Local Factor Models for Large-Scale Inductive Recommendation

Longqi Yang
Microsoft
Redmond, WA, USA
Longqi.Yang@microsoft.com

Tobias Schnabel
Microsoft
Redmond, WA, USA
toschnab@microsoft.com

Paul N. Bennett
Microsoft
Redmond, WA, USA
pauben@microsoft.com

Susan Dumais
Microsoft
Redmond, WA, USA
sdumais@microsoft.com

ABSTRACT

In many domains, user preferences are similar *locally* within like-minded subgroups of users, but typically differ *globally* between those subgroups. Local recommendation models were shown to substantially improve top-K recommendation performance in such settings. However, existing local models do not scale to large-scale datasets with an increasing number of subgroups and do not support inductive recommendations for users not appearing in the training set. Key reasons for this are that subgroup detection and recommendation get implemented as separate steps in the model or that local models are explicitly instantiated for each subgroup. In this paper, we propose an End-to-end Local Factor Model (ELFM) which overcomes these limitations by combining both steps and incorporating local structures through an inductive bias. Our model can be optimized end-to-end and supports incremental inference, does not require a full separate model for each subgroup, and has overall small memory and computational costs for incorporating local structures. Empirical results show that our method substantially improves recommendation performance on large-scale datasets with millions of users and items with considerably smaller model size. Our user study also shows that our approach produces coherent item subgroups which could aid in the generation of explainable recommendations.

KEYWORDS

Recommendation; local model; large-scale; end-to-end

ACM Reference Format:
Longqi Yang, Tobias Schnabel, Paul N. Bennett, and Susan Dumais. 2021. Local Factor Models for Large-Scale Inductive Recommendation. In *Fifteenth ACM Conference on Recommender Systems (RecSys '21), September 27-October 1, 2021, Amsterdam, Netherlands.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3460231.3474276

RecSys '21, September 27-October 1, 2021, Amsterdam, Netherlands

ACM ISBN 978-1-4503-8458-2/21/09…$15.00
https://doi.org/10.1145/3460231.3474276

1 INTRODUCTION

In many real-world settings, even though user preferences differ globally across an entire population, preferences tend to be consistent on a local level for users that are part of a certain subgroup. For example, a senior high school student visits websites for reasons that are likely different from those of a Psychology undergraduate student. However, focusing on a certain subgroup – say the one of senior high school students, we would expect to see large similarities in behavior due to their similar curriculums. These subgroups of like-minded users naturally induce subgroups of items that are likely to be consumed together, e.g., websites on SAT preparation. We will refer to these coupled subgroups of items and users as *co-clusters* throughout this paper. Many of the recommendation algorithms developed to date, from nearest neighbor-based approaches [8, 45] to matrix factorization [29, 44] to deep neural networks [26, 47, 48, 59], do not explicitly model specific subgroup structures and instead estimate a single global model. As a result, these models can be limited in their expressivity of the behavior and preferences of subgroups, especially of subgroups that are small in size or whose consumption patterns substantially deviate from the general population.

To address this problem, prior work developed *local models* for recommendation [9, 14, 15, 33, 34, 40, 49, 53]. The core idea is to train many models locally for subgroups of users and items to complement a global model. A good local recommendation model should also allow for a sufficiently fine-grained representation of these subgroups to not only improve modeling but also support high-fidelity user analysis and interpretability. However, key limitations of existing local models face are (i) their lack of scalability to large real-world datasets, (ii) their inability to support sufficiently many subgroups and (iii) their inability to accommodate unseen users and update existing user profiles efficiently, which is a significant pain point for large-scale online platforms with new user interactions happening in real time. For example, on the web-scale dataset that we present in this paper, RGLSVD [15], a state-of-the-art local recommendation model, requires over 1 billion parameters to represent only 100 subgroups. Recently developed *disentangled models* [40, 49] suffer from a similar problem and the number of clusters that these methods can handle is less than 20. Existing work [9, 14, 15, 33, 34, 53] explored various ways of improving computational efficiency, but these typically rely on simplifications that may degrade recommendation performance, such as precomputing

and keeping subgroup structures fixed, or assigning each user to only one subgroup.

In this paper, we present End-to-end Local Factor Models (ELFM), an approach for large-scale recommendation that can learn local preference and behavior patterns in an end-to-end fashion and can support efficient inductive learning and incremental inference. Going beyond previous local models, we encode both subgroup structures in the user and item space, i.e., co-clusters of users and items. We encode this structure via an explicit inductive bias into the model as opposed to prior work that trains a full-fledged model for each local subgroup. This is achieved through a differentiable latent co-clustering module serving as a middle layer between user-item representations and supervision. The module learns to construct and leverage overlapping and sparse user-item co-clusters without adhoc rules and allows ELFM to capture multi-level patterns with a single integrated and lightweight model.

We evaluate ELFM on three datasets of different sizes and density levels. Our empirical results show that ELFM substantially improves recommendation performance on very large and sparse datasets across users with diverse activity levels, while maintaining competitive performance on the ones with dense user-item interactions. Furthermore, ELFM achieves these performance gains with only 0.5% to 2% of the size of existing local models. To explicitly test the quality of the detected item subgroups, we conducted a novel user study with crowd workers. The results confirm the coherence of the subgroups, which can also potentially serve as a way inspecting and controlling recommendations.

The main contributions of this paper are threefold:

- We present a generic mechanism for recommenders to leverage local consumption patterns with minimal overhead in model complexity, making local modeling feasible for large-scale inductive recommendation and incremental inference.
- We introduce a method for incorporating local patterns in a cohesive and end-to-end fashion without adhoc simplifications. Both user and items are dynamically co-clustered into overlapping subgroups during optimization.
- We conduct a user study and extensive offline experiments to demonstrate our model significantly improves recommendation performance across a diverse range of scenarios.

2 NOTATION AND TASKS

We assume that we are given a set of user-item interactions $\mathcal{M} = \{(u, i)\}$ where user u interacted with item i (e.g., click, visit, revisit, etc.). Let $\mathcal{U}$ be the set of all users, and $\mathcal{I}$ be the set of all items. While our model solves recommendation as a primary task, it does so by simultaneously considering the auxiliary task of user-item co-clustering.

Top-K inductive recommendation. Given the set of items that user u interacted with in the past, $\mathcal{I}_u \subseteq \mathcal{I}$, the task of top-$K$ inductive recommendation is to first learn a function r that predicts a users preferences on all items,

$$s_{ui} = r(\mathcal{I}_u, i), \quad \forall i \in \{1, \ldots, |\mathcal{I}|\} \tag{1}$$

where $s_{ui} \in \mathbb{R}$ denotes user u's preferences for item i, and then find K items with the highest s_{ui} score. To support inductive recommendation, we design function r to share its parameters across users so that it can readily serve unseen users, which is fundamentally different from transductive models that instantiate explicit user-specific embeddings hindering generalization.

User-item co-clustering. The goal of this auxiliary task is to co-cluster all users and items into N overlapping co-clusters such that each co-cluster can be optimized to learn patterns specific to a subset of users and items. We denote user u's and item i's affinities to co-cluster c as $a_u^c \in \mathbb{R}^+$ and $a_i^c \in \mathbb{R}^+$ respectively. The non-negativity of both scores ensure that their absolute values directly reflect tie strength. Under this soft assignment paradigm, any users or items can belong to multiple co-clusters.

3 END-TO-END LOCAL FACTOR MODELS

The design of ELFM is based on the insight that user-item co-clusters drive the dynamics of user-item interactions — a user is more likely to interact with an item if they belong to a similar set of co-clusters. From a modeling perspective, this means that we want to express the preference function s_{ui} of a user u through her latent co-clusters.

Having co-clusters as part of the model enables ELFM to have fine-grained subgroup structures, making it possible to leverage patterns that are local to subsets of users and items. As shown in Figure. 1a, ELFM consists of three main components: (1) an item embedding representing each item i with a dedicated dense item vector e_i, (2) an itemset aggregator module that constructs user representations by aggregating the representations of items that a user interacted with in the past, and (3) a differentiable latent co-clustering (DLCC) module that constructs and leverages user-item co-clusters for predicting the final user preference vector s_u. All components are fully differentiable and can be trained end-to-end. The next sections discuss the details of the two core modules – the itemset aggregator and DLCC modules.

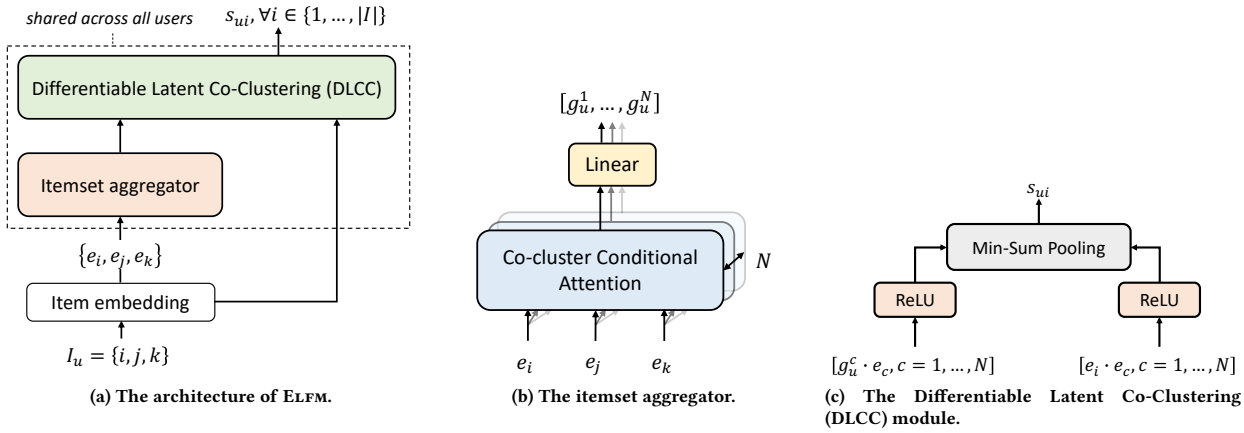
3.1 Itemset aggregator

Given items that a user interacted with ($\mathcal{I}_u$) and their representations ($e_i, i \in \mathcal{I}_u$), the itemset aggregator (Figure. 1b) employs a co-cluster-conditional attention mechanism to construct user representations from them. We employ an N-head dot-product attention approach [46]

$$g_u^c = W \sum_{i \in \mathcal{I}_u} \text{softmax}\left(\frac{h_c^\intercal \cdot e_i}{\sqrt{d}}\right) e_i, \tag{2}$$

where d denotes the dimensionality of item embeddings, and each head h_c builds a co-cluster-conditional user representation $g_u^c, c = 1, \ldots, N$ by assigning higher weights to more predictive items. Compared to a global attention mechanism, co-cluster-conditional attention allows predictive items to vary across co-clusters. In addition, we add a scaling factor $\frac{1}{\sqrt{d}}$ to address the potential vanishing gradients problem [46], and a linear projection ($W \in \mathbb{R}^{d \times d}$) to align the feature spaces of users and items.

Our itemset aggregator comes with a number of advantages compared to having explicit user representations. Beyond eliminating the need to fit such individual user representations, it consumes constant memory with respect to the number of users because the parameters of our itemset aggregator are shared across all users. Hence, our model can be easily scaled to serve millions of users, whereas traditional methods typically scale as $O(|\mathcal{U}|)$. Moreover,



(a) The architecture of ELFM.

(b) The itemset aggregator.

(c) The Differentiable Latent Co-Clustering (DLCC) module.

Figure 1: The architecture of ELFM and its key components. The itemset aggregator builds user profiles from item embeddings and the Differentiable Latent Co-Clustering (DLCC) module models co-cluster memberships. (I_u: items that user u interacted with, e_i: embedding for item i, e_c: embedding for co-cluster c, g_u^c: user u's profiles, s_{ui}: user u's preference score for item i).

the itemset aggregator allows ELFM to support inductive recommendations since inferring g_u^c from scratch only takes two highly parallelizable linear operations. We can further reduce the computational cost by caching a scalar Z_c as

$$Z_c = \sum_{i \in \mathcal{I}_u} \exp\left(\frac{h_c^\intercal \cdot e_i}{\sqrt{d}}\right). \tag{3}$$

and computed embeddings g_u^c. Whenever user u interacts with a new item j, her representation g_u^c can be updated via

$$\begin{aligned} Z_c' &= Z_c + \exp\left(\frac{h_c^\intercal \cdot e_j}{\sqrt{d}}\right) \\ g_u^{c\prime} &= \frac{1}{Z_c'}\left[g_u^c \cdot Z_c + W \cdot \exp\left(\frac{h_c^\intercal \cdot e_j}{\sqrt{d}}\right) e_j\right] \end{aligned} \tag{4}$$

without recomputing representations for items $i \in \mathcal{I}_u$ (Z_c' and $g_u^{c\prime}$ denote the updated values). The computational cost for such updates remains constant and does not grow with the volume of user history. This update step directly supports cases where users intentionally alter their profiles (e.g., item removal).

3.2 Differentiable latent co-clustering

We represent co-clusters of users and items by modeling their affinities [56] to N latent co-clusters (Figure. 1c). Specifically, we learn a representation e_c for each co-cluster c and compute affinity scores a_u^c and a_i^c as

$$\begin{aligned} a_u^c &= \max\left(g_u^{c\,\intercal} \cdot e_c, 0\right) \\ a_i^c &= \max\left(e_i^\intercal \cdot e_c, 0\right) \end{aligned} \tag{5}$$

where we apply rectified linear units (ReLU) [42] to enforce non-negativity of both scores, and share co-cluster representation e_c among all users and items address the potential concern of ReLU blocking gradients. Compared to direct dot products, non-negative scalars a_u^c and a_i^c allow us to directly read off the cluster memberships without post processing. This is useful for a variety of scenarios – for example, one can speed up candidate generation by pruning items from co-clusters that users are not part of, or enable users to steer recommendations through feedback on item clusters (Section 5.2).

Based on a_u^c and a_i^c, we apply a lightweight min-sum pooling operation to link user-item co-clusters scores and recommendations. More specifically, we compute user u's preference score towards item i, s_{ui}, as

$$s_{ui} = \sum_c \min\left(a_u^c, a_i^c\right). \tag{6}$$

where the inner $\min(\cdot, \cdot)$ calculates the amount of co-cluster overlap between u and i in terms of c and can be viewed as a "soft AND" operator that requires both the user and the item have high affinity to a cluster. The outer sum then aggregates such overlaps across all co-clusters. The above design allows us to isolate the contributions that each co-cluster makes to s_u, while also being fully differentiable to enable end-to-end optimization. As a result, each co-cluster can capture local patterns within potentially overlapping groups of users and items. The design of differentiable latent co-clustering encodes our inductive bias that an interaction between a user and an item is driven by their common memberships in latent subgroups.

3.3 Masking-based training loss

We use a softmax classification loss to train ELFM, following prior literature [38] indicating that it is preferable over pointwise losses in binary recommendation because it encourages better weight assignments for top-K rankings. To be able to handle large item vocabulary sizes, we do not compute the full softmax loss but rather compute a sampled version in each step. In addition, we mask out items that the user has already interacted with to prevent the model from learning trivial relationships.

For the set of items $\mathcal{I}_u$ that user u interacted with, we randomly hold out an item $k \in \mathcal{I}_u$ during training time and and feed everything else into the model to compute the representations and scores. Given those, we compute the following loss on a sampled

set $\mathcal{I}_{\text{sp}}$ of negative training examples similar in spirit to the sampled softmax [30]. More formally, the per-user loss being minimized is

$$\mathcal{L}_u(\mathcal{I}_{\text{sp}}, k) = \underbrace{-\alpha \log\left(\frac{s_{uk}}{\sum_{v_i \in \mathcal{I}_{\text{sp}}} s_{ui}}\right)}_{\text{cross entropy}} + \underbrace{\lambda \sum_c \left(a_u^c + \sum_{i \in \mathcal{I}_{\text{sp}}} a_i^c\right)}_{\text{L1 regularization}}, \quad (7)$$

where α is a smoothing term and λ controls the amount of regularization, with higher values of λ increasing sparsity in the user and item subgroups. To create the set of negative samples $\mathcal{I}_{\text{sp}}$, we employ a masking mechanism to prevent trivial solutions (e.g., the identity mapping). More specifically, we exclude items $\mathcal{I}_u$ that the user has already interacted with:

$$\mathcal{I}_{\text{sp}} = \text{Uniform-Sample}\left(\mathcal{I} \setminus \mathcal{I}_u\right) \cup \{k\}. \quad (8)$$

In other words, we do not penalize the model for predictions it makes on positive examples other than k. Also, since user interaction data is known to be incomplete and uncertain [29], we use a label smoothing term α to soften the targets in the cross entropy term. To encourage compact co-cluster structures, we additionally use L1 regularization on the sum of affinity scores across users and items. Eventually, the per-user loss is averaged across a minibatch.

3.4 Model complexity

Method	Memory	Time (Incremental inference)
BPR	$(\lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert)\, d$	$\lvert\mathcal{I}_u\rvert\lvert\mathcal{I}\rvert d$
WRMF	$(\lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert)\, d$	$\lvert\mathcal{I}\rvert d$
NEUMF	$(\lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert)\, d$	$\lvert\mathcal{I}\rvert d$
UCML	$(\lvert\mathcal{U}\rvert + \lvert\mathcal{I}\rvert)\, d$	$\lvert\mathcal{I}_u\rvert\lvert\mathcal{I}\rvert d$
RGLSVD	$(2\lvert\mathcal{U}\rvert + N\lvert\mathcal{I}\rvert)\, d$	$\lvert\mathcal{I}_u\rvert\lvert\mathcal{I}\rvert d$
ELFM	$(2N + \lvert\mathcal{I}\rvert)\, d$	Nd

Table 1: Comparison of model complexity in Big O notation demonstrating ELFM's high memory and time efficiency.

We summarize the memory and time complexity (in Big O notation) of ELFM in Table. 1 and compare them to traditional and local recommendation model baselines (details discussed in Section 4.3). Compared to existing local models (RGLSVD), ELFM consumes substantially less memory ($|\mathcal{I}| \ll N|\mathcal{I}|$), because our model does not require a separate model for each subgroup. Moreover, our model is generally smaller than traditional recommendation models ($N \ll |\mathcal{U}|$), because it does not explicitly store user representations ($O\left(|\mathcal{U}|d\right)$) and cluster assignments ($O\left((|\mathcal{U}| + |\mathcal{I}|)N\right)$). In addition, ELFM is substantially faster for incrementally adding or removing interaction records ($O\left(1\right)$). Altogether, ELFM is particularly suited to leverage local patterns for recommendations at scale.

4 EXPERIMENTAL SETUP

We evaluate ELFM's performance on the two different tasks of Section 2 – the primary task top-K inductive recommendation and the secondary task of user-item co-clustering. For each task, we compare against the set of baselines that are most appropriate for the task at hand.

Dataset	$\lvert\mathcal{U}\rvert$ (users)	$\lvert\mathcal{I}\rvert$ (items)	# entries	density
Web-35M	3,794,691	427,147	34,870,333	$2.2 \cdot 10^{-5}$
LastFM-17M	359,126	87,709	17,423,558	$5.5 \cdot 10^{-4}$
Movielens-10M	136,674	13,681	9,977,455	$5.3 \cdot 10^{-3}$

Table 2: Dataset statistics (density=# entries/$|\mathcal{U}||\mathcal{I}|$).

4.1 Datasets

We evaluate performance on three different datasets whose statistics are summarized in Table. 2. These datasets vary in size, density, and distributions. The Web-35M dataset we collected is the largest, sparsest, and most skewed among the three. LastFM-17M and MovieLens-10M are widely used benchmarking datasets by prior literature [16]. We argue that Web-35M is much more challenging than other existing large scale datasets (e.g., Netflix challenge [5] and Yahoo! music [1]) that have more non-zero entries, because Web-35M covers over 10 to 20 times more users and items, and the resulting interaction matrix is significantly sparser. Below are the details of each dataset.

Web-35M. We collected a large-scale dataset on website revisitation, which is shown to reflect people's long-term information needs [3]. Specifically, we collaborated with the Bing search engine to gather website host visits from a large sample of pseudonymized search logs. Over the period of a month, we tracked the websites that were revisited by each user at least once and then treated revisitation as users' implicit preference, i.e., the number of revisitations was not used. We included hosts that were revisited by at least ten users but were not in the top 30 of the most popular hosts to filter out trivial websites. We also only kept users who had visited at least ten different hosts. Due to the diverse and distributed nature of the web, the resulting Web-35M dataset is extremely sparse.

LastFM-17M. We leveraged the LastFM dataset [11] that records the number of times that 360K listeners play music from artists. We treat non-zero plays as positive feedback and filter users and items based on the criteria adopted for Web-35M with the exception of keeping popular items.

MovieLens-10M. This is derived from MovieLens-20M [24], the largest dataset available from MovieLens platform. We converted the original movie ratings to implicit feedback by treating those greater than or equal to four as positives, following prior literature [28, 44]. After applying the same filters used in LastFM-17M, the resulting MovieLens-10M is significantly richer and denser (more than 200 times than Web-35M).

4.2 Evaluation metrics

4.2.1 Top-K inductive recommendation. To evaluate models' recommendation performance in inductive settings using offline datasets, we hold out **a set of users and their entire interaction history** from each dataset for validation and testing (10K users each for Web-35M, and 10% users each for LastFM-17M and MovieLens-10M). Such a split guarantees that users in the validation and testing sets are not seen during training and is sometimes referred to as strong generalization [41]. For hold-one-out evaluation, the most recent interaction from each user was held out for prediction and the rest were used as inputs. Since timestamps are not available in the

LastFM-17M dataset, a random interaction was chosen. We choose the best performing model on the validation set, and report performance of the chosen model on the test set in terms of Mean Reciprocal Rank (MRR) and Hit Ratio@50 (HR@50) [26].

4.2.2 User-item co-clustering. In addition to explaining user behavior well, a good local recommendation model should group a coherent set of items inside each user-item co-cluster, i.e., related or similar. We follow the intrusion task proposed by Chang et al. [12] to measure coherence via human judgements. Given a set of three items from an item subgroup and a random intruder, people are asked to identify the item that does not belong with the others. Success on this task measures coherence in item subgroups because if subgroups are indeed coherent, it should be easy to spot the intruder as the one item not belonging, but challenging otherwise. We focus on the Web-35M dataset for human evaluation since crowd-workers are unlikely to be able to fully judge movies or songs without substantial training. Figure 2 shows one example task with the intruder being the second item from the top.

We randomly sampled 100 item subgroups from each model, ranked items in descending order of their membership scores (a_i^c) and then chose the items at position 1, 10, and 20 to form the set of three subgroup items. For the intruder, we chose a random intruder that did not appear in the subgroup but possessed a similar popularity as the three subgroup items to control for popularity bias (i.e., picked it within a small range of the average popularity of the subgroup items). Each subgroup was labeled by five different workers. We report overall precision [12] for each method, where we first compute the mean precision for each item subgroup as the percentage of workers that correctly identified the intruder, and then average those across all 100 subgroups.

This user study was reviewed and approved by Microsoft's Institutional Review Board (IRB). All participants gave explicit consent, and the data was kept fully anonymous and was only available to select researchers.

4.3 Baselines

For the top-K inductive recommendation task, we considered the following baselines:

- **Popularity** is a non-personalized baseline that ranks items based on the number of interactions they received in the training set. The comparison to Popularity tells us how much a *personalized* model is able to improve recommendation performance.
- **UserKNN [8, 16]**. Well-tuned nearest-neighbor-based recommenders deliver competitive performance compared to recent deep learning based models [16], especially on dense datasets. We implemented a user-based k-nearest-neighbor recommender by first applying Truncated SVD [23] on binary user-item interaction matrix to derive user representations that enable fast nearest neighbor search. We then compute users' preference scores as

$$s_{ui} = \sum_{u' \in \text{KNN}(u)} \frac{|\mathcal{I}_u \cap \mathcal{I}_{u'}|}{|\mathcal{I}|} \mathbb{1}\left(i \in \mathcal{I}_{u'}\right). \tag{9}$$

- **Bayesian personalized ranking (Bpr)** [44] is a classical pairwise ranking model that learns a dedicated representation for each user and each item by minimizing a logarithmic sigmoid loss. This is a state-of-the-art of pairwise matrix factorization method [25].
- **Weighted regularized matrix factorization (WrMF)** [29] adds a smoothing term to address varied confidence levels associated with users' implicit feedback. It has been a standard MF baseline in the literature.
- **Neural Collaborative Filtering (NeuMF)** [26] is a current recommendation framework based on a multi-layer feed-forward network. It combines the flexibility of neural networks with a traditional MF objective to improve its expressiveness. We choose the GMF architecture as in previous work [26] to make it computationally feasible for strong generalization-based evaluation.
- **uniform Collaborative Metric Learning (uCML)** [28, 50] is a variant of CML [28] with uniformly sampled training triplets. Since uCML embeds users and items into a joint Euclidean distance space, it can be considered as a baseline that naively co-clusters users and items and serves as a conceptually similar baselines to evaluate the added benefits that are due to Elfm's special architecture.
- **Global and Local Singular Value Decomposition with varying ranks (rGLSVD)** [15] is the state-of-the-art local recommendation model that outperforms all prior approaches of its kind [15]. rGLSVD combines a global truncated SVD model with a dedicated local model for each cluster. rGLSVD only groups users, and each user is only assigned to one cluster beforehand using K-means.

Other related but inapplicable methods for our problem include: (1) methods that rely on extra information beyond user-item interactions, e.g., sequential models [31, 52] and graph convolutional neural networks [22] requiring explicit user and item features; (2) methods that cannot support inductive recommendations, e.g., finetuning or retraining very deep neural networks [20, 47–49] in real time. Nevertheless, the baselines we choose (e.g., NeuMF, WRMF, UserKNN) have been shown to achieve competitive and sometimes even better performance [16].

For item coherence, we considered the following baselines:

- ***X*-Clustering**. This method takes a straight-forward approach to detect user-item co-clusters by first using an existing recommendation algorithm X and then clustering the learned user and item embeddings [9]. These representations are learned by deep neural networks or latent matrix factorization. Ideally, one would apply soft clustering (e.g., fuzzy c-means [35]) – however, soft methods need to store cluster assignments for all user and items, which is intractable on large datasets. To mitigate this problem, we instead use k-means to first find cluster centers and then soft-assign entities based on corresponding Euclidean distances. We use this method in combination with Bpr, WrMF and NeuMF as these produced best results on the recommendation task. rGLSVD is not evaluated here as it does not cluster items, and there is no canonical representations of items.

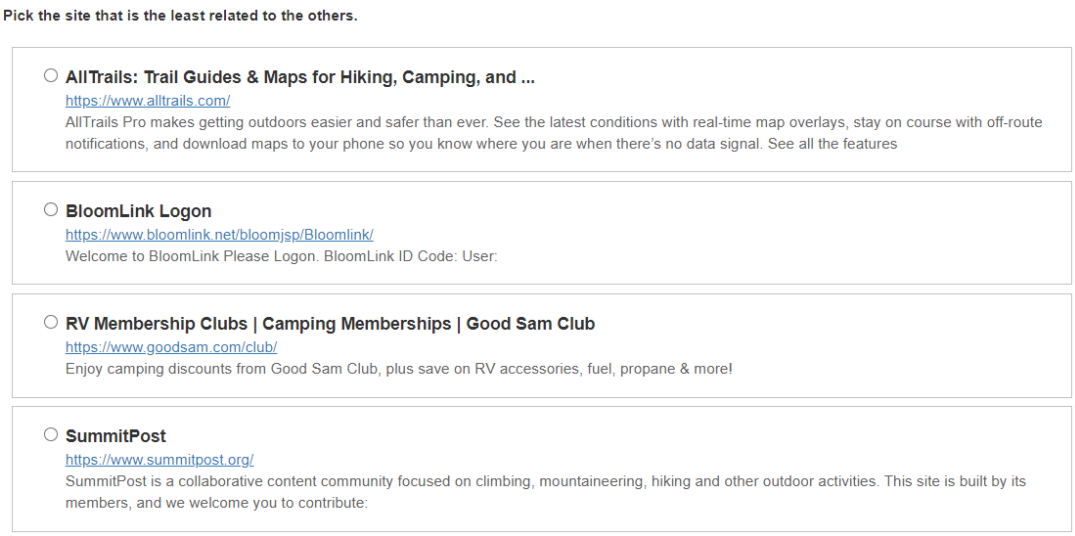


Figure 2: Intrusion task used to measure item coherency within user-item co-clusters. People had to identify the intruder (here: the second item from the top) in a set of four items.

- **Cluster Affiliation Model for Big Networks (BigClam)** is a state-of-the-art algorithm for overlapping community detection at scale [56]. We represent user-item interactions as links in an undirected graph. One inherent limitation of BigClam is that it cannot conduct recommendations or provide user clusters.

We also tried traditional latent topic modeling methods [7] to discover co-clusters, but it fails to capture the correlations from the extremely sparse datasets, and the performance is not comparable to any of the baselines above.

4.4 Implementation details

We trained all models for 150 epochs using Adam [32] with a learning rate of 0.001 and early stopping [10]. That is, the optimal number of training iterations was selected through a validation set. To control for expressive power, we varied dimensionality parameter $d = \{16, 32, 64\}$ for embeddings in Elfm and baselines.

For Elfm, we sampled 1000 negative items for each minibatch with size 1024 and divided the learning rate by 10 for every 50 epochs. The hyperparameters we considered for model selection were as follows: $\alpha \in \{0.1, 0.05\}, \lambda \in \{1e{-}6, 1e{-}4\}$, $N = \{512, 1024, 2048\}$. We implemented Elfm using Tensorflow [2] and trained it on four NVIDIA Tesla P100 GPU cards, taking less than two days to finish.

We implemented baseline models including Bpr, WrMF, NeuMF, and uCML using the OpenRec library [57] and experimented models with different levels of L2 regularization $(0, 1e{-}6, 1e{-}5, 1e{-}4)$. During inference, we froze all model parameters except user embeddings and fine-tuned them for 150 epochs using validation or testing datasets. We implemented rGLSVD using Scikit-learn [43] and followed the original paper [15] to allow the dimensionality of local models to vary. Due to memory constraints, we fixed the number of user clusters for rGLSVD to be 100 which is also the largest size reported by the original paper [15].

For X-clustering, we used *MiniBatchKMeans* in Scikit-learn [43] to jointly group users and items into 2048 clusters. We ran KMeans three times and picked best results according to inertia. For BigClam, the original implementation and recommended parameter settings [56] were adopted to detect 2048 communities.

5 RESULTS AND DISCUSSION

We now present and discuss the results of our empirical evaluation, starting with recommendation performance.

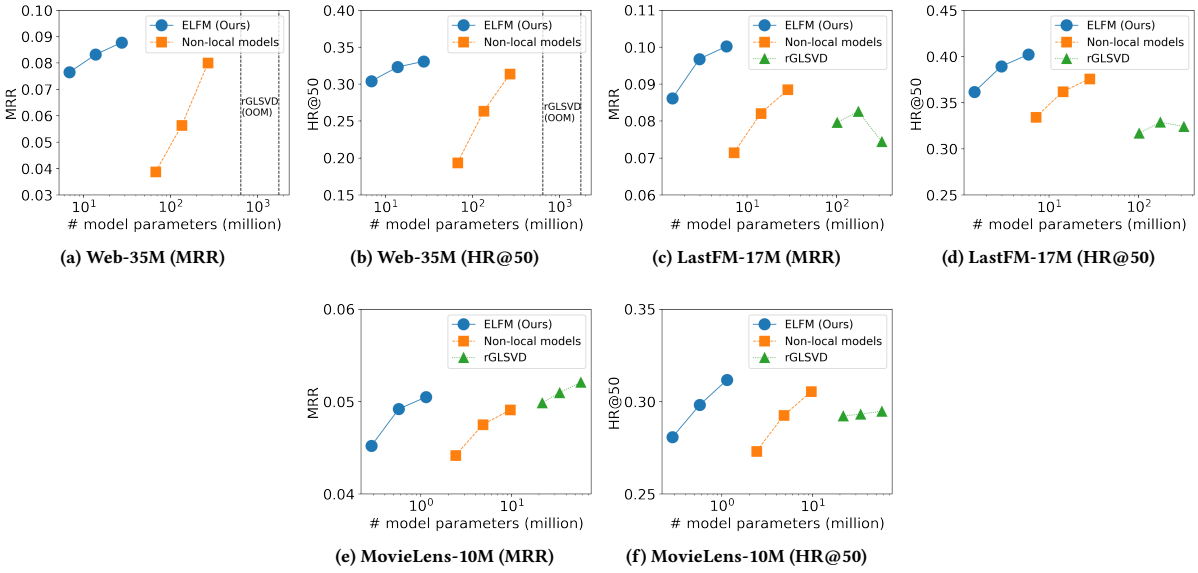
5.1 Top-K inductive recommendation

We present the results for the top-K inductive recommendation task in Table 3. Under both metrics, Elfm substantially outperforms all baselines on Web-35M and LastFM-17M with the gains becoming larger as the interaction signals become sparser (cf. Table 2). Moreover, Elfm achieves this boost while requiring only constant time per user during inference, whereas other baselines are much slower. On the MovieLens dataset, Elfm outperforms all baselines in terms of HR@50 and performs competitively with rGLSVD. The marginal gain on MovieLens dataset is in line with findings in recent work [16] showing that advanced methods such as neural architectures may not improve performance on dense datasets due to overfitting.

5.1.1 Tradeoff between performance and model size. We show the performance of Elfm and baseline methods under different model sizes in Figure. 3. For all methods, larger models generally result in better performance. However, prior local models (rGLSVD) are orders of magnitudes larger without significant performance gain. In fact, on LastFM-17M, local models perform even worse than non-local solutions, and they can not serve on Web-35M due to out of memory errors. In contrast, Elfm achieves significantly better

Model	Web-35M		LastFM-17M		MovieLens-10M	
	MRR	HR@50	MRR	HR@50	MRR	HR@50
Popularity	0.0162	0.1570	0.0312	0.1763	0.0247	0.1993
UserKNN	0.0510	0.1626	0.0844	0.3373	0.0483	0.2897
Bpr	0.0567	0.2479	0.0871	0.3758	0.0491	0.3054
WrMF	0.0654	0.2965	0.0885	0.3745	0.0489	0.2915
NeuMF	0.0800	0.3138	0.0714	0.3562	0.0420	0.2982
uCML	0.0195	0.1216	0.0760	0.3470	0.0466	0.2852
rGLSVD	OOM	OOM	0.0843	0.3357	**0.0521**	0.3020
Elfm (Ours)	**0.0878*****	**0.3308*****	**0.1002*****	**0.4020*****	0.0505	**0.3118**

Table 3: Top-K inductive recommendation performance. The best result of each column is in bold. The second best result is underlined. Elfm performed substantially better on Web-35M and LastFM-17M and competitively on MovieLens-10M. rGLSVD runs out of memory (OOM) on Web-35M dataset. Asterisks denote the statistical significance (paired t-test) of the best method comparing to the second best (***: $P < 0.001$).



(a) Web-35M (MRR) (b) Web-35M (HR@50) (c) LastFM-17M (MRR) (d) LastFM-17M (HR@50)

(e) MovieLens-10M (MRR) (f) MovieLens-10M (HR@50)

Figure 3: Top-K inductive recommendation performance of models with varied sizes. For latent factor-based models (Bpr, WrMF, NeuMF, and uCML), we report the best performance under each model size (orange lines) . On all three datasets, Elfm achieves significantly better or competitive performance with substantially fewer model parameters (OOM: Out Of Memory).

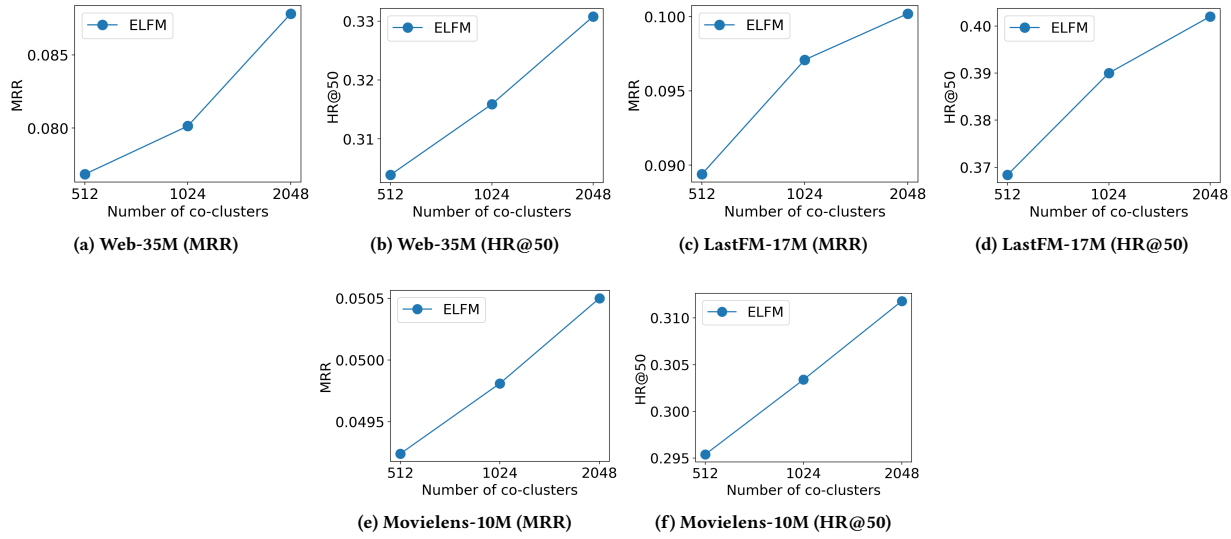
or similar performance with much fewer parameters — Elfm's size is only < 10% (MovieLens-10M), < 1% (LastFM-17M), and < 0.1% (Web-35M) of the size of rGLSVD.

5.1.2 The effect of the number of co-clusters. As shown in Figure. 4, we can boost Elfm's performance by increasing the number of co-clusters for the model to leverage. The number of clusters that Elfm can handle is at least 20 times more than that rGLSVD can address on million-scale datasets, and this allows Elfm to capture much more fine-grained local patterns for recommendations. However, we were not able to go beyond $N = 2048$ in our experiments with the described hardware and will pursue alternative implementations in future work.

5.2 User-item co-clustering

Elfm's outstanding recommendation performance demonstrates that fine-grained local patterns are highly useful for recommendation. We additionally investigate the effectiveness of local models by evaluating the coherence of the induced item subgroups through an intrusion task.

5.2.1 Item coherence. As the human evaluation results in Figure 5 show, Elfm and WrMF are the two top-performing methods with respect to precision, indicating that they produce the most coherent item subgroups among all methods. Last comes NeuMF where human performance was only marginally better than picking items at random, suggesting that NeuMF does not produce semantically



(a) Web-35M (MRR) (b) Web-35M (HR@50) (c) LastFM-17M (MRR) (d) LastFM-17M (HR@50)

(e) Movielens-10M (MRR) (f) Movielens-10M (HR@50)

Figure 4: Top-K inductive recommendation performance of ELFM with different number of co-clusters (N). ELFM improves with larger N.

Item Subgroup	BPR-Clustering	NEUMF-Clustering	WRMF-Clustering	ELFM
1	www.thesafarination.com	e-taxes.gov.az	www.personnel.alabama.gov	studentaid.ed.gov
	www.eliteautoraleigh.com	my.zunos.com	www.rsa-al.gov	studentloans.gov
	migration.kentucky.gov	insite.agoc.com	www.yourasecu.com	fafsa.ed.gov
	hairmax.com	c2healthcare.com	dhr.alabama.gov	myfedloan.org
	onthegoinmco.com	www.netminder.com	www.mymax.com	www.greatschools.org
2	www.eliteautoraleigh.com	e-taxes.gov.az	www.bwwb.org	usaa.versaic.com
	www.theenclavegainesville.com	www.insideepic.com	www.alabamaone.org	www.navyfederal.org
	www.thesafarination.com	help.miami.edu	www.alfainsurance.com	my.navyfederal.org
	playnet.fun	[removed]	www.avadiancu.com	www.usaa.com
	waterlandlife.org	my.icomtech.io	www.mymax.com	www.shopmyexchange.com
3	playnet.fun	e-taxes.gov.az	www.valdosta.edu	pearsonmylabandmastering.com
	www.thesafarination.com	laredo-cc.instructure.com	www.gmc.edu	www.ratemyprofessors.com
	www.eliteautoraleigh.com	elmis.co.tz	www.westga.edu	fafsa.ed.gov
	waterlandlife.org	[removed]	www.clayton.edu	www.citationmachine.net
	onthegoinmco.com	my.icomtech.io	www.westgatech.edu	www.chegg.com

Table 4: Item subgroups predicted for a randomly selected user. Each cell lists the top five hosts in that subgroup. ELFM's predictions are the most coherent within and distinct between subgroups. Some sites marked [removed] were not included in the publication as they were deemed as potentially questionable sites.

meaningful item representations. In between are the performances of BPR and BIGCLAM with only a small, statistically insignificant difference. Note that always randomly choosing an item would result in a precision of 0.25 (25%), as there were four items in total.

5.2.2 Quantity and quality of co-clusters. Even through WRMF-Clustering is good at discovering coherent item subgroups, it falls short in generating appropriate co-clusters, i.e., coupled subgroups of items and users. We define a *non-degenerate* co-cluster as one with at least 20 users and 20 items respectively. Considering the number of non-degenerate co-clusters shown in parentheses above each bar in Figure. 5, WRMF-clustering only yields 435 non-degenerate co-clusters – a third of what ELFM yields. The higher number of non-degenerate co-clusters from ELFM compared to WRMF can be seen as identifying many more fine-grained local co-clusters while maintaining similar precision. This more fine-grained co-clustering can be seen as a manifestation of local models' benefits.

In general, X-clustering-based approaches are limited because distances between user and item embeddings may be undefined for BPR, WRMF, and NEUMF, as is discussed in prior work [28]. Although not presented here, we found in pilot experiments where

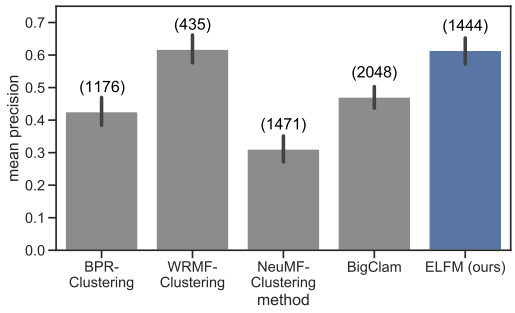


Figure 5: Average precision in intrusion task. Error bars indicate 95% confidence intervals. The number of non-degenerate co-clusters that a method produced is in parentheses. ELFM produces coherent item subgroups under a high number of non-degenerate co-clusters.

we tried using the clusters discovered by X-Clustering for recommendations that performance dropped substantially. Moreover, even for algorithms that directly learn distances between users and items such as UCML, its performance degrades dramatically for sparse interaction signals (shown in Table. 3), because the triangle inequality [28] poses a strong prior impeding the model to learn embeddings that have a multimodal distribution.

The inability of X-Clustering based methods to produce good co-clusters of users and items can also be seen in the qualitative example for a randomly sampled user in Table. 4. The rows correspond to the user's likeliest item subgroups and each cell lists the top 5 most associated items with that subgroup. This example highlights two things. First, the item subgroups from ELFM and WRMF are much more coherent than those from BPR and NEUMF which is in line with the human results. Moreover, WRMF fails to create sets of diverse and distinct item subgroups as the first two subgroups have strong topical overlap. In contrast, ELFM is able to capture local aspects of user's preferences – namely interests in web pages related to student loans or colleges, banking, and college classes.

6 RELATED WORK

Our work is primarily built upon and contributes to local models for recommendations, and is additionally inspired by research in community detection and co-clustering.

6.1 Local recommendation models

Collaborative filtering has been the prevailing paradigm used in modern recommendation algorithms. The core idea is to mine correlations between users and the items they consume. Common approaches include nearest neighbor-based methods (e.g., user-based [8] or item-based [45]), matrix factorization (e.g., WRMF [29] and BPR [44]), and deep neural networks (e.g., CDL [47], NeuMF [26], PinSage [59], and NGCF [48]). However, these algorithms are designed to mainly capture common consumption patterns manifested in the entire population through a global model. Unsurprisingly, these patterns can be limited or ineffective when applied to local clusters of users or items. For example, prior work has suggested that underrepresented users [21, 58] and items in niche categories [6] are insufficiently modeled and served.

To address this limitation, prior work developed local recommendation models [9, 14, 15, 33, 34, 53] to learn and leverage fine-grained patterns within subsets of users and items. These local models usually follow a three-step recipe — first they cluster users or items (through anchor points [33, 34] or K-means alike [9, 14, 15, 53]), then they train a fully-fledged model on each cluster, and finally combine local and global models for prediction. For example, Lee et al. [33, 34] developed a local collaborative ranking model (LCR) that groups users and items according anchor points, learns a low-rank model around each neighborhood, and computes weighted combinations as recommendations. The RGLSVD model developed by Christakopoulou et al. [14, 15] uses a similar procedure except that it directly clusters users and allows ranks of local models to vary. Methods like this suffer from two key limitations that create scalability issues in larger datasets. First, model sizes quickly become intractable as the number of clusters that one wants to model increases, because a full model is needed for each cluster. Second, models often have to be greatly simplified, because the required iterative optimization is limited in its ability to effectively search for optimal parameters. Commonly adopted simplifications include limiting the number of clusters that a user or item can belong to (often set to one [14, 15]), precomputing and fixing cluster assignments [33, 34], etc. Recent work on disentangled models [40, 49] share these limitations as these explicitly instantiate user and item embeddings for every intent [49] or prototype [40], and can only model disjoint user or item clusters.

In this paper, we address these two limitations of local recommendation models by proposing an end-to-end paradigm that encodes local information as an inductive bias into the model. Via our approach, the additional memory overhead is significantly reduced and the model is able to scale to millions of users and items. Also, soft assignment mechanism we employ eliminates the need for any adhoc simplifications — users and items are dynamically assigned to co-clusters that can overlap with each other.

6.2 Community detection and co-clustering

Clustering users and items is related to community detection, a widely studied topic in the context of networks and graphs. Existing approaches include factorization [55, 56], deep learning [13, 39], label propagation [54], and spectral methods [36, 37]. Many existing algorithms are not directly applicable to discover user-item co-clusters because they can not handle overlapping communities or are not scalable to million-scale datasets. Most relevant to our problem setting is BIGCLAM [56] which we also compare against in this paper. Through our human evaluation, we find that ELFM finds item clusters that are significantly more coherent than those from BIGCLAM. This may be due to the fact that BIGCLAM does not sufficiently leverage the bipartite structure of user interaction data.

The user-item consumption matrix can also be viewed as a bipartite graph with edges between users and items indicating whether or not an item was consumed. This transformation allows co-clustering methods to be applied to the problem. Prior literature has developed a rich set of methods for diverse problem settings,

such as overlapping co-clustering [51], theoretically grounded co-clustering [4, 19], and co-clustering over noisy data [18]. Topic modeling methods [7, 17, 27] can also be applied for co-clustering by casting items to "words", users to "documents", and co-clusters to "topics". However, most of the existing co-clustering methods can only operate on medium-sized data (at most thousands of rows or columns), rendering them impractical for large-scale problems.

In contrast to ELFM, community detection and co-clustering methods are not designed with recommendation as a primary task and thus provide no native interface for it. Existing unsupervised co-clustering methods seek to optimize certain geometry-based objectives, whereas ours train co-clusters in a supervised manner. The co-clusters are considered good if they perform well for the recommendation task (i.e., optimize the sampled loss) and form sparse clusters (i.e., optimize the regularization term) that also make sense to users (tested in our evaluation). As we show in our evaluation, the above methods result in suboptimal recommendation performance as well as suboptimal user-item co-clusters.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed ELFM, an End-to-end Local Factor model that leverages local consumption patterns through inductive bias for recommendations at scale. Beyond the strong empirical results demonstrating ELFM's superior recommendation performance and the coherence of the induced item clusters, it is computationally and memory efficient, and thus addresses a critical pain point of existing local models. ELFM also opens up many exciting application areas – for example using user-item co-clusters for explanations, direct preference elicitation at the co-cluster level, or detection of information silos in co-clusters (information known broadly in a co-cluster but limited knowledge elsewhere). Since ELFM does not directly handle cold-start recommendations for new users future work could leverage richer user profile signals to handle such cases. Other interesting avenues are extending the framework to capture temporal effects to detect trending items or determine experts or influential users, and incorporating auxiliary information (e.g., text, images, and meta-data) to enrich item representations.

REFERENCES

[1] 2006. *Yahoo! Webscope dataset ydata-ymusic-rating-study-v1.* http://research.yahoo.com/Academic_Relations

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *USENIX Symposium on Operating Systems Design and Implementation*.

[3] Eytan Adar, Jaime Teevan, and Susan T Dumais. 2008. Large scale analysis of web revisitation patterns. In *SIGCHI conference on Human Factors in Computing Systems*.

[4] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S Modha. 2007. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research* (2007).

[5] James Bennett, Stan Lanning, et al. 2007. The netflix prize. In *Proceedings of KDD Cup and Workshop*. Citeseer.

[6] Alex Beutel, Ed H Chi, Zhiyuan Cheng, Hubert Pham, and John Anderson. 2017. Beyond globally optimal: Focused learning for improved recommendations. In *International Conference on World Wide Web*.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* (2003).

[8] John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Conference on Uncertainty in Artificial Intelligence*.

[9] Jiajun Bu, Xin Shen, Bin Xu, Chun Chen, Xiaofei He, and Deng Cai. 2016. Improving collaborative recommendation via user-item subgroups. *IEEE Transactions on Knowledge and Data Engineering* (2016).

[10] Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems*.

[11] O. Celma. 2010. *Music Recommendation and Discovery in the Long Tail.* Springer.

[12] Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Advances in Neural Information Processing Systems*.

[13] Zhengdao Chen, Lisha Li, and Joan Bruna. 2019. Supervised Community Detection with Line Graph Neural Networks. In *International Conference on Learning Representations*.

[14] Evangelia Christakopoulou and George Karypis. 2016. Local item-item models for top-n recommendation. In *ACM Conference on Recommender Systems*.

[15] Evangelia Christakopoulou and George Karypis. 2018. Local latent space models for top-n recommendation. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[16] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *ACM Conference on Recommender Systems*.

[17] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* (1990).

[18] Meghana Deodhar, Gunjan Gupta, Joydeep Ghosh, Hyuk Cho, and Inderjit Dhillon. 2009. A scalable framework for discovering coherent co-clusters in noisy data. In *International Conference on Machine Learning*.

[19] Inderjit S Dhillon, Subramanyam Mallela, and Dharmendra S Modha. 2003. Information-theoretic co-clustering. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

[20] Travis Ebesu, Bin Shen, and Yi Fang. 2018. Collaborative memory network for recommendation systems. In *ACM SIGIR Conference on Research & Development in Information Retrieval*.

[21] Michael D Ekstrand, Mucun Tian, Ion Madrazo Azpiazu, Jennifer D Ekstrand, Oghenemaro Anuyah, David McNeill, and Maria Soledad Pera. 2018. All the cool kids, how do they fit in?: Popularity and demographic biases in recommender evaluation and effectiveness. In *Conference on Fairness, Accountability and Transparency*.

[22] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*.

[23] Per Christian Hansen. 1987. The truncatedsvd as a method for regularization. *BIT Numerical Mathematics* (1987).

[24] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* (2015).

[25] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI Conference on Artificial Intelligence*.

[26] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *International Conference on World Wide Web*.

[27] Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *ACM SIGIR Conference on Research and Development in Information Retrieval*.

[28] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *International Conference on World Wide Web*.

[29] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining*.

[30] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*.

[31] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *IEEE International Conference on Data Mining*.

[32] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[33] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2014. Local collaborative ranking. In *International Conference on World Wide Web*.

[34] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *International Conference on Machine Learning*.

[35] Jacek Łęski. 2003. Towards a robust fuzzy clustering. *Fuzzy Sets and Systems* (2003).

[36] Xiang Li, Ben Kao, Zhaochun Ren, and Dawei Yin. 2019. Spectral clustering in heterogeneous information networks. In *AAAI Conference on Artificial Intelligence*.

[37] Yeqing Li, Feiping Nie, Heng Huang, and Junzhou Huang. 2015. Large-scale multi-view spectral clustering via bipartite graph. In *AAAI Conference on Artificial Intelligence*.

[38] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *International Conference on World Wide Web*.
[39] Dongsheng Luo, Jingchao Ni, Suhang Wang, Yuchen Bian, Xiong Yu, and Xiang Zhang. 2020. Deep Multi-Graph Clustering via Attentive Cross-Graph Association. In *ACM International Conference on Web Search and Data Mining*.
[40] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. 2019. Learning Disentangled Representations for Recommendation. In *Advances in Neural Information Processing Systems*.
[41] Benjamin M Marlin. 2004. Modeling user rating profiles for collaborative filtering. In *Advances in Neural Information Processing Systems*.
[42] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*.
[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* (2011).
[44] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Conference on Uncertainty in Artificial Intelligence*.
[45] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *International Conference on World Wide Web*.
[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.
[47] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
[48] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *ACM SIGIR Conference on Research and Development in Information Retrieval*.
[49] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled Graph Collaborative Filtering. In *ACM SIGIR Conference on Research and Development in Information Retrieval*.
[50] Hongyi Wen, Longqi Yang, Michael Sobolev, and Deborah Estrin. 2018. Exploring recommendations under user-controlled data filtering. In *ACM Conference on Recommender Systems*.
[51] Joyce Jiyoung Whang and Inderjit S. Dhillon. 2017. Non-Exhaustive, Overlapping Co-Clustering. In *ACM on Conference on Information and Knowledge Management*.
[52] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *ACM International Conference on Web Search and Data Mining*.
[53] Yao Wu, Xudong Liu, Min Xie, Martin Ester, and Qing Yang. 2016. Cccf: Improving collaborative filtering via scalable user-item co-clustering. In *ACM International Conference on Web Search and Data Mining*.
[54] Yaowei Yan, Yuchen Bian, Dongsheng Luo, Dongwon Lee, and Xiang Zhang. 2019. Constrained local graph clustering by colored random walk. In *The World Wide Web Conference*.
[55] Jaewon Yang and Jure Leskovec. 2012. Community-affiliation graph model for overlapping network community detection. In *IEEE International Conference on Data Mining*.
[56] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *ACM International Conference on Web Search and Data Mining*.
[57] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh, and Deborah Estrin. 2018. Openrec: A modular framework for extensible and adaptable recommendation algorithms. In *ACM International Conference on Web Search and Data Mining*.
[58] Sirui Yao and Bert Huang. 2017. Beyond parity: Fairness objectives for collaborative filtering. In *Advances in Neural Information Processing Systems*.
[59] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.