.

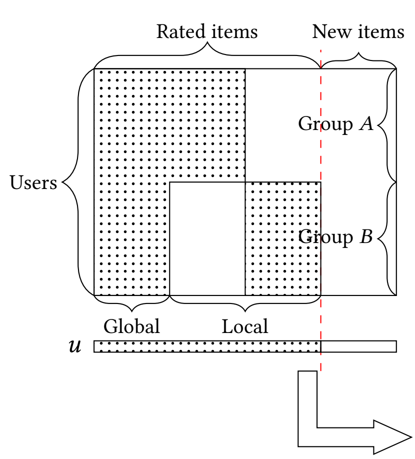
3.4 Review-based recommendation

User reviews are an important source of information for recommendation; they can help to address the rating sparsity for collaborative filtering methods [7, 11, 16, 34, 93]. Existing review-based recommenders show their effectiveness by applying sentiment analysis [55], topic modeling [49, 67] or aspect extraction [7, 13, 89] to user reviews. By concatenating all the reviews belonging to an item as item features, these methods can also help to tackle the item cold-start problem [16, 93]. Unlike these methods, we propose to utilize generic item features. We only assume to have similarity information rather instead of the semantic information behind item features. Techniques applied to user reviews cannot be applied in our setting.

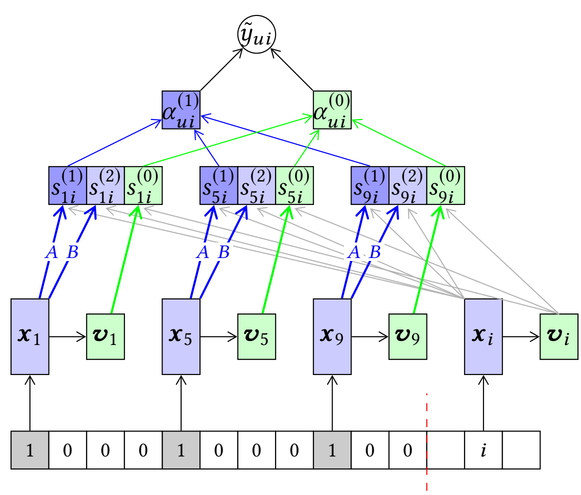
4 LOCAL VARIATIONAL FEATURE-BASED SIMILARITY MODELS

4.1 Overview

In this paper, we study the problem of recommending top-N new items to users. The solution provided in our work falls into the cold-start recommendation framework introduced in Section 2 and we contribute a more effective model for predicting scores for new items. Specifically, we propose a Bayesian generative model, an addition to the family of FSMs, namely the *Local Variational Feature-based Similarity Model* (LVSM). LVSM extends linear similarity functions to non-linear ones by learning a global similarity function via a variational autoencoder (VAE) [40]. LVSM also identifies user groups and learns the corresponding local similarity functions.



(a) Rating matrix.



(b) Calculating local and global similarities in LVSM.

Fig. 2. An illustrative example of LVSM.

Fig. 2 gives an illustrative example to describe how LVSM works. Fig. 2(a) depicts a rating matrix, where rows are users and columns are items. The dotted areas indicate the rated items by users. New items are on the right of the red dashed line. For the rated items on the left of the red dashed line, some have been rated by all users (global), some have been rated only by users in group A or group B (local). Given a user u and her history of rated items, LVSM calculates local and global similarities between the new item i and user rated items 1, 5, 9 (Fig. 2(b)). Here, $s^{(1)}_{ji}, s^{(2)}_{ji}$ are the local similarities between i and j based on users from group A and B; $s^{(0)}_{ji}$ is the global similarity. We assume user u is from group A. Thus, we formulate the prediction as $\tilde{y}_{ui} = \alpha^{(1)}_{ui} + \alpha^{(0)}_{ui} = \sum_{j\in\{1,5,9\}}(s^{(1)}_{ji} + s^{(0)}_{ji})$.

The rating matrix portrayed in Fig. 2(a) shows two distinct user groups. Learning a local model for each user group is beneficial for fitting the data. The two groups also have overlapping ratings. Therefore, the combination of global and local models can better capture user behavior. As we only cluster users (clustering items is possible but beyond the scope of this paper), a single global model is enough to capture the overlapping ratings.

For the generality of LVSM, we assume item features to be generic contents, that is, we do not presume the availability of semantic information behind the features. Therefore, the embedding layers that utilize word embeddings are excluded from LVSM.

4.2 Model description

Modeling ratings. We start by modeling ratings. As we assume that user ratings are binarized, we define the rating y_{ui} to follow a Bernoulli distribution:

$$y_{ui} \sim Bernoulli(\sigma(\tilde{y}_{ui})), \tag{7}$$

where $\sigma(\cdot)$ is the sigmoid function and $\tilde{y}_{ui}$ is the predicted score. We propose to compute $\tilde{y}_{ui}$ by:

$$\tilde{y}_{ui} = \alpha_{ui}^{(0)} + \alpha_{ui}^{(z_u)},$$

where $z_u \in \{1, \ldots, c\}$ is a variable that indicates which group user u belongs to. Furthermore, $\alpha_{ui}^{(0)}$ and $\alpha_{ui}^{(z_u)}$ are the scores calculated based on the global similarity function and the z_u-th local similarity function. Following FSM, we assume that $\alpha_{ui}^{(k)}$ is calculated by aggregating item similarities:

$$\alpha_{ui}^{(k)} = \sum_{j \in \mathcal{R}_{u-i}^{+}} s_{ji}^{(k)}, \quad \forall k \in \{0, \ldots, c\}, \tag{8}$$

where $\mathcal{R}_{u-i}^{+}$ is the set of items that are rated by user u excluding item i, and s_{ij} is the similarity between item i and j. The motivation for excluding item i is based on the estimation constraint [39] that known rating information for a particular user-item pair y_{ui} is not used when the rating for that item is being estimated. Therefore, $\tilde{y}_{ui}$ can be computed by:

$$\tilde{y}_{ui} = \sum_{j \in \mathcal{R}_{u-i}^{+}} s_{ji}^{(0)} + s_{ji}^{(z_u)}. \tag{9}$$

We combine $s_{ji}^{(0)}$ and $s_{ji}^{(z_u)}$, linearly and equally, following [63], where the local and global similarity functions capture the diagonal and off-diagonal feature interactions (Eq. (6)). The linear combination is especially useful for inference; we can derive the expectation of $\alpha_{ui}^{(k)}$ (Eq. (26)).

Modeling global similarities. Inspired by Eq. (6), we define the global similarity function to capture feature interactions. Recently, several publications have explored Deep Neural Networks (DNNs) to learn non-linear feature interactions [15, 35, 62, 87]. However, capturing feature interactions by these methods is not suitable for the global similarity function as they do not have a Bayesian nature, which complicates combinations with item-based CF. Instead, we utilize a VAE [40]. Then, the global similarity function is defined as the inner product of latent item representations learned by the VAE:

$$s_{ij}^{(0)} = \mathrm{sim}_0(\boldsymbol{x}_i, \boldsymbol{x}_j) = f_\rho(\boldsymbol{x}_i)^T f_\rho(\boldsymbol{x}_j) = \boldsymbol{v}_i^T \boldsymbol{v}_j, \tag{10}$$

where $\boldsymbol{v}_i, \boldsymbol{v}_j$ are the latent representations of item i, j, respectively; $f_\rho(\cdot)$ stands for the inference network of VAE, which is parameterized by ρ. As suggested by the VAE, we use a unit Gaussian prior for $\boldsymbol{v}_i$:

$$\boldsymbol{v}_i \sim \mathcal{N}(0, I). \tag{11}$$

Note that $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are used directly to calculate the similarity, rather than introducing offset variables like [45, 74]. This is because the offset cannot be inferred for new items. However, this complicates inference as the DNN is directly coupled with the model. Fortunately, we can derive an efficient inference thanks to the linear combination in Eq. (9).

Modeling local similarities. We define the local similarity function with respect to the k-th user group by:

$$s_{ij}^{(k)} = \mathrm{sim}_k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\omega}_k^T \left(\boldsymbol{x}_i \circ \boldsymbol{x}_j\right), \tag{12}$$

where $\circ$ is the element-wise product between vectors; $\boldsymbol{\omega}_k$ is the feature weight vector for the k-th user group, where we use a Gaussian prior:

$$\boldsymbol{\omega}_k \sim \mathcal{N}(0, \lambda_\omega^{-1} I), \tag{13}$$

Given user u, the local similarity will be calculated based on which group u belongs to, denoted by z_u. As z_u is discrete, we use a multinomial distribution for z_u. As we presume no information about which group users belong to, we assume equal probabilities:

$$z_u \sim \mathit{Multi}(1/c). \tag{14}$$

Modeling item features. The item feature $\boldsymbol{x}_i$ is generated from its latent representation $\boldsymbol{v}_i$ through a DNN. Let W_l and $\boldsymbol{b}_l$ the parameters associated with the l-layer of the DNN. Following [45, 74], we model W_l and $\boldsymbol{b}_l$ with a Gaussian distribution:

$$W_l \sim \mathcal{N}(0, \lambda_W^{-1} I), \quad \boldsymbol{b}_l \sim \mathcal{N}(0, \lambda_b^{-1} I). \tag{15}$$

The output of each layer $\boldsymbol{h}_l$ also follows a Gaussian distribution:

$$\boldsymbol{h}_l \sim \mathcal{N}(\phi(\boldsymbol{h}_{l-1}^T W_l + \boldsymbol{b}_l), I). \tag{16}$$

The feature $\boldsymbol{x}_i$ is generated from the last layer output $\boldsymbol{h}_L$. Depending on what type of data the item feature is, $\boldsymbol{x}_i$ can be assumed to be generated from a multivariate Bernoulli distribution if it is binary, or it can be generated from a Gaussian distribution if it is a real number:

$$\boldsymbol{x}_i \sim \begin{cases} \mathit{Bernoulli}(\sigma(\boldsymbol{h}_L)), & \text{if } \boldsymbol{x}_i \text{ is binary,} \\ \mathcal{N}(\boldsymbol{h}_L, \lambda_h^{-1} I), & \text{if } \boldsymbol{x}_i \text{ is real.} \end{cases} \tag{17}$$

The overall generation procedure is as follow:

(1) For each layer $l = 1, \ldots, L$,
   (a) draw the parameter $W_l \sim \mathcal{N}(0, \lambda_W^{-1} I)$;
   (b) draw the bias $\boldsymbol{b}_l \sim \mathcal{N}(0, \lambda_b^{-1} I)$.
(2) For each item $i \in \mathcal{I}$,
   (a) draw item representation $\boldsymbol{v}_i \sim \mathcal{N}(0, I)$;
   (b) draw hidden layer $\boldsymbol{h}_1 \sim \mathcal{N}(\phi(\boldsymbol{v}_i^T W_l + \boldsymbol{b}_l), \lambda_h^{-1} I)$, where $\phi(\cdot)$ is the activation function;
   (c) for each layer $l = 2, \ldots, L$, draw hidden layer $\boldsymbol{h}_l \sim \mathcal{N}(\phi(\boldsymbol{h}_{i-1}^T W_l + \boldsymbol{b}_l), \lambda_h^{-1} I)$;
   (d) draw item feature $\boldsymbol{x}_i \sim \mathit{Bernoulli}(\sigma(\boldsymbol{h}_L))$ if $\boldsymbol{x}_i$ is binary; $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{h}_L, \lambda_x^{-1} I)$ if $\boldsymbol{x}_i$ is real, where $\sigma(\cdot)$ is the sigmoid function.
(3) For each user $u \in \mathcal{U}$, draw $z_u \sim \mathit{Multi}(1/c)$.
(4) For each user group $k = 1, \ldots, c$, draw $\boldsymbol{\omega}_k \sim \mathcal{N}(0, \lambda_\omega^{-1} I)$
(5) For each user-item pair $(u, i), u \in Y$, draw $y_{ui} \sim \mathit{Bernoulli}\,(\sigma(\tilde{y}_{ui}))$, where $\tilde{y}_{ui}$ is calculated based on Eq. (9).

Once the model is optimized, we can predict the score of a new item i for user u throughout the inference of user rating $\tilde{y}_{ui}$.

5 MODEL OPTIMIZATION

In this section, we describe an optimization method for LVSM, i.e., how to optimize the parameters $\Omega = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_c\}$ for the similarity functions and the parameter θ for the generation network. Let $\Theta = \{\Omega, \theta\}$. We perform Maximum A Posteriori (MAP) estimation to infer LVSM by optimizing the following posterior:

$$\begin{aligned} p(\Theta \mid X, Y) &\simeq p(\Theta, X, Y) \\ &= p(X, Y \mid \Theta) p(\Theta) \\ &= p(\Theta) \int_V \sum_Z p(V, Z, X, Y \mid \Theta) dV. \end{aligned} \tag{18}$$

The posterior in Eq. (18) is intractable for exact inference, as the marginalization of latent variables is extremely difficult, due to the complex entanglement of variables and the non-linear structure of the deep network. Therefore, we turn to approximate inference algorithms. Based on the idea of VAE, we perform variational inference for LVSM. We first write the log-joint likelihood of LVSM:

$$\begin{aligned} \log p(V, Z, X, Y \mid \Theta) = &\log p(Y \mid V, X, Z, \Omega) + \\ &\log p(X \mid V, \theta) + \\ &\log p(V) + \\ &\log p(Z). \end{aligned} \tag{19}$$

We model the variational distribution of latent variables as $q(V, Z \mid \Phi)$, where Φ is the set of variational parameters. The ELBO [83] is given as:

$$\mathcal{L}(\Theta, \Phi; q) = \mathbb{E}_q \left[\log p(V, Z, X, Y \mid \Theta) - q(V, Z \mid \Phi)\right] + \log p(\Theta).$$

Given the ELBO, we can thus find approximate empirical Bayes estimates for LVSM via an alternating variational EM procedure that maximizes a lower bound w.r.t. the variational parameters Φ, and then, for fixed values of the variational parameters, maximizes the lower bound w.r.t. the model parameters Θ. We summarize the variational EM algorithm in Algorithm 1.

Algorithm 1: Variational EM Algorithm

```
t ← 0;
Θ^(0) ← randomly initialize parameters;
while not converge do
    Φ^(t) ← arg max_Φ L(Θ^(t), Φ; q), see Section 5.2 ;        /* E-step */
    Θ^(t+1) ← arg max_Θ L(Θ, Φ^(t); q), see Section 5.3 ;      /* M-step */
    t ← t + 1;
```

5.1 Variational inference

We discuss in detail how to derive the ELBO. For the variational distributions, we assume

$$\boldsymbol{z}_u \sim Multi(\boldsymbol{\pi}_u), \quad \boldsymbol{v}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \varsigma_i^2).$$

Based on the mean-field assumption, we fully factorize $q(V, Y, Z \mid \Phi)$:

$$q(V, Y, Z \mid \Phi) = \prod_{u=1}^{m} q(\boldsymbol{z}_u \mid \boldsymbol{\pi}_u) \prod_{i=1}^{n} q(\boldsymbol{v}_i \mid \boldsymbol{\mu}_i, \varsigma_i^2),$$

where $\pi = \{\boldsymbol{\pi}_u\}$, $\rho = \{\boldsymbol{\mu}_i, \varsigma_i^2\}$ are the free variational parameters. The number of parameters to optimize grows with the number of users and items, which becomes a bottleneck for real-world applications with millions of users and items. To address this issue, we utilize a VAE [40] to replace individual parameters $\{\boldsymbol{\mu}_i, \boldsymbol{\varsigma}_i\}$ with a data-dependent function through an inference network parameterized by ρ, i.e., $f_\rho(\boldsymbol{x}_i)$, where ρ is independent of samples and thus the scale of ρ is free from n; ρ is the parameters of inference network, which is designed to have an identical neural network structure with the generation network parameterized by θ.

Therefore, the ELBO is given as:

$$\begin{aligned}\mathcal{L}(q;\Theta,\Phi) = & \sum_{u=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{c}\pi_{uk}\mathbb{E}_{q_\rho}\left[\log p(y_{ui} \mid X, V, \boldsymbol{\omega}_k)\right] + \\ & \sum_{i=1}^{n}\mathbb{E}_{q_\rho}\left[\log p(\boldsymbol{x}_i \mid \boldsymbol{v}_i, \theta)\right] - \mathbb{KL}\left(q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho) \parallel p(\boldsymbol{v}_i)\right) + \\ & \sum_{u=1}^{m}\sum_{k=1}^{c}\pi_{uk}\left(\log p(z_u) - \log \pi_{uk}\right) + \log p(\Theta),\end{aligned} \tag{20}$$

where q_ρ is an abbreviation for $q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho)$.

We start from deriving $\mathbb{E}_{q_\rho}\left[\log p(y_{ui} \mid X, V, \boldsymbol{\omega}_k)\right]$, which is in the first line of Eq. (20):

$$\begin{aligned}\mathbb{E}_{q_\rho}\left[\log p(y_{ui} \mid X, V, \boldsymbol{\omega}_k)\right] &= \mathbb{E}_{q_\rho}\left[y_{ui}\log\sigma(\tilde{y}_{ui}^{(k)}) + (1-y_{ui})\log(1-\sigma(\tilde{y}_{ui}^{(k)}))\right] \\ &= y_{ui}\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \mathbb{E}_{q_\rho}\left[\log(\exp\{\tilde{y}_{ui}^{(k)}\}+1)\right],\end{aligned} \tag{21}$$

where $\tilde{y}_{ui}^{(k)}$ is calculated by Eq. (9) with $z_u = k$. It is not easy to infer $\mathbb{E}_{q_\rho}\left[\log(\exp\{\tilde{y}_{ui}^{(k)}\}+1)\right]$. Therefore, we approximate it as follows:

$$\begin{aligned}\mathbb{E}_{q_\rho}\left[\log(\exp\{\tilde{y}_{ui}^{(k)}\}+1)\right] &\approx \mathbb{E}_{q_\rho}\left[\log(\exp\{\tilde{y}_{ui}^{(k)}\})\right] \\ &= \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] = \log\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\} \\ &\approx \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\}+1\right).\end{aligned} \tag{22}$$

We then derive the expectation $\mathbb{E}_{q_\rho}\left[\log p\left(\boldsymbol{x}_i \mid \boldsymbol{v}_i, \theta\right)\right]$, which is the first term in the second line of Eq. (20). It is problematic to derive $\mathbb{E}_{q_\rho}\left[\log p\left(\boldsymbol{x}_i \mid \boldsymbol{v}_i, \theta\right)\right]$ due to the non-linear transformation within the inference network parameterized by ρ. While we can obtain an unbiased estimate of it by sampling $\boldsymbol{v}_i \sim q_\rho$ and perform stochastic gradient ascent to optimize it, the challenge is that we cannot trivially take gradients with respect to ρ through this sampling process. Therefore, we apply the re-parameterization trick [40], which works as follows in this setting: we first draw a sample $\boldsymbol{\epsilon}^{(l)}$, which is independent from φ and $\boldsymbol{x}_i$, and then re-parameterize $\boldsymbol{v}_i$ as follows:

$$\begin{aligned}\boldsymbol{\epsilon}^{(l)} &\sim \mathcal{N}(0, I), \\ \boldsymbol{v}_i^{(l)} &= \boldsymbol{\mu}_i + \boldsymbol{\epsilon}^{(l)} \circ \boldsymbol{\varsigma}_i^2.\end{aligned} \tag{23}$$

The $\mathbb{KL}\left(q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho) \parallel p(\boldsymbol{v}_i)\right)$, which is the second term in the second line of Eq. (20), has an analytical solution:

$$\mathbb{KL}\left(q(\boldsymbol{v}_i \mid \boldsymbol{x}_i, \rho) \parallel p(\boldsymbol{v}_i)\right) = \frac{1}{2}\left(2\log(\boldsymbol{\varsigma}_i) - \boldsymbol{\mu}_i^2 - \boldsymbol{\varsigma}_i^2\right). \tag{24}$$

Putting Eq. (21), (22), (23) and (24) together, we can rewrite the ELBO in Eq. (20) as:

$$
\begin{aligned}
\mathcal{L}(q;\Theta,\Phi) \simeq & \sum_{u=1}^{m}\sum_{i=1}^{n}\sum_{k=1}^{c}\pi_{uk}\left[y_{ui}\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\}+1\right)\right] + \log p(\Theta) + \\
& \frac{1}{L}\sum_{l=1}^{L}\sum_{i=1}^{n}\log p(\boldsymbol{x}_i \mid \boldsymbol{v}_i^{(l)},\theta) - \frac{1}{2}\sum_{i=1}^{n}\left(2\log(\varsigma_i) - \boldsymbol{\mu}_i^2 - \varsigma_i^2\right) - \sum_{u=1}^{m}\sum_{k=1}^{c}\pi_{uk}\log\pi_{uk},
\end{aligned}
\tag{25}
$$

where

$$
\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] = \boldsymbol{\omega}_k^T(\boldsymbol{x}_u \circ \boldsymbol{x}_i) + \mathbb{E}_{q_\rho}[\boldsymbol{v}_i]^T \sum\nolimits_{k\in\mathcal{R}_{u-i}^+}\mathbb{E}_{q_\rho}\left[\boldsymbol{v}_j\right]. \tag{26}
$$

The reason that we can write $\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]$ as Eq. (26) is that $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are i.i.d. samples generated form the inference network. It is worth noting that $\mathbb{E}_{q_\rho}[\boldsymbol{v}_i] = \boldsymbol{\mu}_i, \mathbb{E}_{q_\rho}\left[\boldsymbol{v}_j\right] = \boldsymbol{\mu}_j$.

Based on the variational inference in this section, we can detail the variational E-step and M-step, respectively in Section 5.2 and 5.3.

5.2 Variational E-step

We update the variational parameter $\pi = \{\boldsymbol{\pi}_u\}$ in the E-step. We isolate the optimization problem for $\boldsymbol{\pi}_u$ as:

$$
\min_{\boldsymbol{\pi}_u} \quad \sum_{k=1}^{c}\pi_{uk}\gamma_{uk} - \pi_{uk}\log\pi_{uk},
$$

where $\sum_{k=1}^{c}\pi_{uk} = 1$ and

$$
\gamma_{uk} = \sum_{i=1}^{n}\pi_{uk}\left[y_{ui}\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\}+1\right)\right]. \tag{27}
$$

By introducing the Lagrange multiplier λ_u, we have:

$$
\mathcal{L} = \sum_{k=1}^{c}(\pi_{uk}\gamma_{uk} - \pi_{uk}\log\pi_{uk}) - \lambda_u(\sum_{k=1}^{c}\pi_{uk} - 1).
$$

The derivative of $\mathcal{L}$ over π_{uk} is

$$
\nabla_{\pi_{uk}}\mathcal{L} = \gamma_{uk} - \log\pi_{uk} - \lambda_u - 1.
$$

Applying the Karush-Kuhn-Tucker (KKT) first-order optimality conditions, we have:

$$
\pi_{uk} = \frac{\exp(\gamma_{uk}-1)}{\sum_{k=1}^{c}\exp(\gamma_{uk}-1)},
$$

which provides the desired closed-form.

5.3 Variational M-step

We update the parameters of VAE (ρ and θ) and the parameters of local similarity functions ($\Omega = \{\boldsymbol{\omega}_k\}$) in the M-step. We propose to optimize these parameters through stochastic gradient ascent. At each time we select a user u and an item i. We write $\mathcal{L}_{ui}$ to denote the loss of ELBO regarding u, i:

$$
\begin{aligned}
\mathcal{L}_{ui} = & \sum_{k=1}^{c}\pi_{uk}\left[y_{ui}\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] - \log\left(\exp\left\{\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]\right\}+1\right)\right] + \\
& \sum_{j\in\mathcal{R}_u^+}\left[\mathbb{E}_{q_\varphi}\left[\log p(\boldsymbol{x}_j \mid \boldsymbol{v}_j,\theta)\right] - \frac{1}{2}\left(2\log(\varsigma_j) - \boldsymbol{\mu}_j^2 - \varsigma_j^2\right)\right].
\end{aligned}
\tag{28}
$$

The gradient of $\mathcal{L}_{ui}$ w.r.t. $\boldsymbol{\mu}_j, \forall j \in \mathcal{R}_u^+$ is

$$\nabla_{\boldsymbol{\mu}_j} \mathcal{L}_{ui} = \sum_{k=1}^{c} \pi_{uk} \left[y_{ui} - \sigma(\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]) \right] \nabla_{\boldsymbol{\mu}_j} \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] + \frac{1}{L} \sum_{l=1}^{L} \nabla_{\boldsymbol{v}_j^{(l)}} \log p(\boldsymbol{x}_j \mid f_\theta(\boldsymbol{v}_j^{(l)})) + \boldsymbol{\mu}_j, \tag{29}$$

where

$$\nabla_{\boldsymbol{\mu}_i} \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] = \sum_{j \in \mathcal{R}_{u-i}^+} \boldsymbol{\mu}_j,$$

$$\nabla_{\boldsymbol{\mu}_j} \mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}] = \boldsymbol{\mu}_i, \quad j \in \mathcal{R}_{u-i}^+.$$

The gradient of $\mathcal{L}_{ui}$ w.r.t. $\boldsymbol{\varsigma}_j, \forall j \in \mathcal{R}_u^+$ is:

$$\nabla_{\boldsymbol{\varsigma}_j} \mathcal{L}_{ui} = \frac{1}{L} \sum_{l=1}^{L} \nabla_{\boldsymbol{v}_j^{(l)}} \log p(\boldsymbol{x}_j \mid f_\theta(\boldsymbol{v}_j^{(l)})) \circ \boldsymbol{\epsilon}^{(l)} - \frac{1}{\boldsymbol{\varsigma}_i} + \boldsymbol{\varsigma}_i. \tag{30}$$

The parameters of the generation network (θ) and the inference network (ρ) can be updated through backpropagation, once $\boldsymbol{\mu}_j$ and $\boldsymbol{\varsigma}_j$ have been updated. The gradient of $\mathcal{L}_{ui}$ w.r.t. $\boldsymbol{\omega}_k$ is:

$$\nabla_{\boldsymbol{\omega}_k} \mathcal{L}_{ui} = -\pi_{uk} \left[y_{ui} - \sigma(\mathbb{E}_{q_\rho}[\tilde{y}_{ui}^{(k)}]) \right] (\boldsymbol{x}_u \circ \boldsymbol{x}_i). \tag{31}$$

Algorithm 2 summarizes the variational M-step.

Algorithm 2: Variational M-step

while not converged **do**
- **for** user $u \in \mathcal{U}$ **do**
  - $i \leftarrow$ randomly select item from $\mathcal{R}_u^+ \cup \mathcal{R}_u^-$;
  - **for** item $j \in \mathcal{R}_u^+$ **do**
    - $\boldsymbol{\mu}_j, \boldsymbol{\sigma}_j \leftarrow$ generate through inference network;
    - **for** $l = 1$ **to** L **do**
      - $\boldsymbol{v}_j^{(l)} \leftarrow$ sample according to Eq. (23);
    - $\nabla_{\boldsymbol{\mu}_j} \mathcal{L}_{ui} \leftarrow$ calculate according to Eq. (29);
    - $\nabla_{\boldsymbol{\varsigma}_j} \mathcal{L}_{ij} \leftarrow$ calculate according to Eq. (30);
    - $\boldsymbol{\mu}_j \leftarrow \boldsymbol{\mu}_j + \eta \nabla_{\boldsymbol{\mu}_j} \mathcal{L}_{ui}$;
    - $\boldsymbol{\sigma}_j \leftarrow \boldsymbol{\sigma}_j + \eta \nabla_{\boldsymbol{\sigma}_j} \mathcal{L}_{ui}$;
  - **for** $k = 1$ **to** c **do**
    - $\nabla_{\boldsymbol{\omega}_k} \mathcal{L}_{ij} \leftarrow$ calculate according to Eq. (31);
    - $\boldsymbol{\omega}_k \leftarrow \boldsymbol{\omega}_k + \eta \left(\nabla_{\boldsymbol{\omega}_k} \mathcal{L}_{ui} - \lambda_\omega \boldsymbol{\omega}_k \right)$;

5.4 Computational analysis

We analyze the complexity of optimizing LVSM. In the variational E-step (Section 5.2), we calculate π_u based on Eq. (27), before which we calculate $\tilde{y}_{ui}$ based on Eq. (26). Since the calculations of π_{u_1} and π_{u_2} are independent, $\forall u_1 \neq u_2 \in \mathcal{U}$, we can optimize the variational parameter π_u in parallel. Therefore, we only analyze the complexity of computing π_u for one user, which is $O(n(d + h \left|\mathcal{R}_{u-i}^+\right|))$, where h is the size of item representations.

As the variational M-step (Section 5.3) is optimized via stochastic optimization, we analyze the complexity of evaluating a single sample, a user-item pair (u, i), i.e., the complexity of calculating Eq. (28), which is $O(c(d + h\left|\mathcal{R}^+_{u-i}\right|) + 2\left|\mathcal{R}^+_u\right| \sum_{l=1}^{L}(h_{l-1}h_l))$. Here, h_l denotes the size of the l-th layer, where $h_0 = d$ and $h_L = h$, and the term $2\sum_{l=1}^{L}(h_{l-1}d_l)$ is included for the complexity of the VAE, which has an inference network and a generation network with an identical network structure.

6 EXPERIMENTAL SETUP

In this section, we detail the experimental setup used to evaluate LVSM.

6.1 Research questions

We seek to answer the following research questions:

(RQ1) How does LVSM perform on the item cold-start top-N recommendation task?
(RQ2) Does modeling local and global similarities help to improve performance?
(RQ3) What is the impact of feature sparsity on the recommendation performance?
(RQ4) What is the effect of the fraction of cold-start items on the recommendation performance?
(RQ5) How well can LVSM perform on large-scale datasets?

6.2 Datasets

To answer our research questions we conduct experiments on five datasets, respectively Beauty, Games, Sports, Kindle, CUL-a and CUL-t. Beauty, Games, Sports and Kindle are constructed from different categories of Amazon products [49]. For each category, the original dataset contains transactions between users and items, indicating implicit user feedback. We convert the multivariate rating values to 1s and filtered out less popular product items and users that appeared less than three transactions to construct the implicit rating matrix. For each dataset, we use the product reviews as the features of the product items. We extract unigram features from the review articles and remove stopwords. For Beauty, Games and Sports, we select the most frequent 8,000 features as the item features and represent each product item as a bag-of-words feature vector, where feature value is binarized. We retain the original features of Kindle as we evaluate scalability of LVSM on Kindle.

CUL-a and CUL-t are datasets of user libraries of articles with different scales and degrees of sparsity obtained from CiteULike.[1] The first dataset, CUL-a, has been collected by Wang and Blei [72]. The second dataset, CUL-t, has been independently collected by Wang et al. [73] and is even larger and sparser. Each article in the two datasets has a title and abstract. The content information of the articles is the concatenation of the titles and abstracts. We follow the same procedure as in [72] to preprocess the text content information. After removing stop words, the vocabulary for each dataset is selected according to the tf-idf value of each word.

The statistics of the datasets are presented in Table 2. *#User*, *#Item* and *#Feature* denote the number of users, items and features, respectively. The rating density is calculated as:

$$Rating\ density = \frac{\#Rating}{\#User \times \#Item},$$

where *#Rating* is the number of interactions between user and item. It is common that values of item features are missing, especially when an item feature is high-dimensional, e.g., the bag-of-words representation from a text. We write 0 for missing feature values. Therefore, we can also measure the density of features:

$$Feature\ sparsity = \frac{\#Nonzero}{\#Item \times \#Feature},$$

[1] http://www.citeulike.org

where #*Nonzero* is the number of non-zero feature values.

Table 2. Statistics of the datasets used.

Dataset	*#User*	*#Item*	*#Feature*	*Rating density*	*Feature density*	*Feature type*
Beauty	5,083	11,909	8,000	0.150%	0.528%	binary
Games	6,255	10,672	8,000	0.180%	0.324%	binary
Sports	6,174	13,257	8,000	0.116%	0.665%	binary
Kindle	26,555	22,203	11,308	0.085%	2.011%	binary
CUL-a	5,480	11,564	7,988	0.276%	0.838%	tf-idf
CUL-t	7,947	7,582	7,715	0.162%	0.224%	tf-idf

Note that we binarize feature values for the Amazon datasets (Beauty, Games, Sports, and Kindle) and calculate tf-idf for CiteULike. The reason is that user reviews are generally noisy in terms of how they describe items, which might harm the performance of recommendation. We binarize item features for the Amazon datasets to overcome noise. On the other hand, the features for CiteULike are extracted from scientific papers, which we assume to be qualified and relevant as item features. Tf-idf values could well benefit the recommendations. It is also worth noting that tf-idf values are only suitable for text features, where binary values are applicable for other features, e.g., item attributes, image pixels, etc. Experimenting with binary values can demonstrate the generality of the proposed model.

6.3 Evaluation protocol

We follow the evaluation methodology of [27, 63] to evaluate the performance of item cold-start top-N recommendation. Specifically, we split the user rating matrix Y into Y_{train}, Y_{valid} and Y_{test}, respectively, for training, validating and testing. We assume that each subset of ratings contains non-overlapping items (columns) of Y, so that we can evaluate the performance of recommending new items as users in Y_{train} do not have any preferences for items in Y_{valid} and Y_{test}. In this paper, we randomly select 80%, 10% and 10% items for Y_{train}, Y_{valid} and Y_{test}, respectively. For each user, the cold-start items are sorted in decreasing order and the first N items are returned as the top-N recommendations for that user. The list of recommended items is validated with Y_{valid} and evaluated with Y_{test} using two metrics: Recall at N ($Rec@N$) and Discounted Cumulative Gain at N ($DCG@N$) [38]. Given the list of top-N recommended items for user u, $Rec@N$ measures how many of the items liked by u appeared in that list, whereas the $DCG@N$ measures how high the relevant items were placed in the list. For a fair comparison with the most relevant works, UFSM [27] and FBSM [63], which also study recommending top-N new items, we follow their definition of $DCG@N$, which ensures $DCG@N$ to take on values within the $[0, 1]$ interval. $Rec@N$ and $DCG@N$ are defined as follows:

$$Rec@N = \frac{|\text{relevant items} \cap \text{recommended items}|}{|\text{relevant items}|},$$

$$DCG@N = imp_1 + \sum_{i=2}^{N} \frac{imp_i}{\log_2(i)},$$

where the importance score imp_i of the items with rank i in the top-N list is:

$$imp_i = \begin{cases} \frac{1}{N}, & \text{if the item at rank } i \text{ is relevant,} \\ 0, & \text{otherwise.} \end{cases}$$

The main difference between *Rec@N* and *DCG@N* is that *DCG@N* is sensitive to the rank of the items in the top-N list. Both *Rec@N* and the *DCG@N* are computed for each user and then averaged over all users.

6.4 Methods used for comparison

We evaluate LVSM by comparing it against eight other feature-based models for recommending new items. We use the categories of models introduced in Section 3.1 to characterize the methods we consider.

- **Feature-based Singular Value Decomposition (SVDFeature)** [12]: A feature-based matrix factorization method. For item cold-start recommendation, the rating score y_{ui} is estimated as:

$$y_{ui} = b_u + \boldsymbol{b}_i^T \boldsymbol{x}_i + \boldsymbol{p}_u^T \sum_{j=1}^{d} x_{ij} \boldsymbol{w}_j = \sum_{j=0}^{d} b'_j x_{ij}, \tag{32}$$

  where $b'_0 = b_u, b'_t = \boldsymbol{p}_u^T \boldsymbol{w}_j, j \geq 1$. Thus, SVDFeature can be categorized as UM. We utilize the ranking method of SVDFeature for our experiments.
- **Simple Cosine-Similarity (coSim)** [9]: A memory-based neighborhood method that estimates item similarities by cosine similarity based on item features. The preference score y_{ui} is estimated as:

$$y_{ui} = \sum_{j \in \mathcal{R}_u^+} \frac{\boldsymbol{x}_i^T \boldsymbol{x}_j}{|\boldsymbol{x}_i| \left|\boldsymbol{x}_j\right|}. \tag{33}$$

- **Personalized Feature Weighting (PFW)** [9]: A non-collaborative technique that learns user models independently. A feature weighting vector $\boldsymbol{\omega}_u$ of length d is estimated for each user u to reflect the importance of the different item features for each user. The preference score y_{ui} of user u for item i is estimated as:

$$y_{ui} = \sum_{j \in \mathcal{R}_u^+} \boldsymbol{\omega}_u^T (\boldsymbol{x}_i \circ \boldsymbol{x}_j). \tag{34}$$

- **Local Collective Embeddings (LCE)** [60]: A typical LFM that collectively factorizes a user rating matrix and an item feature matrix. For a new item i with feature $\boldsymbol{x}_i$, the item factor $\boldsymbol{v}_i$ is first inferred by solving $\boldsymbol{x}_i = \boldsymbol{v}_i W^T$, where $W \in \mathbb{R}^{d \times h}$ is the feature factor.
- **Neural Semantic Personalized Ranking (NSPR)** [25]: A typical IFM that maps item feature via a DNN. During recommendation, the expectation of item representation $\boldsymbol{v}_i$ is inferred from item feature $\boldsymbol{x}_i$ via the DNN.
- **Collaborative Variational Autoencoder (CVAE)** [45]: A state-of-the-art IFM. The difference between NSPR and CVAE is that CVAE learns item representations from item features via a VAE.
- **User-specific Feature-based Similarity Model (UFSM)** [27]: A FSM that models global aspects by learning multiple global similarity functions; the user-specific similarity function is calculated by aggregating global similarities with personalized weights.
- **Feature-based factorized Bilinear Similarity Model (FBSM)** [63]: A FSM that models the interaction between features. The interaction matrix is further factorized to reduce the complexity. The similarity function is defined by Eq. (6).

We summarize these methods in Table 3. Note that while all methods can generate personalized recommendations, coSim and PFW fail to take advantage of collaborative filtering. For PFW, UFSM and FBSM, we train both a point-wise loss function and a pair-wise loss function, where the model with the pair-wise loss function is subscripted with $_{pair}$, i.e., PFW$_{pair}$, UFSM$_{pair}$ and FBSM$_{pair}$. We train NSPR with two types of pairwise probability, respectively, logistic probability and probit probability, as proposed in [25]; we refer to the respective models as NSPR-L and NSPR-P. By comparing LVSM_1 with FBSM, we can evaluate the effectiveness of non-linear similarity functions, since the only difference between LVSM_1 and FBSM is that LVSM_1 utilizes non-linear similarity function to approximate the off-diagonal of feature interaction matrix of FBSM.

Table 3. Methods used for comparison.

Method	Personalized	Collaborative	Category
SVDFeature [12]	√	√	UM
LCE [60]	√	√	LFM
NSPR [25]	√	√	IFM
CVAE [45]	√	√	IFM
coSim [9]	√	–	FSM
PFW [9]	√	–	FSM
FBSM [63]	√	√	FSM
UFSM [27]	√	√	FSM
LVSM (this paper)	√	√	FSM

6.5 Parameter settings

For LVSM we fix $\lambda_W = \lambda_b = \lambda_h = \lambda_\omega = 0.1$ for the prior of parameters $p(\Theta)$. We choose a two-layer Multi Layer Perceptron (MLP) network architecture (50−10 for the inference network and 10−50 for the generation newtork) with a ReLU activation function [33]. We select a smaller network scale for CUL-t (10−5 for the inference network and 5−10 for the generation newtork) as the item feature of the dataset is extremely sparse so that algorithms easily overfit. For the number of local aspects, we try $c = 1, 2, 3$, respectively, and denote the corresponding model as LVSM_1, LVSM_2 and LVSM_3.

We select the same network structure for CVAE as it also utilizes a VAE. We select a larger scale for the network of VAE for CVAE as it is used for learning item factors (100−50 for the inference network and 50−100 for the generation network). We also try to find a smaller network scale for CVAE on CUL-t, but find out that it is not possible to improve the performance. Therefore, we keep the same network scale for CVAE over all datasets. Similarly, we also select a two-layer perceptron (100−50) for NSPR.

For the methods used for comparison, we select the hyper-parameters by $Rec@10$ on the validation set R_{valid}. A detailed list of parameter settings is included in Table 4; there, we tune the ℓ_2-norm regularization parameter μ for PFW, which is selected from $\{0.01, 0.1, 1, 10\}$. We tune μ_1, μ_2, λ and l for UFSM, where μ_1, μ_2, λ are selected from $\{0.01, 0.1, 1, 10\}$ and l from $\{1, 2, 3, 4, 5, 6\}$. We tune β, λ and k for FBSM, where β, λ are selected from $\{0.01, 0.1, 1, 10\}$ and k from $\{1, 5, 10, 20\}$. We tune λ_1 and λ_2 for SVDFeature, which, respectively, stand for the regularization parameter of the user factor and the item factor; both λ_1 and λ_2 are selected from $\{0.01, 0.1, 1, 10\}$. We also tune the latent dimension k for SVDFeature, which is selected from $\{50, 100, 200, 500\}$. We tune α, β, λ for LCE, where α balances the importance of user rating and item feature, which is within $[0, 1]$; we select α from $\{0.1, 0.2, \ldots, 0.9\}$ and β, λ from $\{0.01, 0.1, 1, 10\}$. We also tune the latent dimension k for LCE, which is selected from $\{50, 100, 200, 500\}$. We tune λ_v and λ_u for CVAE; λ_v controls the contribution of latent item representation to item factor, which we select from $\{0.1, 0.2, \ldots, 0.9\}$; λ_u is the regularization for user factor, which we selected from $\{0.01, 0.1, 1, 10\}$. Similarly, we also tune λ_v and λ_u for CVAE, which are selected from $\{0.1, 0.2, \ldots, 0.9\}$ and $\{0.01, 0.1, 1, 10\}$ respectively.

6.6 Experiments

To answer our research questions, we conduct different set of experiments:

Table 4. Tuned parameter values of different methods on different datasets.

Dataset	PFW [9]	PFW$_{pair}$ [9]	UFSM [27]				UFSM$_{pair}$ [27]			
	μ	μ	μ_1	μ_2	λ	c	μ_1	μ_2	λ	c
Beauty	10	1	0.01	1	0.1	3	1	1	0.1	2
Games	10	0.01	1	0.01	1	1	0.1	1	0.1	2
Sports	0.1	10	1	1	0.01	4	0.1	0.01	1	6
CUL-a	0.1	0.01	1	0.01	1	6	0.1	1	0.01	6
CUL-t	1	1	1	0.01	1	5	0.1	0.01	0.1	5

Dataset	FBSM [63]			FBSM$_{pair}$ [63]			NSPR-L [25]		NSPR-P [25]	
	β	λ	k	β	λ	k	λ_v	λ_u	λ_v	λ_u
Beauty	0.1	0.01	1	0.1	0.01	5	0.1	1	0.5	0.01
Games	0.01	1	10	0.1	0.01	5	0.9	10	0.9	1
Sports	0.01	0.1	5	1	0.01	10	0.9	0.1	0.5	1
CUL-a	0.01	1	1	0.1	0.1	1	0.5	0.1	0.5	0.01
CUL-t	0.01	1	1	0.1	0.1	1	0.9	0.01	0.1	0.01

Dataset	SVDFeature [12]			CVAE [45]		LCE [60]			
	α	β	k	λ_v	λ_u	α	β	λ	k
Beauty	0.01	1	50	0.5	1	0.9	1	0.1	200
Games	0.1	0.1	50	0.5	1	0.5	10	1	200
Sports	0.01	0.01	50	0.9	1	0.5	1	1	500
CUL-a	0.01	0.01	500	0.5	1	0.9	1	0.1	500
CUL-t	0.01	0.01	500	0.9	1	0.9	1	1	500

- To answer RQ1, we generate the top-N recommendations of new items by comparing all baselines with LVSM on Beauty, Games, Sports, CUL-a and CUL-t (Section 7.1).
- To answer RQ2, we run incremental experiments on the Beauty, Games, CUL-a and CUL-t datasets to evaluate the modeling of local and global similarity functions of FSMs. We also show what user sub-groups learned by LVSM look like through a qualitative example on Games (Section 7.2).
- To answer RQ3, we manually sparsify the Beauty dataset and evaluate the performance on the Beauty dataset with different feature densities (Section 7.3).
- To answer RQ4, we vary the number of new items and run experiments on the Sports datasets (Section 7.4).
- To answer RQ5, we compare both efficiency and accuracy of LVSM with other FSMs on the Kindle dataset (Section 7.5).

7 RESULTS AND ANALYSIS

In this section we report on the results of our experiments and answer our research questions.

7.1 Performance comparison

To address RQ1, we present an overall comparison of the top-N recommenders that we consider. We report the recommendation results in terms of *Rec@N* and *DCG@N* in Table 5, where respectively 5, 10, 15, 20 items are recommended. We show the best score in **boldface** and the second best is <u>underlined</u>. We conducted two-sided tests for the null hypothesis that the best and the second best have identical average values. We attach asterisks to the best score if the improvement over the second best is statistically significant; we use a single asterisk * if $p < 0.05$ and two asterisks ** if $p < 0.01$. Note that we do not take LVSM for both best and second best, that is, if LVSM$_1$ performs the best in one metric, we do not take LVSM_2 as the second best in the same metric even if it is. This is because we want to show whether the improvement of LVSM is significant over other baselines.

Table 5. Performance of recommending top-N new items on different datasets in terms of $Rec@N$ and $DCG@N$, where N equals to 5, 10, 15, 20, respectively.

Beauty	*Rec@5*	*Rec@10*	*Rec@15*	*Rec@20*	*DCG@5*	*DCG@10*	*DCG@15*	*DCG@20*
coSim [9]	0.1045	0.1490	0.1885	<u>0.2245</u>	<u>0.0380</u>	<u>0.0225</u>	<u>0.0168</u>	<u>0.0137</u>
SVDFeature [12]	0.0035	0.0261	0.0273	0.0297	0.0013	0.0026	0.0018	0.0014
NSPR-L [25]	0.0378	0.0535	0.0707	0.0814	0.0145	0.0086	0.0065	0.0052
NSPR-P [25]	0.0083	0.0275	0.0415	0.0596	0.0026	0.0029	0.0026	0.0024
CVAE [12]	0.0932	0.1318	0.1584	0.1816	0.0346	0.0206	0.0151	0.0121
LCE [60]	0.0996	0.1408	0.1801	0.2065	0.0367	0.0220	0.0163	0.0130
PFW [9]	0.1009	0.1445	0.1848	0.2179	0.0361	0.0216	0.0162	0.0132
PFW_{pair} [9]	0.1005	<u>0.1501</u>	<u>0.1893</u>	0.2233	0.0362	0.0219	0.0164	0.0133
UFSM [27]	<u>0.1065</u>	0.1486	0.1870	0.2220	<u>0.0380</u>	0.0223	0.0166	0.0135
UFSM_{pair} [27]	0.1000	0.1478	0.1864	0.2220	0.0360	0.0218	0.0162	0.0132
FBSM [63]	0.0501	0.0836	0.1180	0.1443	0.0177	0.0117	0.0092	0.0079
FBSM_{pair} [63]	0.0503	0.0849	0.1201	0.1488	0.0188	0.0120	0.0096	0.0079
LVSM_1	0.1061	0.1544	0.1899	0.2242	0.0396	0.0239	0.0176	**0.0143****
LVSM_2	0.1077	0.1523	0.1904	0.2232	0.0395	0.0235	0.0175	0.0141
LVSM_3	**0.1100**	**0.1579***	**0.1940**	**0.2276**	**0.0401***	**0.0242****	**0.0177****	**0.0143****
Games	*Rec@5*	*Rec@10*	*Rec@15*	*Rec@20*	*DCG@5*	*DCG@10*	*DCG@15*	*DCG@20*
coSim [9]	0.0653	<u>0.1050</u>	0.1401	0.1645	0.0216	0.0139	0.0107	0.0088
SVDFeature [12]	0.0025	0.0054	0.0086	0.0121	0.0008	0.0007	0.0005	0.0005
NSPR-L [25]	0.0080	0.0187	0.0236	0.0296	0.0027	0.0022	0.0017	0.0014
NSPR-P [25]	0.0130	0.0177	0.0232	0.0335	0.0044	0.0026	0.0020	0.0018
CVAE [12]	0.0538	0.0854	0.1149	0.1381	0.0175	0.0113	0.0088	0.0072
LCE [60]	0.0475	0.0768	0.1028	0.1274	0.0161	0.0106	0.0082	0.0068
PFW [9]	0.0611	0.1015	0.1282	0.1535	0.0202	0.0132	0.0100	0.0083
PFW_{pair} [9]	0.0576	0.0968	0.1261	0.1524	0.0194	0.0128	0.0098	0.0081
UFSM [27]	0.0596	0.0974	0.1291	0.1553	0.0196	0.0126	0.0098	0.0081
UFSM_{pair} [27]	0.0621	0.0987	0.1311	0.1585	0.0210	0.0134	0.0103	0.0085
FBSM [63]	**0.0739**	0.1007	<u>0.1414</u>	<u>0.1699</u>	<u>0.0236</u>	<u>0.0141</u>	<u>0.0110</u>	<u>0.0091</u>
FBSM_{pair} [63]	**0.0739**	0.1007	0.1402	<u>0.1699</u>	0.0232	0.0139	0.0108	0.0090
LVSM_1	0.0732	0.1084	0.1420	0.1704	0.0242	0.0148	0.0112	0.0092
LVSM_2	**0.0739**	0.1078	0.1387	0.1698	0.0238	0.0146	0.0111	0.0092
LVSM_3	0.0736	**0.1112***	**0.1441**	**0.1757***	**0.0243**	**0.0151****	**0.0114****	**0.0094****
Sports	*Rec@5*	*Rec@10*	*Rec@15*	*Rec@20*	*DCG@5*	*DCG@10*	*DCG@15*	*DCG@20*
coSim [9]	0.0599	<u>0.0924</u>	<u>0.1185</u>	<u>0.1402</u>	0.0167	0.0106	0.0079	<u>0.0065</u>
SVDFeature [12]	0.0042	0.0078	0.0194	0.0230	0.0011	0.0007	0.0009	0.0007
NSPR-L [25]	0.0128	0.0224	0.0289	0.0380	0.0040	0.0026	0.0020	0.0017
NSPR-P [25]	0.0141	0.0223	0.0318	0.0380	0.0043	0.0027	0.0022	0.0017
CVAE [12]	**0.0648**	0.0875	0.1004	0.1124	**0.0202**	**0.0116**	<u>0.0082</u>	0.0064
LCE [60]	0.0539	0.0866	0.1092	0.1294	0.0165	0.0104	0.0077	0.0063
PFW [9]	0.0554	0.0871	0.1121	0.1362	0.0162	0.0102	0.0077	0.0063
PFW_{pair} [9]	0.0555	0.0881	0.1125	0.1368	0.0163	0.0103	0.0077	0.0064
UFSM [27]	0.0575	0.0917	0.1178	0.1387	0.0160	0.0103	0.0077	0.0063
UFSM_{pair} [27]	0.0561	0.0919	0.1168	0.1376	0.0162	0.0105	0.0079	0.0064
FBSM [63]	0.0313	0.0550	0.0678	0.0850	0.0078	0.0055	0.0041	0.0035

FBSM$_{pair}$ [63]	0.0313	0.0550	0.0708	0.0880	0.0078	0.0055	0.0042	0.0036
LVSM$_1$	0.0618	0.0972	**0.1265****	**0.1480***	0.0184	0.0115	**0.0087**	**0.0070***
LVSM$_2$	0.0607	0.0936	0.1244	0.1467	0.0184	0.0114	0.0086	0.0069
LVSM$_3$	0.0640	**0.0981***	0.1206	0.1443	0.0187	**0.0116**	0.0086	**0.0070***
CUL-a	*Rec*@5	*Rec*@10	*Rec*@15	*Rec*@20	*DCG*@5	*DCG*@10	*DCG*@15	*DCG*@20
coSim [9]	0.1977	0.2819	0.3394	0.3840	0.0973	0.0590	0.0435	0.0348
SVDFeature [12]	0.0017	0.0065	0.0109	0.0149	0.0013	0.0013	0.0011	0.0010
NSPR-L [25]	0.0020	0.0065	0.0102	0.0120	0.0022	0.0018	0.0015	0.0012
NSPR-P [25]	0.0034	0.0087	0.0135	0.0197	0.0021	0.0017	0.0015	0.0014
CVAE [12]	0.0348	0.0594	0.0823	0.1020	0.0255	0.0166	0.0128	0.0106
LCE [60]	0.1639	0.2572	0.3203	0.3657	0.0871	0.0556	0.0420	0.0337
PFW [9]	0.1798	0.2574	0.3146	0.3602	0.0890	0.0544	0.0403	0.0323
PFW$_{pair}$ [9]	0.1812	0.2609	0.3148	0.3558	0.0900	0.0552	0.0406	0.0325
UFSM [27]	0.1905	0.2738	0.3291	0.3724	0.0935	0.0577	0.0424	0.0339
UFSM$_{pair}$ [27]	0.1940	0.2770	0.3330	0.3766	0.0941	0.0574	0.0424	0.0340
FBSM [63]	0.0053	0.0093	0.0154	0.0212	0.0042	0.0026	0.0021	0.0018
FBSM$_{pair}$ [63]	0.0043	0.0093	0.0147	0.0170	0.0028	0.0022	0.0018	0.0014
LVSM$_1$	0.2108	0.3019	0.3633	0.4089	0.1083	0.0664	0.0487	0.0388
LVSM$_2$	**0.2226****	**0.3172****	0.3772	0.4225	0.1116	0.0683	0.0501	0.0400
LVSM$_3$	0.2173	0.3108	**0.3800****	**0.4226****	**0.1121****	**0.0684****	**0.0504****	**0.0401****
CUL-t	*Rec*@5	*Rec*@10	*Rec*@15	*Rec*@20	*DCG*@5	*DCG*@10	*DCG*@15	*DCG*@20
coSim [9]	**0.1972**	0.2728	0.3316	0.3704	**0.0629**	**0.0379**	0.0278	0.0221
SVDFeature [12]	0.0093	0.0148	0.0190	0.0304	0.0032	0.0021	0.0016	0.0015
NSPR-L [25]	0.0051	0.0097	0.0140	0.0181	0.0019	0.0014	0.0013	0.0012
NSPR-P [25]	0.0053	0.0087	0.0135	0.0176	0.0028	0.0020	0.0016	0.0013
CVAE [12]	0.0368	0.0650	0.0927	0.1137	0.0143	0.0096	0.0078	0.0065
LCE [60]	0.1513	0.2298	0.2870	0.3253	0.0502	0.0315	0.0236	0.0189
PFW [9]	0.1777	0.2557	0.3089	0.3517	0.0562	0.0343	0.0252	0.0203
PFW$_{pair}$ [9]	0.1821	0.2559	0.3076	0.3502	0.0570	0.0344	0.0253	0.0203
UFSM [27]	0.1865	0.2671	0.3213	0.3640	0.0599	0.0364	0.0268	0.0214
UFSM$_{pair}$ [27]	0.1920	0.2740	0.3264	0.3691	0.0612	0.0370	0.0270	0.0216
FBSM [63]	0.0033	0.0141	0.0191	0.0245	0.0011	0.0013	0.0010	0.0010
FBSM$_{pair}$ [63]	0.0029	0.0135	0.0229	0.0271	0.0010	0.0011	0.0011	0.0009
LVSM$_1$	0.1935	0.2760	0.3335	0.3762	0.0616	0.0375	0.0276	0.0219
LVSM$_2$	0.1918	0.2769	0.3318	0.3693	0.0591	0.0363	0.0266	0.0211
LVSM$_3$	0.1923	**0.2805**	**0.3379**	**0.3834**	0.0615	**0.0379**	**0.0280**	**0.0223**

We organize the discussion of the results by dataset. We first look at the Beauty dataset. We note that LVSM$_3$ dominates the performance on all metrics. The second best results are achieved by coSim, PFW and UFSM. The improvement of LVSM$_3$ over the second best is significant in terms of *Rec*@10, *DCG*@5, *DCG*@10, *DCG*@15 and *DCG*@20. This demonstrates the superiority of FSMs for item cold-start top-N recommendation. This also shows the power of LVSM as it significantly improves over the state-of-the-art FSMs.

Next, we look at the Games dataset, which shows similar results. The difference is that FBSM performs the second best for this task and FBSM achieves a comparable performance as LVSM$_2$ on *Rec*@5. The Games dataset has the sparsest item feature but least sparse ratings of all Amazon datasets. This characteristic of the Games dataset benefits FBSM as it models the interaction among features to overcome feature sparsity while the least sparse rating helps it to learn feature interactions. LVSM can also benefit from the characteristics of the Games dataset as the modeling of global item similarities captures the feature interaction in a more advanced way. The improvement of LVSM over FBSM is significant on *Rec*@10, *Rec*@20, *DCG*@10, *DCG*@15 and *DCG*@20.

We turn to the Sports dataset. LVSM generally has an advantage, but is outperformed by CVAE in terms of *Rec*@5 and *DCG*@5. As shown in Table 2, the ratings of the Sports dataset are the sparsest among all the datasets that we consider. LVSM and CVAE show their advantage of utilizing a VAE by benefiting from the automatic denoising property of the VAE. While CVAE shows promising results when $N = 5$, the effectiveness of CVAE drops as N increases. Also, a comparison between CVAE and NSPR reveals that a VAE is a better tool for extracting information from raw features than DNN. The superiority of LVSM is well demonstrated on Sports when N is getting larger. The improvement of LVSM over the second best approach is significant in terms of *Rec*@10, *Rec*@15, *Rec*@20 and *DCG*@20. With insufficient label information, methods that better extract information from item feature will show their advancement on Sports.

Next, we consider CUL-a. As CiteULike has better formatted features than the Amazon datasets to measure item similarity, we can expect a better performance achieved by FSMs. Surprisingly, although FSMs perform better than other methods, it actually fails to beat the non-collaborative filtering method coSim. A possible explanation is that the features of CUL-a are well-qualified to capture item similarities, where existing FSMs reached a bottleneck to further improve the performance, due to the sparsity of ratings. However, LVSM has the ability to improve the performance over coSim by a large margin. LVSM_2 is significantly better than coSim in terms of *Rec*@5 and *Rec*@10, and LVSM_3 is significantly better than coSim in terms of every DCG metric.

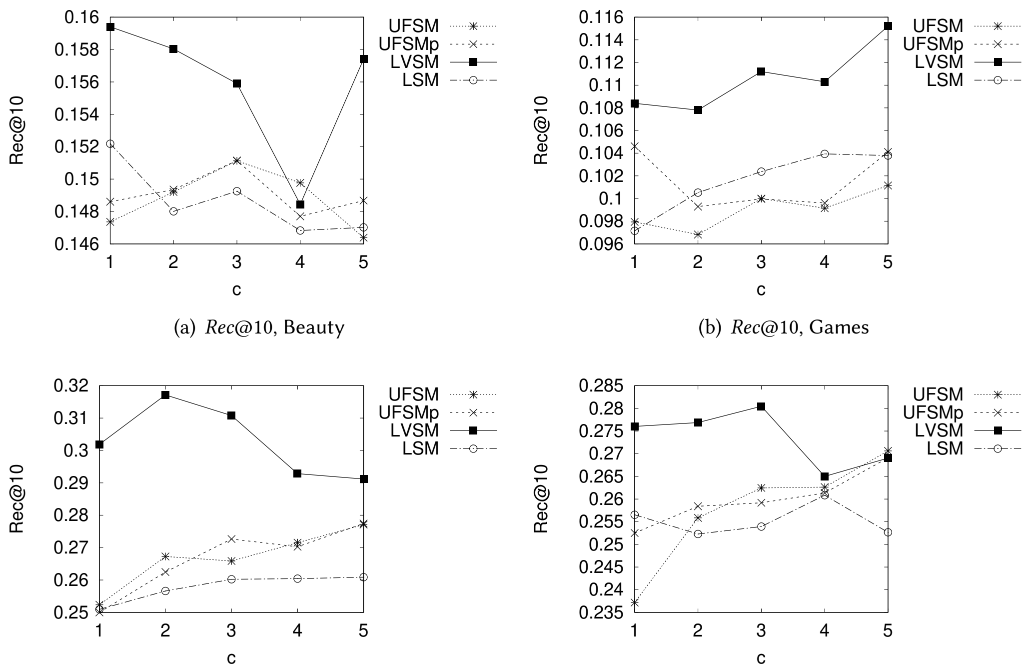
Finally, we look at CUL-t. While the performance of LVSM is very promising on CUL-a, it cannot significantly improve the performance on CUL-t. LVSM is outperformed by coSim in terms of *Rec*@5 and *DCG*@5, and achieves a tie with coSim in terms of *DCG*@10. The improvements of LVSM over coSim in terms of *Rec*@10, *Rec*@15, *Rec*@20, *DCG*@10, *DCG*@15 and *DCG*@20 is not significant. CUL-t has the sparsest features among all datasets. All methods except coSim all include learning, which is heavily impacted by the sparsity of features. Although the performance of LVSM is not exceptional, it actually shows a good denoising ability as the performance generally is at least as good as that of coSim. In comparison, other methods, especially FBSM, perform much worse.

To summarize, LVSM has generally shown its superiority over other methods on all datasets. Except on CUL-a, the improvement of LVSM over the second best method is usually significant. On the other hand, FSMs shows a better performance than other types of method (in other categories), for the task of item cold-start top-N recommendation. CVAE enjoys the benefits of VAE for denoising with sparse features, compared with NSPR. However, as it belongs to IFM, which is not designed for top-N recommendation task, it fails to perform well, especially on CUL-a and CUL-t. LVSM takes advantage of both FSM and VAE to yield overall better performance.

7.2 Effect of modeling global and local similarities

We seek to answer RQ2, whether modeling global and local item similarities helps to improve performance. We form another baseline LSM from LVSM, which calculates local similarities only. We also compare LVSM with UFSM, which calculates global similarities only. We summarize LVSM, LSM and UFSM in Table 6. We vary the number of user groups c and plot the *Rec*@10 scores obtained by LVSM, LSM, UFSM and UFSM_{pair}; see Fig. 3.

Fig. 3(a) displays the results on the Beauty dataset. UFSM, UFSM_{pair} reach their peak performance when learning 3 global similarity functions. LVSM and LSM generally decrease their performance when modeling more local similarity functions. LSM is outperformed by UFSM, UFSM_{pair} when $c \geq 2$ and LVSM is outperformed by UFSM_{pair} when $c = 4$. In short, modeling global similarity functions only achieves the best performance, which shows that there may not exist user subgroups



(a) *Rec*@10, Beauty (b) *Rec*@10, Games

(c) *Rec*@10, CUL-a (d) *Rec*@10, CUL-t

Fig. 3. Effect of the number of similarity functions on *Rec*@10.

Table 6. Different ways of modeling item similarity functions.

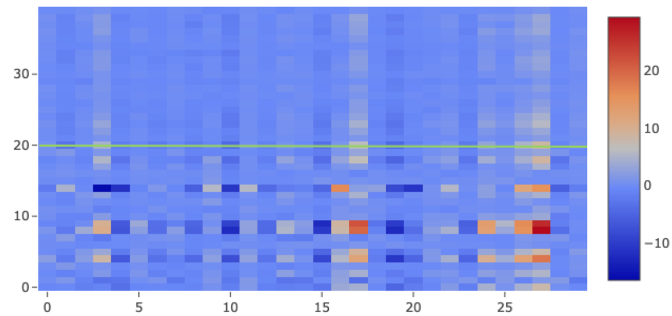
Method	LVSM	LSM	UFSM
Global similarity	1	0	c
Lobal similarity	c	c	0

on the Beauty dataset. LVSM also shows better modeling capacity of global item similarities than UFSM.

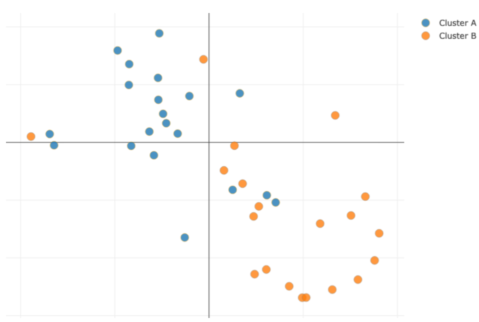
Fig. 3(b) shows a converse result. LVSM and LSM increase their performance by modeling local similarity functions. When $c = 5$, LVSM achieves its best performance, although the figure of LSM drops slightly. LSM outperforms UFSM and UFSM$_{pair}$ when $c \geq 2$ and LVSM further evidently improves over LSM. In short, learning local item similarity functions well captures user subsets in the Games dataset. The advantage of LVSM over UFSM is better illustrated as UFSM fails to model local similarities.

Fig. 3(c) further demonstrates the suitability of learning item similarities with LVSM. Although LSM is outperformed by UFSM and UFSM$_{pair}$, LVSM outperforms UFSM and UFSM$_{pair}$, and the best performance is achieved when $c = 2$. We can conclude from Fig. 3(c) that while solely modeling of local similarities is sub-optimal, the integration of modeling local and global similarities well captures the essence of the application on CUL-a.

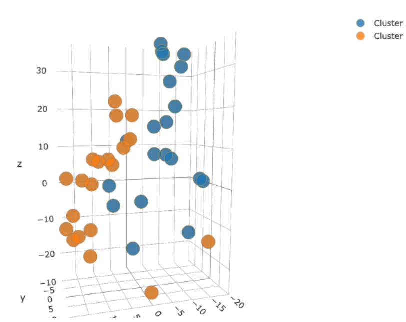
The results in Fig. 3(d) are similar. LVSM shows better results when $c \leq 3$. UFSM and UFSM$_{pair}$ catch up when c is increased. Note that while LSM achieves the best performance at $c = 4$, that setting is where LVSM actually generates its worst recommendation performance. This further



(a) Example of rating predictions. Above the green line are the predicted ratings for users in group A and the ratings below the green line are for group B



(b) TSNE with 2 components



(c) TSNE with 3 components

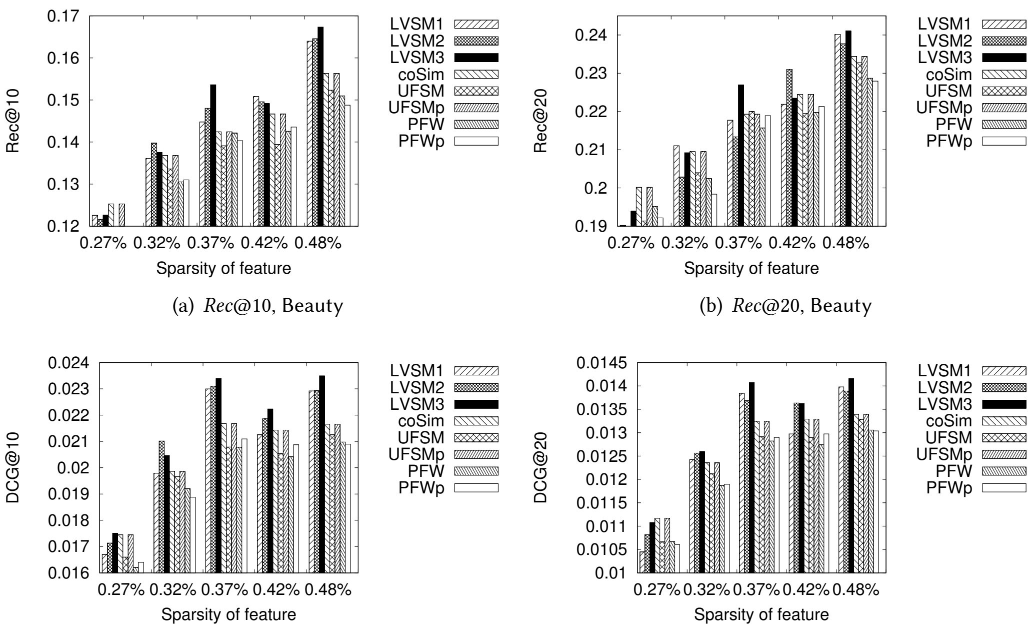
Fig. 4. Qualitative examples showing how LVSM captures user groups.

confirms the effectiveness of jointly modeling local and global similarities, which is the advantage of bayesian graphical modeling.

The estimation of local models is essential to the performance of LVSM. As each local model corresponds to a user group, it will be interesting to see what the learned user group looks like. Therefore, we provide a qualitative example in Fig. 4, using the Games dataset. For the sake of obtaining a clear visualization, we consider two user groups only. We visualize the predicted scores $\tilde{y}_{ui}$ of users over new items. This is because users in the same group have similar behaviors, whereas users from different groups have different behaviors. We randomly select 30 items from all new items. For each group, we randomly select 20 users.

We visualize the predicted ratings for the 20 items in Fig. 4(a). Clearly, users from different groups show different behaviors, illustrated by the different ratings given to the same items. Users from group A generally give lower ratings to items, compared with users from group B. Besides, similarities are clearly visible for users in group A, whereas behaviors of users from group B show some difference. We can also see the commonality in behaviors from both groups, which reflects the effect of the global similarity function.

We also visualize the spatial proximity by conducting t-SNE [48] on the predicted rating scores. t-SNE identifies 2 and 3 main components, depicted in Fig. 4(b) and Fig. 4(c), respectively. Users from the two groups can be clustered with clear and different centroids.



(a) *Rec*@10, Beauty

(b) *Rec*@20, Beauty

(c) *DCG*@10, Beauty

(d) *DCG*@20, Beauty

Fig. 5. Effect of feature sparsity on the performance of methods on the Beauty dataset.

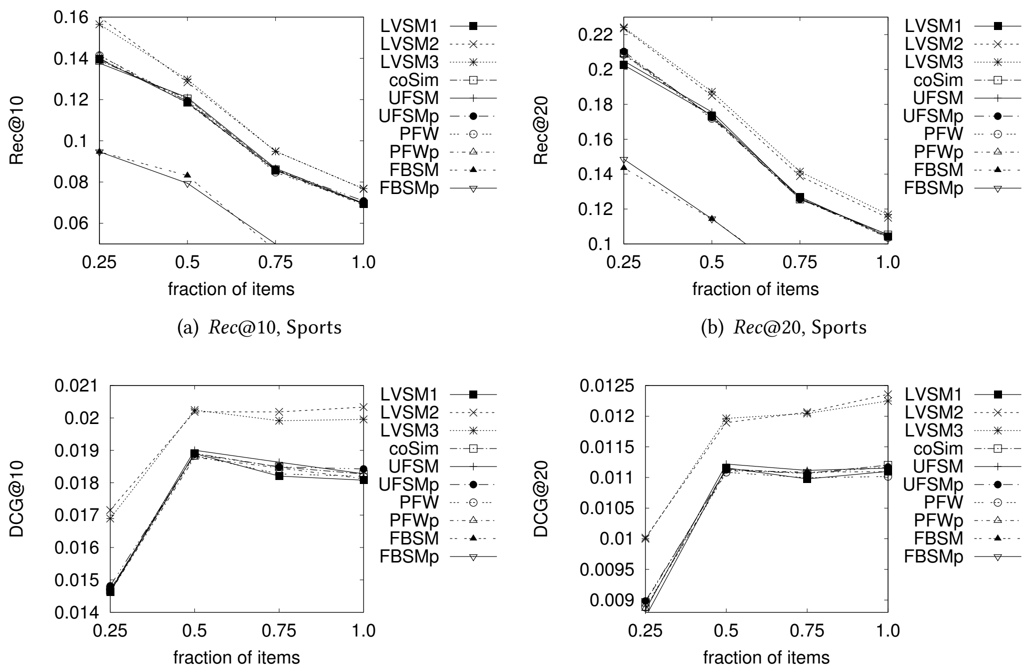
7.3 Effect of feature sparsity

We proceed to answer RQ3. We evaluate the effect of feature sparsity on the performance of recommenders. We manually sparsify item features, by randomly selecting dimensions in the feature to be excluded, where the feature density is roughly controlled as 0.27%, 0.32%, 0.37%, 0.42%, 0.48%, which are respectively 50%, 60%, 70%, 80%, 90% to the original density of Beauty dataset. As we have already demonstrated the superiority of models in the FSM category over models in the UM, LFM and IFM categories, we only care about the impact of feature sparsity on FSMs, i.e., coSim, PFW, UFSM and LVSM. Note that we also exclude FBSM for comparison as it generates very poor recommendations when item features are even sparser. For illustration, we depict the results of a comparison in terms of *Rec*@10, *Rec*@20, *DCG*@10 and *DCG*@20 on the Beauty dataset in Fig. 5.

As shown in Fig. 5(a), coSim and PFW outperform LVSM when *Feature density* = 0.27%. This is understandable: when item feature is extremely sparse, we have less information from data so that the simple models generally perform better, e.g., coSim. LVSM performs better when the item feature sparsity is 0.32%, 0.37%, 0.42%, 0.48%, respectively, where other models also surpass coSim and PFW. This shows that LVSM can overcome feature sparsity to a certain degree. When it is extremely sparse, we should turn to simpler models.

Similar results are shown for *Rec*@20 in Fig. 5(b), where the superiority of LVSM is shown when item features are not extremely sparse. It is also interesting to see that while LVSM_2 generally shows less effective results than LVSM_1 and LVSM_3, LVSM_2 outperforms LVSM_1 and LVSM_3 when *feature sparsity* = 0.42%. It seems that feature sparsity can also affect the number of local similarity functions in the data.

Next we look at the results in terms of DCG. While LVSM loses out to simple models when *feature sparsity* = 0.27% in terms of *Rec*@10 and *Rec*@20; it wins back in terms of *DCG*@10,



(a) *Rec*@10, Sports

(b) *Rec*@20, Sports

(c) *DCG*@10, Sports

(d) *DCG*@20, Sports

Fig. 6. Effect of the fractions of cold-start items on the performance of methods on the Sports dataset.

as shown in Fig. 5(c). LVSM outperforms other methods with other degrees of feature sparsity. Interestingly, LVSM_1 and LVSM_2 perform even better when *feature sparsity* = 0.37% than that when *feature sparsity* = 0.48% and when *feature sparsity* = 0.42%, the performance actually degrades. We think that the dataset formed by controlling sparsity at 0.42% ignores some important features. The 0.37%-sparsity dataset might preserve these important feature and ignore some noisy features, which works similarly as feature selection.

While Fig. 5(d) shows similar results for DCH@20; LVSM is again outperformed by simple models when *feature sparsity* = 0.27%. Similar to Fig. (5(b)), LVSM_2 also outperforms LVSM_1 and LVSM_3 when *feature sparsity* = 0.42%, which further demonstrate the impact of feature sparsity on the number of local similarity functions to model.

7.4 Effect of item cold-start

Next, we turn to RQ4. We evaluate the effect of the fraction of cold-start items on the performance of top-N recommenders. The construction of cold-start items follows exactly what is described in Section 6.3. We evaluate the performance of recommending top-N new items when we have different numbers of new items. We set different fractions of items to be cold-start items: we split Y into Y_{train}, Y_{test}, where Y_{train} contains 5/7 items and Y_{test} contains 2/7 items. By training the different methods on Y_{train} given the tuned parameters (Table 4), we test the performance of the trained model over different test set, with respectively 25%, 50%, 75% and 100% columns of Y_{test}, e.g., 1/14, 1/7, 3/14 and 2/7 items. As before, we only consider the effect on FSMs. We report the result of *Rec*@10, *Rec*@20, *DCG*@10, *DCG*@20, respectively in Fig. 6(a)–6(d).

A general trend revealed by Fig. 6 is that the performance in terms of Recall decreases with the growth of the number of cold-start items. If we increase the number of cold-start items, the number of relevant items also increases, causing further difficulty for recommenders to identify all the relevant items. Inversely, DCG shows an increasing trend; when the number of relevant items increases, it is more likely that the relevant items appear in the recommendation list.

As shown by Fig. 6(a) and 6(b), based on their performance in terms of Recall, the methods are generally categorized into three clusters. The first cluster consists of LVSM_2 and LVSM_3. The second cluster contains LVSM_1, coSim, UFSM, UFSM_{pair}, PFW and PFW_{pair}, which are inferior to the first cluster, but also provide good recommendations. The third cluster includes only FBSM and FBSM_{pair}, which is far behind the performance of the second cluster.

Unlike the performance in terms of Recall, over DCG the methods naturally cluster into two clusters, as shown by Fig. 6(c) and 6(d). Besides LVSM_2 and LVSM_3 in the first cluster, other methods are all contained in the second cluster.

In short, LVSM beats other methods on all occasions of cold-start items. The superiority of LVSM is demonstrated by jointly modeling local and global similarity functions of items (LVSM_1 models only global similarities).

7.5 Performance on a large-scale dataset

And, finally, we turn to RQ5. To show the scalability of LVSM, we run experiments on the Kindle dataset. We compare LVSM with other FSMs. We exclude the comparison with other baselines since FSMs already show superior performance (Section 7.1). As training on the Kindle dataset is time-consuming, we only partially explore the parameter space. We tune λ for PFW from $0.1, 0.2, \ldots, 1$; c for UFSM from $1, 2, \ldots, 10$; k for FBSM from $1, 5, 10, 20$. Similarly, we train UFSM, FBSM and PFW with both point-wise and pair-wise losses. For LVSM, we explore LVSM_1, LVSM_2 and LVSM_3. For a fair comparison, we fix other parameters that are used for regularization to 0.1. We train every methods for 50 epochs and at most 24 hours. We save the model after each epoch and use the one with the highest *Rec*@10 on the validation set to evaluate on the test set.

We report results with $N = 10, 20$ in Table 7. As shown in Table 7, LVSM_1, LVSM_2 and LVSM_3 significantly and substantially outperform other FSMs. LVSM_2 outperforms LVSM_1 while LVSM_3 further improves over LVSM_2.[2] The effectiveness of LVSM is further confirmed on large-scale datasets, showing a bigger improvement than on other datasets. The potential reason is that users' behaviors are highly diversified on the large-scale dataset, where estimating local models benefit more from the diversity. The large-scale dataset also contains more noise in item features, and here the de-noising property of the VAE is especially useful.

We also report the number of parameters and the time for training for each epoch in Table 7. PFW and PFW_{pair} have the largest number of parameters as they will learn for each user a separate set of parameters. Therefore, they are also time-consuming to train. In comparison, UFSM and FBSM are efficient to train due to the small number of parameters. While LVSM has 3–5 times more parameters than UFSM and FBSM, this number is a hundred times less than for PFW. LVSM is also efficient to train during the M-step, which is comparable to UFSM and FBSM. However, the E-step is a bit time-consuming. While the training time for M-step remains the same, it grows substantially when c increased to 3 for the E-step.

In short, LVSM shows good performance on large-scale datasets, both in terms of effectiveness and efficiency.

[2] We can expect even better performances by estimating more local models. We leave further investigations for future work.

Table 7. Performance of recommending top-N new items on the Kindle dataset.

Method	*Rec*@10	*Rec*@20	*DCG*@10	*DCG*@20	E-step (secs.)	M-step/ Train (secs.)	#Params
coSim	0.0747	0.1204	0.0190	0.0160	–	–	–
PFW	0.0320	0.0539	0.0097	0.0081	–	2031	299m
PFW_{pair}	0.0443	0.0732	0.0131	0.0109	–	2031	299m
UFSM	0.0786	0.1146	0.0203	0.0151	–	79	377k
$UFSM_{pair}$	0.0794	0.1252	0.0199	0.0160	–	72	113k
FBSM	0.0358	0.0574	0.0092	0.0076	–	88	237k
$FBSM_{pair}$	0.0262	0.0481	0.0076	0.0067	–	93	124k
$LVSM_1$	0.0894**	0.1369**	0.0232**	0.0183**	430	256	1.18m
$LVSM_2$	0.0925**	0.1425**	0.0242**	0.0191**	412	266	1.22m
$LVSM_3$	**0.0998****	**0.1518****	**0.0258****	**0.0201****	1167	277	1.26m

The best parameters selected for methods: $\lambda = 0.1$ for PFW; $\lambda = 1.0$ for PFW_{pair}; $c = 10$ for UFSM; $c = 3$ for $UFSM_{pair}$; $k = 20$ for FBSM; and $k = 10$ for $FBSM_{pair}$.

8 CONCLUSION

We have revisited the task of recommending top-N new items. We have proposed a LVSM to address this problem by exploiting high-dimensional and sparse item features. Our method is a Bayesian generative model that jointly unifies item representation learning, user clustering, and item collaborative filtering. LVSM can learn deep representations from item features to facilitate similarity measurement. LVSM clusters users into subsets, where a separate similarity function is learned for each subset. To achieve efficiency, we conduct variational inference, and optimize the model through variational EM-algorithms.

Through a broad set of experiments, we have evaluated the efficacy of LVSM. LVSM outperforms state-of-the-art feature-based methods for recommending top-N new items. It provides robust recommendations independent of the quality of item features. It generates good performance in extreme cases, e.g., with a large fraction of new items or with extremely sparse features. It also demonstrates an effective performance on the large-scale dataset.

As to limitations of LVSM, the model is more complicated than other feature-based similarity models. On an extremely sparse dataset such as CUL-t, LVSM still outperforms other methods on most metrics, but relatively simple methods such as coSim approach its performance.

In future work, we would like to extend LVSM to online settings, where user subsets constantly evolve and the local similarity functions should be learned adaptively. Also, we should improve our understanding of the relation between the sparsity of the dataset and the performance of LVSM.

REFERENCES

[1] Deepak Agarwal and Bee-Chung Chen. 2009. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '09)*. ACM, Paris, France, 19–28. https://doi.org/10.1145/1557019.1557029

[2] Fabio Aiolli. 2013. A Preliminary Study on a Recommender System for the Million Songs Dataset Challenge. In *Proceedings of the 4th Italian Information Retrieval Workshop (IIR '13)*. Pisa, Italy, 73–83.

[3] Mohammad Aliannejadi and Fabio Crestani. 2018. Personalized Context-Aware Point of Interest Recommendation. *ACM Trans. Inf. Syst.* 36, 4, Article 45 (Oct. 2018), 28 pages. https://doi.org/10.1145/3231933

[4] Evangelos Banos, Ioannis Katakis, Nick Bassiliades, Grigorios Tsoumakas, and Ioannis P. Vlahavas. 2006. PersoNews: A Personalized News Reader Enhanced by Machine Learning and Semantic Filtering. In *On the Move to Meaningful Internet Systems (OTM '06)*. Springer, Montpellier, France, 975–982. https://doi.org/10.1007/11914853_62

[5] Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Cold-Start Item and User Recommendation with Decoupled Completion and Transduction. In *Proceedings of the 9th ACM Conference*

on Recommender Systems (RecSys '15). ACM, Vienna, Austria, 91–98. https://doi.org/10.1145/2792838.2800196

[6] I. Barjasteh, R. Forsati, D. Ross, A. Esfahanian, and H. Radha. 2016. Cold-Start Recommendation with Provable Guarantees: A Decoupled Approach. *IEEE Trans. Knowl. Data Eng.* 28, 6 (June 2016), 1462–1474. https://doi.org/10.1109/TKDE.2016.2522422

[7] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect Based Recommendations: Recommending Items with the Most Valuable Aspects Based on User Reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '17)*. ACM, Halifax, NS, Canada, 717–725. https://doi.org/10.1145/3097983.3098170

[8] Alex Beutel, Ed Huai-hsin Chi, Zhiyuan Cheng, Hubert Pham, and John R. Anderson. 2017. Beyond Globally Optimal: Focused Learning for Improved Recommendations. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. Perth, Australia, 203–212. https://doi.org/10.1145/3038912.3052713

[9] Daniel Billsus and Michael J Pazzani. 1999. A Hybrid User Model for News Story Classification. In *Proceedings of the 7th International Conference on User Modeling (UM '99)*. Springer, Banff, Canada, 99–108.

[10] Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu, and Shaoping Ma. 2019. An Efficient Adaptive Transfer Neural Network for Social-aware Recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19)*. ACM, Paris, France, 225–234. https://doi.org/10.1145/3331184.3331192

[11] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender Systems based on User Reviews: The State of the Art. *User Model. User-Adapt. Interact.* 25, 2 (Jun 2015), 99–154. https://doi.org/10.1007/s11257-015-9155-5

[12] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. 2012. SVDFeature: A Toolkit for Feature-based Collaborative Filtering. *J. Mach. Learn. Res.* 13 (2012), 3619–3622.

[13] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to Rank Features for Recommendation over Multiple Categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, Pisa, Italy, 305–314. https://doi.org/10.1145/2911451.2911549

[14] Yifan Chen, Pengjie Ren, Yang Wang, and Maarten de Rijke. 2019. Bayesian Personalized Feature Interaction Selection for Factorization Machines. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19)*. ACM, Paris, France, 665–674. https://doi.org/10.1145/3331184.3331196

[15] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS@RecSys '16)*. ACM, Boston, MA, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[16] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan S. Kankanhalli. 2018. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW '18)*. Lyon, France, 639–648. https://doi.org/10.1145/3178876.3186145

[17] Szu-Yu Chou, Yi-Hsuan Yang, Jyh-Shing Roger Jang, and Yu-Ching Lin. 2016. Addressing Cold Start for Next-song Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, Boston, MA, USA, 115–118. https://doi.org/10.1145/2959100.2959156

[18] Evangelia Christakopoulou and George Karypis. 2016. Local Item-Item Models For Top-N Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, Boston, MA, USA, 67–74. https://doi.org/10.1145/2959100.2959185

[19] Evangelia Christakopoulou and George Karypis. 2018. Local Latent Space Models for Top-N Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (SIGKDD '18)*. ACM, London, UK, 1235–1243. https://doi.org/10.1145/3219819.3220112

[20] Gabriella Contardo, Ludovic Denoyer, and Thierry Artières. 2015. Representation Learning for cold-start recommendation. In *Workshop Track Proceedings of the 3rd International Conference on Learning Representations (ICLR '15)*. San Diego, CA, USA.

[21] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the 4th ACM Conference on Recommender Systems (RecSys '10)*. ACM, Barcelona, Spain, 39–46. https://doi.org/10.1145/1864708.1864721

[22] Manuel de Buenaga Rodríguez, Manuel J. Maña López, Alberto Díaz Esteban, and Pablo Gervás Gómez-Navarro. 2001. A User Model Based on Content Analysis for the Intelligent Personalization of a News Service. In *Proceedings of the 8th International Conference on User Modeling (UM '01)*. Sonthofen, Germany, 216–218. https://doi.org/10.1007/3-540-44566-8_25

[23] Mukund Deshpande and George Karypis. 2004. Item-based top-N Recommendation Algorithms. *ACM Trans. Inf. Syst.* 22, 1 (Jan. 2004), 143–177. https://doi.org/10.1145/963770.963776

[24] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin, and Jiajie Yu. 2018. Improving Implicit Recommender Systems with View Data. In *Proceedings of the 27th International Joint Conference on*

Artificial Intelligence (IJCAI '18). Stockholm, Sweden, 3343–3349. https://doi.org/10.24963/ijcai.2018/464

[25] Travis Ebesu and Yi Fang. 2017. Neural Semantic Personalized Ranking for item cold-start recommendation. *Inf. Retr. Journal* 20, 2 (2017), 109–131. https://doi.org/10.1007/s10791-017-9295-9

[26] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2014. Active Learning Strategies for Rating Elicitation in Collaborative Filtering: A System-wide Perspective. *ACM Trans. Intell. Syst. Technol.* 5, 1, Article 13 (Jan. 2014), 33 pages. https://doi.org/10.1145/2542182.2542195

[27] Asmaa Elbadrawy and George Karypis. 2015. User-Specific Feature-Based Similarity Models for Top-n Recommendation of New Items. *ACM Trans. Intell. Syst. Technol.* 6, 3, Article 33 (April 2015), 20 pages. https://doi.org/10.1145/2700495

[28] Rana Forsati, Mehrdad Mahdavi, Mehrnoush Shamsfard, and Mohamed Sarwat. 2014. Matrix Factorization with Explicit Trust and Distrust Side Information for Improved Social Recommendation. *ACM Trans. Inf. Syst.* 32, 4, Article 17 (Oct. 2014), 38 pages. https://doi.org/10.1145/2641564

[29] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM '10)*. IEEE, Sydney, Australia, 176–185. https://doi.org/10.1109/ICDM.2010.129

[30] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, and Depeng Jin. 2019. Neural Multi-task Recommendation from Multi-behavior Data. In *Proceedings of the 35th IEEE International Conference on Data Engineering (ICDE '19)*. IEEE, Macao, China, 1554–1557. https://doi.org/10.1109/ICDE.2019.00140

[31] Thomas George and Srujana Merugu. 2005. A Scalable Collaborative Filtering Framework Based on Co-Clustering. In *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM '05)*. IEEE, Houston, Texas, USA, 625–628. https://doi.org/10.1109/ICDM.2005.14

[32] Kostadin Georgiev and Preslav Nakov. 2013. A non-IID Framework for Collaborative Filtering with Restricted Boltzmann Machines. In *Proceedings of the 30th International Conference on Machine Learning (ICML '13)*. Atlanta, GA, USA, 1148–1156.

[33] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS '11)*. Fort Lauderdale, USA, 315–323.

[34] Xinyu Guan, Zhiyong Cheng, Xiangnan He, Yongfeng Zhang, Zhibo Zhu, Qinke Peng, and Tat-Seng Chua. 2019. Attentive Aspect Modeling for Review-Aware Recommendation. *ACM Trans. Inf. Syst.* 37, 3, Article 28 (March 2019), 27 pages. https://doi.org/10.1145/3309546

[35] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '19)*. ACM, Shinjuku, Tokyo, Japan, 355–364. https://doi.org/10.1145/3077136.3080777

[36] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*. IEEE, Pisa, Italy, 263–272. https://doi.org/10.1109/ICDM.2008.22

[37] Mohsen Jamali and Laks V. S. Lakshmanan. 2013. HeteroMF: recommendation in heterogeneous information networks using context dependent factor models. In *Proceedings of the 22nd International World Wide Web Conference (WWW '13)*. Rio de Janeiro, Brazil, 643–654. https://doi.org/10.1145/2488388.2488445

[38] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* 20, 4 (Oct. 2002), 422–446. https://doi.org/10.1145/582415.582418

[39] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: factored item similarity models for top-N recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '13)*. ACM, Chicago, IL, USA, 659–667. https://doi.org/10.1145/2487575.2487589

[40] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR '14)*. Banff, AB, Canada.

[41] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '08)*. ACM, Las Vegas, Nevada, USA, 426–434. https://doi.org/10.1145/1401890.1401944

[42] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *Proceedings of the 23rd International World Wide Web Conference (WWW '14)*. Seoul, Republic of Korea, 85–96. https://doi.org/10.1145/2566486.2567970

[43] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. 2016. LLORMA: Local Low-Rank Matrix Approximation. *J. Mach. Learn. Res.* 17 (2016), 15:1–15:24.

[44] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM '15)*. ACM, Melbourne, VIC, Australia, 811–820. https://doi.org/10.1145/2806416.2806527

[45] Xiaopeng Li and James She. 2017. Collaborative Variational Autoencoder for Recommender Systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '17)*. ACM, Halifax,

NS, Canada, 305–314. https://doi.org/10.1145/3097983.3098077
[46] Nathan Nan Liu, Xiangrui Meng, Chao Liu, and Qiang Yang. 2011. Wisdom of the better few: cold start recommendation via representative based rating elicitation. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys '11)*. ACM, Chicago, IL, USA, 37–44. https://doi.org/10.1145/2043932.2043943
[47] Hao Ma, Tom Chao Zhou, Michael R. Lyu, and Irwin King. 2011. Improving Recommender Systems by Incorporating Social Contextual Information. *ACM Trans. Inf. Syst.* 29, 2, Article 9 (April 2011), 23 pages. https://doi.org/10.1145/1961209.1961212
[48] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data Using t-SNE. *J. Mach. Learn. Res.* 9, Nov (2008), 2579–2605.
[49] Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, Hong Kong, China, 165–172. https://doi.org/10.1145/2507157.2507163
[50] Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. 2002. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Proceedings of the 18th National Conference on Artificial Intelligence and 14th Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI '02)*. Edmonton, Alberta, Canada, 187–192.
[51] Xia Ning and George Karypis. 2011. SLIM: Sparse Linear Methods for Top-N Recommender Systems. In *Proceedings of the 11th IEEE International Conference on Data Mining (ICDM '11)*. IEEE, Vancouver, BC, Canada, 497–506. https://doi.org/10.1109/ICDM.2011.134
[52] Uros Ocepek, Joze Rugelj, and Zoran Bosnic. 2015. Improving Matrix Factorization Recommendations for Examples in Cold Start. *Expert Syst. Appl.* 42, 19 (2015), 6784–6794. https://doi.org/10.1016/j.eswa.2015.04.071
[53] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. 2008. One-Class Collaborative Filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*. IEEE, Pisa, Italy, 502–511. https://doi.org/10.1109/ICDM.2008.16
[54] Seung-Taek Park and Wei Chu. 2009. Pairwise preference regression for cold-start recommendation. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys '09)*. ACM, New York, NY, USA, 21–28. https://doi.org/10.1145/1639714.1639720
[55] Stefan Pero and Tomás Horváth. 2013. Opinion-Driven Matrix Factorization for Rating Prediction. In *Proceedings of the 21st International Conference on User Modeling, Adaptation, and Personalization (UMAP '13)*. Rome, Italy, 1–13. https://doi.org/10.1007/978-3-642-38844-6_1
[56] Alexandrin Popescul, Lyle H. Ungar, David M. Pennock, and Steve Lawrence. 2001. Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI '01)*. Seattle, Washington, USA, 437–444.
[57] Ian Porteous, Arthur U. Asuncion, and Max Welling. 2010. Bayesian Matrix Factorization with Side Information and Dirichlet Process Mixtures. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*. Atlanta, Georgia, USA.
[58] Steffen Rendle. 2010. Factorization Machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM '10)*. IEEE, Sydney, Australia, 995–1000. https://doi.org/10.1109/ICDM.2010.127
[59] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference (WWW '10)*. Hong Kong, China, 285–295. https://doi.org/10.1145/371920.372071
[60] Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, Foster City, Silicon Valley, CA, USA, 89–96. https://doi.org/10.1145/2645710.2645751
[61] Yanir Seroussi, Fabian Bohnert, and Ingrid Zukerman. 2011. Personalised rating prediction for new users using latent factor models. In *Proceedings of the 22nd ACM Conference on Hypertext and Hypermedia (HT'11)*. ACM, Eindhoven, The Netherlands, 47–56. https://doi.org/10.1145/1995966.1995976
[62] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and J. C. Mao. 2016. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '16)*. ACM, San Francisco, CA, USA, 255–262. https://doi.org/10.1145/2939672.2939704
[63] Mohit Sharma, Jiayu Zhou, Junling Hu, and George Karypis. 2015. Feature-based factorized Bilinear Similarity Model for Cold-Start Top-n Item Recommendation. In *Proceedings of the 15th SIAM International Conference on Data Mining (SDM '15)*. SIAM, Vancouver, BC, Canada, 190–198. https://doi.org/10.1137/1.9781611974010.22
[64] Lei Shi, Wayne Xin Zhao, and Yi-Dong Shen. 2017. Local Representative-Based Matrix Factorization for Cold-Start Recommendation. *ACM Trans. Inf. Syst.* 36, 2, Article 22 (Aug. 2017), 28 pages. https://doi.org/10.1145/3108148
[65] Ajit Paul Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '08)*. ACM, Las

Vegas, Nevada, USA, 650–658. https://doi.org/10.1145/1401890.1401969

[66] Ian Soboroff and Charles Nicholas. 1999. Combining Content and Collaboration in Text Filtering. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99)*. Stockholm, Sweden, 86–91.

[67] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-Boosted Latent Topics: Understanding Users and Items with Ratings and Reviews. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI '16)*. New York, NY, USA, 2640–2646.

[68] Michele Trevisiol, Luca Maria Aiello, Rossano Schifanella, and Alejandro Jaimes. 2014. Cold-start news recommendation with domain-dependent browse graph. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, Foster City, Silicon Valley, CA, USA, 81–88. https://doi.org/10.1145/2645710.2645726

[69] Aäron van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Proceedings of the 27th Advances in Neural Information Processing Systems (NIPS '13)*. Lake Tahoe, Nevada, United States, 2643–2651.

[70] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* 11 (2010), 3371–3408.

[71] Maksims Volkovs and Guang Wei Yu. 2015. Effective Latent Models for Binary Feedback in Recommender Systems. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, Santiago, Chile, 313–322. https://doi.org/10.1145/2766462.2767716

[72] Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '11)*. ACM, San Diego, CA, USA, 448–456. https://doi.org/10.1145/2020408.2020480

[73] Hao Wang, Binyi Chen, and Wu-Jun Li. 2013. Collaborative Topic Regression with Social Regularization for Tag Recommendation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI '13)*. Beijing, China, 2719–2725.

[74] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '15)*. ACM, Sydney, NSW, Australia, 1235–1244. https://doi.org/10.1145/2783258.2783273

[75] Keqiang Wang, Wayne Xin Zhao, Hongwei Peng, and Xiaoling Wang. 2016. Bayesian Probabilistic Multi-Topic Matrix Factorization for Rating Prediction. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI '16)*. New York, NY, USA, 3910–3916.

[76] Xinxi Wang and Ye Wang. 2014. Improving Content-based and Hybrid Music Recommendation using Deep Learning. In *Proceedings of the ACM International Conference on Multimedia (MM '14)*. ACM, Orlando, FL, USA, 627–636. https://doi.org/10.1145/2647868.2654940

[77] Y. Wang, X. Lin, L. Wu, and W. Zhang. 2017. Effective Multi-Query Expansions: Collaborative Deep Networks for Robust Landmark Retrieval. *IEEE Trans. on Image Processing* 26, 3 (March 2017), 1393–1404. https://doi.org/10.1109/TIP.2017.2655449

[78] Yang Wang, Wenjie Zhang, Lin Wu, Xuemin Lin, Meng Fang, and Shirui Pan. 2016. Iterative Views Agreement: An Iterative Low-Rank Based Structured Optimization Method to Multi-View Spectral Clustering. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*. New York, NY, USA, 2153–2159.

[79] Yao Wu, Xudong Liu, Min Xie, Martin Ester, and Qing Yang. 2016. CCCF: Improving Collaborative Filtering via Scalable User-Item Co-Clustering. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining (WSDM '16)*. ACM, San Francisco, CA, USA, 73–82. https://doi.org/10.1145/2835776.2835836

[80] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. In *Proceedings of the 33nd International Conference on Machine Learning (ICML '16)*. New York City, NY, USA, 478–487.

[81] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. 2012. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st World Wide Web Conference (WWW '12)*. Lyon, France, 21–30. https://doi.org/10.1145/2187836.2187840

[82] Chunfeng Yang, Huan Yan, Donghan Yu, Yong Li, and Dah Ming Chiu. 2017. Multi-site User Behavior Modeling and Its Application in Video Recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, Shinjuku, Tokyo, Japan, 175–184. https://doi.org/10.1145/3077136.3080769

[83] Xitong Yang. 2017. Understanding the Variational Lower Bound. https://xyang35.github.io/2017/04/14/variational-lower-bound/.

[84] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. 2014. LCARS: A Spatial Item Recommender System. *ACM Trans. Inf. Syst.* 32, 3, Article 11 (July 2014), 37 pages. https://doi.org/10.1145/2629461

[85] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia Sadiq. 2016. Joint Modeling of User Check-in Behaviors for Real-time Point-of-Interest Recommendation. *ACM Trans. Inf. Syst.* 35, 2, Article 11 (Oct. 2016),

44 pages. https://doi.org/10.1145/2873055

[86] Liang Zhang, Deepak Agarwal, and Bee-Chung Chen. 2011. Generalizing matrix factorization through flexible regression priors. In *Proceedings of the 5th ACM Conference on Recommender Systems (RecSys '11)*. ACM, Chicago, IL, USA, 13–20. https://doi.org/10.1145/2043932.2043940

[87] Weinan Zhang, Tianming Du, and Jun Wang. 2016. Deep Learning over Multi-field Categorical Data - - A Case Study on User Response Prediction. In *Proceedings of the 38th European Conference on IR Research (ECIR '16)*. Padua, Italy, 45–57. https://doi.org/10.1007/978-3-319-30671-1_4

[88] Yi Zhang and Jonathan Koren. 2007. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '07)*. ACM, Amsterdam, The Netherlands, 47–54. https://doi.org/10.1145/1277741.1277752

[89] 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 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '14)*. ACM, Gold Coast , QLD, Australia, 83–92. https://doi.org/10.1145/2600428.2609579

[90] Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma, and Shi Feng. 2013. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *Proceedings of the 22nd International World Wide Web Conference (WWW '13)*. Rio de Janeiro, Brazil, 1511–1520. https://doi.org/10.1145/2488388.2488520

[91] Wayne Xin Zhao, Yanwei Guo, Yulan He, Han Jiang, Yuexin Wu, and Xiaoming Li. 2014. We know what you want to buy: a demographic-based system for product recommendation on microblogs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD '14)*. ACM, New York, NY, USA, 1935–1944. https://doi.org/10.1145/2623330.2623351

[92] W. X. Zhao, S. Li, Y. He, E. Y. Chang, J. Wen, and X. Li. 2016. Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information. *IEEE Trans. Knowl. Data Eng.* 28, 5 (May 2016), 1147–1159. https://doi.org/10.1109/TKDE.2015.2508816

[93] Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint Deep Modeling of Users and Items Using Reviews for Recommendation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, Cambridge, United Kingdom, 425–434. https://doi.org/10.1145/3018661.3018665

[94] Tinghui Zhou, Hanhuai Shan, Arindam Banerjee, and Guillermo Sapiro. 2012. Kernelized Probabilistic Matrix Factorization: Exploiting Graphs and Side Information. In *Proceedings of the 12th SIAM International Conference on Data Mining (SDM '12)*. SIAM, Anaheim, California, USA, 403–414. https://doi.org/10.1137/1.9781611972825.35