Personalized Collaborative Clustering

Yisong Yue
Disney Research
yisong.yue@disneyresearch.com

Chong Wang
Carnegie Mellon University
chongw@cs.cmu.edu

Khalid El-Arini
Facebook
kelarini@fb.com

Carlos Guestrin
University of Washington
guestrin@cs.washington.edu

ABSTRACT

We study the problem of learning personalized user models from rich user interactions. In particular, we focus on learning from clustering feedback (i.e., grouping recommended items into clusters), which enables users to express similarity or redundancy between different items. We propose and study a new machine learning problem for personalization, which we call *collaborative clustering*. Analogous to collaborative filtering, in collaborative clustering the goal is to leverage how existing users cluster or group items in order to predict similarity models for other users' clustering tasks. We propose a simple yet effective latent factor model to learn the variability of similarity functions across a user population. We empirically evaluate our approach using data collected from a clustering interface we developed for a goal-oriented data exploration (or sensemaking) task: asking users to explore and organize attractions in Paris. We evaluate using several realistic use cases, and show that our approach learns more effective user models than conventional clustering and metric learning approaches.

Categories and Subject Descriptors

I.5.3 [**Computing Methodologies**]: Pattern Recognition—*Clustering*; H.5.2 [**Information Storage and Retrieval**]: Information Interfaces and Presentation—*User Interfaces*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Design, Experimentation, Human Factors

Keywords

Personalization, Clustering, Tensor Factorization

1. INTRODUCTION

How would you browse and organize attractions for a potential trip to Paris? How would you organize research articles while conducting a literature review? Such tasks are known as goal-oriented data exploration tasks, and the study of such "sensemaking" tasks is an area of intense interest within the human-computer interaction (HCI) community [21] as well as a plentiful source of new machine learning challenges.

One natural way to facilitate these types of sensemaking tasks is by using rich interfaces that allow users to interact more meaningfully with the datasets of interest. In this paper, we focus on modeling and learning from clustering interactions, which have become an increasingly popular paradigm for developing effective rich user interfaces (e.g., [5, 9, 2]) and have also seen commercial adoption.[1] In this setting, a user can cluster or group related items together, and obtain additional (personalized) recommendations based on this existing clustering.

Two key technical issues arise when developing machine learning approaches that are tailored to learning from clustering interaction feedback. First, in order to learn a model over all items, we must often learn from multiple users simultaneously, since each user typically only provides feedback on a small subset of items. However, different users may have very different similarity preferences, which typically requires learning multiple clustering models in order to characterize a heterogeneous user population. Consider the example in Figure 1, which shows how two users might organize attractions in Paris. For such settings, conventional approaches that do not account for personal tastes cannot effectively learn similarity models that are personalized to different users.

The second issue is that, for many applications, it can be difficult to design content-based features that can effectively capture the similarity preferences of different users. Consider again the example in Figure 1. We find that "Luxembourg Gardens" and "River Seine" are grouped in the same cluster by one user, and into different clusters by the other user. It is unclear whether content-based features alone – even those derived from semantically rich data sources such as Wikipedia – can effectively capture such subtleties.

These two issues are both related to *personalization*. In conventional recommendation settings, one popular approach for characterizing the multi-user personalization problem is collaborative filtering [16, 15]. In collaborative filtering, users provide feedback on an absolute scale (e.g., like/dislike or star rating). Most approaches for collaborative filtering are motivated by the intuition that even though users have different preferences, many users share preferences with other users. This intuition is often formalized by employing a latent factor model that automatically learns a low-dimensional representation that reliably characterizes the space of user preferences.

In this paper, we propose and study a clustering analogue to collaborative filtering, which we call *collaborative clustering*. Our

[1] E.g., `http://www.pinterest.com`.

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2744-2/14/04.
http://dx.doi.org/10.1145/2566486.2567991.

(a) "City Stroller"

(b) "Art Enthusiast"

Figure 1: Showing clusterings from two hypothetical users. The top user enjoys a good stroll, and groups the "River Seine" with nice walking spots in the city. The bottom user is more interested in art and architecture, and groups the "River Seine" in a broader "Outdoors" category.

assumption is that, although different users have different similarity preferences, most users share similarity preferences with other users. To tackle this problem, we propose a latent factor model, which we call "Latent Collaborative Clustering," to automatically learn a low-dimensional feature representation to reliably characterize the space (or variability) of users' similarity preferences. One interesting aspect of our approach is that it inherits the benefits of both tensor factorization as well as metric learning approaches (see Section 6).

We evaluate our approach using usage data collected from a clustering interface we developed for the sensemaking task of exploring and organizing attractions in Paris with the intention of planning a trip there (see Section 5 for more details). We conduct our evaluation based on several realistic use cases, and show that our approach learns more effective user models than conventional clustering and metric learning approaches. An implementation of our method as well as our collected dataset is publicly available.[2]

2. COLLABORATIVE CLUSTERING

We now define the problem of collaborative clustering. Suppose we have M users and N items. We assume each user has generated a clustering on a *subset* of the N items.[3] This type of preference data can be naturally collected from rich interfaces that support clustering (e.g., [9]). Figure 1 shows example clusterings that can be collected from our user interface (see Section 5 and Figure 4 for more details on our user interface and data collection process).

Our entire training data $\mathcal{Y}$ can be written as:

$$\mathcal{Y} = \{\mathcal{Y}_m\}_{m=1}^{M}. \tag{1}$$

We define each user's feedback $\mathcal{Y}_m$ as:

$$\mathcal{Y}_m = \left\{Y_m^1, \ldots, Y_m^{C_m}\right\}, \tag{2}$$

where each Y_m^i denotes a set of items that user m has indicated as belonging to the same group or cluster, and C_m indicates the total number of groups for user m. We also use the notation $\bar{\mathcal{Y}}_m = \{Y_m^1 \cup \ldots \cup Y_m^{C_m}\}$ to denote all the items that user m has clustered.

Similar to previous work on clustering with side information (e.g., [29, 13]), we model clustering feedback in the form of pairwise must-link and cannot-link labels: for each user m and every pair of items $i, j \in \bar{\mathcal{Y}}_m$ clustered by user m, define $y_{mij} = 1$ if user m groups items i and j into the same cluster, and $y_{mij} = -1$ if user m clusters items i and j into different clusters.

For example, the user in Figure 1(a) placed "Luxembourg Gardens" and "River Seine" into different clusters, so we would assign

$$y_{\text{"CityStroller"},\text{"LuxembourgGardens"},\text{"RiverSeine"}} = -1.$$

On the other hand, the user in Figure 1(b) placed the two attractions into the same cluster, so we would assign

$$y_{\text{"ArtEnthusiast"},\text{"LuxembourgGardens"},\text{"RiverSeine"}} = +1.$$

Our goal is to learn a similarity model $F(m, i, j)$ such that:

$$\forall m, \forall i, j \in \bar{\mathcal{Y}}_m : \; F(m, i, j) \approx y_{mij}.$$

2.1 Predicting Cluster Memberships

Given a partial clustering of items $\mathcal{Y}_m$ for user m, we are interested in having a model that can generalize to new items not yet recommended (and thus not yet clustered) by user m. For simplicity, we focus here on predicting the cluster membership of a single unclustered item at a time.[4]

For any unclustered item i, we score its affinity to cluster c created by user m as:

$$F(m, i, Y_m^c) = \texttt{mean}\{F(m, i, j) : j \in Y_m^c\}, \tag{3}$$

and define $\bar{c}_{mi}$ as the most likely cluster:

$$\bar{c}_{mi} = \underset{c \in \{1,\ldots,C_m\}}{\text{argmax}} \; F(m, i, Y_m^c). \tag{4}$$

We define our cluster prediction function for item i given partial clustering $\mathcal{Y}_m$ as predicting the cluster with highest affinity:

$$\texttt{predict}(i|m, \mathcal{Y}_m) = \begin{cases} \bar{c}_{mi} & \text{if } F(m, i, Y_m^{\bar{c}_{mi}}) > 0 \\ \perp & \text{if } F(m, i, Y_m^{\bar{c}_{mi}}) \leq 0 \end{cases}, \tag{5}$$

where $\perp$ denotes "new cluster" or "none of the above."

The two cases in (5) represent two types of generalization tasks. The first case corresponds to a conventional setting where all clusters are assumed known during evaluation. In this case, we simply predict the cluster with highest affinity to the unclustered item.

The second case (predicting whether an item belongs to a nonexistent cluster), which is arguably more interesting, is motivated by the intuition that recommending non-redundant items (i.e., those not belonging to any existing cluster) maximizes the novel information presented to the user. One appealing property of the affinity formulation $F(m, i, j)$ is that it naturally accommodates this type of generalization – we simply predict $\perp$ whenever item i has negative affinity with all existing clusters.

A priori, it may seem difficult to learn to predict cluster memberships for nonexistent clusters. We will present an approach that explicitly models how clusters are different through the must-link and cannot-link pairs across all users, thus allowing the resulting model to generalize to nonexistent clusters (for any given user) in a straightforward way.

[2]See: `http://projects.yisongyue.com/collab_cluster`

[3]This is analogous to the assumption common in collaborative filtering that each user has rated a subset of the items.

[4]Predicting for multiple items simultaneously will lead to a more structured prediction problem analogous to semi-supervised (or transductive) clustering [4].

3. LATENT COLLABORATIVE CLUSTERING

We now present a latent factor model, which we call "Latent Collaborative Clustering" (LCC), for learning the similarity function $F(m, i, j)$. Our approach can be viewed as both a natural clustering analogue of matrix factorization for collaborative filtering (see Section 6.1), as well as a latent-variable extension of feature-based metric learning for clustering (see Section 3.2 and Section 6.2).

Rather than relying on content-based features, we represent each item i as a latent vector $x_i \in \Re^D$ in a D-dimensional space; our approach will learn each x_i from the training data $\mathcal{Y}$. We represent each user m using a symmetric and positive semi-definite transform (i.e., a metric) matrix $U_m \in \Re^{D \times D} \succeq 0$, which corresponds to how user m finds items to be interrelated.

Model. We instantiate the similarity function $F(m, i, j)$ as:

$$F(m, i, j) = x_i^\top U_m x_j + b, \tag{6}$$

where $b \in \Re$ is a global offset term. This similarity model can be viewed as a natural merging of latent factor and metric learning approaches, and bears affinity to the pairwise interaction models used in [27, 13].

Learning Objective. We estimate our model using training data as described in (1) and (2). We formulate our training objective as finding item representations $\mathbf{x} = \{x_i\}_{i=1}^N$, user transforms $\mathbf{U} = \{U_m\}_{m=1}^M$, along with a bias term b to minimize the squared reconstruction error of the training data (subject to regularization),

$$\mathcal{L}(\mathbf{U}, \mathbf{x}, b) = R_x(\mathbf{x}) + R_u(\mathbf{U}) + \sum_m L_m, \tag{7}$$

where R_x and R_u denote the regularization penalty terms for item and user representations, respectively. Each L_m denotes the squared reconstruction error for user m,

$$L_m = \frac{1}{2} \sum_{i,j \in \bar{\mathcal{Y}}_m, i \neq j} \beta_{mij} (F(m, i, j) - y_{mij})^2, \tag{8}$$

where we use the weighting terms β_{mij} to control how to balance the relative importance of must-link and cannot-link training instances (see Appendix A.3 for more details on how we set β_{mij}). We also tried other loss functions such as the hinge loss [18]; the choice of loss functions did not greatly impact performance.

Regularization. When regularizing item representations $\mathbf{x}$, we use the standard L_2 norm:

$$R_x(\mathbf{x}) = \lambda_x \sum_i \|x_i\|^2, \tag{9}$$

here $\lambda_x > 0$ is a tunable hyperparameter. When regularizing user transforms $\mathbf{U}$, similar to [10] we regularize to the identity matrix:

$$R_u(\mathbf{U}) = \lambda_u \sum_m \|U_m - \sigma_u I_D\|_{Fro}^2, \tag{10}$$

where $\lambda_u > 0$ and $\sigma_u \in [0, 1]$ are tunable hyperparameters. Intuitively, (10) can be interpreted as an assumption that, *a priori*, each user places a small uniform weight on each latent dimension.

Optimization Problem. We also restrict each U_m to be diagonal,[5] which combined with $U_m \succeq 0$ implies that each $U_m \geq 0$. This leads to the following optimization problem during training:

$$\underset{\mathbf{U}, \mathbf{x}, b \,:\, \text{each } U_m \geq 0 \text{ and } U_m \text{ diagonal}}{\operatorname{argmin}} \quad \mathcal{L}(\mathbf{U}, \mathbf{x}, b). \tag{11}$$

[5]Note that learning a more general non-diagonal transform matrix often does not improve clustering performance in conventional feature-based metric learning approaches (cf. [31, 24]).

Note that, despite U_m being diagonal, (11) is optimizing over a rich class of three-way interaction models. By learning a common item representation $\mathbf{x}$, our LCC model can learn commonalities across users in order to generalize each user's similarity preferences to unclustered items (for that user). Also note that we do not place any constraints on the global offset term b.

Training. Note that (11) is convex with respect to each x_i or U_m, but not jointly in $\mathbf{x}$ and $\mathbf{U}$. We thus optimize (11) as a sequence of alternating constrained convex optimization problems. Appendix A describes in detail our optimization procedure. The form of our learning problem yields closed form solutions, which makes learning efficient. We also present a method to further speed up training in Appendix A.4.

3.1 Illustrative Example

As an illustration, we trained our LCC model over all the usage data $\mathcal{Y}$ we collected for our data exploration task of organizing attractions in Paris (see Section 5 for details on how we collected our usage data), resulting in item representations $\mathbf{x}$ for each attraction in our collection. Note that we did not use any content-based features in training our model, but only used the partial clusterings generated by our users. For each user depicted in Figure 1, we computed a transform U_m using that user's clustering and the previously learned $\mathbf{x}$. For each user, we then computed the most similar attractions to "River Seine" according to the resulting similarity function: $F(m, \text{"River Seine"}, j)$.

Figure 2 shows the eight most similar attractions to "River Seine" according to the two users' estimated similarity functions. In Figure 1, the first user groups "River Seine" under "City Strolling," whereas the second user groups it under "Outdoors." For the first user, the eight attractions most similar to "River Seine" (Figure 2(a)) are all conducive to city strolling. For the second user, the eight attractions most similar to "River Seine" (Figure 2(b)) all correspond to outdoor scenery. This example shows our model's ability to directly learn a latent representation of items that can capture the variability in similarity preferences across users. We refer to Section 4 for a quantitative evaluation.

3.2 Baselines & Extensions

Feature-Based Model. The most natural alternative to our approach is to use a feature-based model that utilizes an *observable feature representation* of items (e.g., features can be mined from a data source such as Wikipedia). In this case, our LCC learning problem reduces to conventional (multi-task) metric learning problems with side information (i.e., each user is a task) [29, 31, 24, 10, 32, 20]. We now describe a natural feature-based variant of our approach, which also serves as a baseline approach.

Let z_i denote a feature representation of item i, let $\mathbf{V} = \{V_m\}_{m=1}^M$ denote the user models, and let $b \in \Re$ again denote the global offset. We define the analogous affinity function:

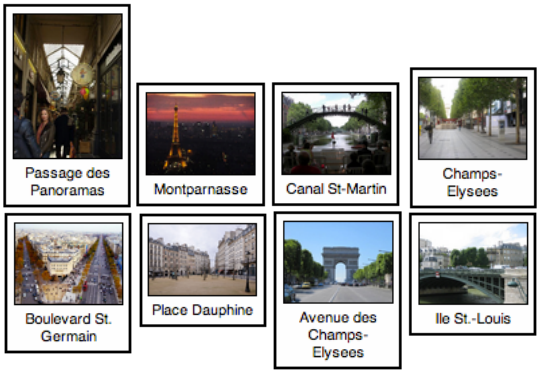
$$F(m, i, j) = z_i^\top V_m z_j + b. \tag{12}$$

Our modified learning objective can be written as:

$$\mathcal{L}(\mathbf{V}) = R_v(\mathbf{V}) + \tilde{R}_v(\mathbf{V}) + \sum_m L_m, \tag{13}$$

where L_m denotes the reconstruction error for user m (8), R_v denotes the per-user regularization term (analogous to (10)), and $\tilde{R}_v$ denotes the multi-task regularizer that enables sharing of information across each of the user tasks. We use the group regularization penalty [11, 20],

$$\tilde{R}_v(\mathbf{V}) = \tilde{\lambda}_v \sum_m \|V_m - \bar{V}\|_{Fro}^2,$$



(a) "City Stroller"

(b) "Art Enthusiast"

Figure 2: Showing the top eight most similar attractions to "River Seine" with respect to the two users in Figure 1 (see Section 3.1). For the top user (Figure 1(a)), the most similar attractions are related to "City Strolling". For the bottom user (Figure 1(b), the most similar attractions are related to "Outdoors". See Section 3.1 for more details.

where $\bar{V} = \frac{1}{M}\sum_m V_m$, although other multi-task regularization terms are possible (e.g., [32]).

Transformed Feature-Based Model. We can further extend our feature-based baseline (12) using a latent tranform $S^{D_z \times D}$ (which is similar to the approach in [6]),

$$F(m,i,j) = z_i^\top S V_m S^T z_j + b, \tag{14}$$

where D_z denotes here the dimensionality of the observed features z_i, and $D \leq D_z$ denotes the latent dimensionality. Essentially, $S^\top z$ transforms the observed features z into a "latent" space whereas our LCC model (6) directly learns the latent representation without using observed features.

One potential advantage of this model over the feature-based model is that it can fully model three-way interactions (between a user and a pair of items) using a latent representation (however, the model is limited to linear transforms of observable features). We will regularize S using the squared norm

$$R_s(S) = \lambda_s \|S\|^2.$$

Appendix A.6 gives a gradient descent formulation for training S.

Note that our LCC model is actually equivalent to having each item i assigned a unique feature, i.e., $z_i = e_i$ for e_i being the canonical unit vector along the i-th axis (so that $D_z = M$). In this case, we can establish an equivalence between LCC and the feature transform as $x_i = S_i$ where S_i denotes the i-th column of S. The special form of our LCC model yields significantly more efficient training algorithms due to having closed form solutions (see Appendix A). Furthermore, as we shall see in the experiments, it is unclear if leveraging observable features yields improved performance over directly learning a latent item representation in our problem setting.

Augmented LCC Model. The augmented LCC model is a natural extension that incorporates both latent and feature-based components. We can combine (6) and (12) (or (14)) to yield:

$$F(m,i,j) = x_i^\top U_m x_j + z_i^\top V_m z_j + b, \tag{15}$$

and define the modified learning objective as $\mathcal{L}(\mathbf{U},\mathbf{V},\mathbf{x},b) =$

$$R_x(\mathbf{x}) + R_u(\mathbf{U}) + R_v(\mathbf{V}) + \tilde{R}_v(\mathbf{V}) + \sum_m L_m. \tag{16}$$

4. EXPERIMENTS

We evaluated our approach using usage data collected from a clustering interface we developed for the data exploration task of organizing attractions in Paris. Each user was asked to organize in groups a small subset of 250 attractions in Paris (that they found interesting), with the intention of planning a trip there. We refer to Section 5 for more details regarding our user study. Overall, we collected data from 218 users, with an average of 18.7 items clustered and 4.5 clusters created per user. We evaluated both static as well as dynamic prediction tasks, which we describe in the following.

4.1 Static Prediction Experiments

4.1.1 Experiment Setup

We conducted 5-fold cross validation using the collected usage data. Each fold comprises approximately 125 training users, 50 validation users, and 43 test users. For each fold, we train all models using the training set with a range of hyperparameter settings (including regularization parameters and latent dimensions). We select the best model based on prediction performance on the validation set, and report performance on the test set.

For each user in the validation and test sets, at prediction time, we split the attractions clustered by the user into an input clustering and a held-out set. Each model is given the input clustering, and must predict to which of the existing clusters (or none) each held-out item belongs to. For our LCC model, the input clustering is used to learn the user-specific transform U. For the feature-based models, the input clustering is used to learn the user-specific V. Recall that predictions are made via `predict` (5).

We evaluate three types of **static prediction tasks**:

- **Hold 50%**. We randomly hold out 50% of the attractions the user clustered, and use the remaining clustered attractions as the input to the model.
- **Hold 25% per Cluster**. We randomly hold out 25% of each cluster the user created, and use the remaining clustered attractions as the input to the model. This setting is analogous to the first prediction task in Section 2.1, and is most analogous to conventional clustering tasks where all clusters are assumed to be known (i.e., none of the held out attractions belong to a nonexistent cluster).
- **Hold One Cluster**. We randomly hold out an entire cluster the user created, and use the remaining clustered attractions as the input to the model. The model must predict that none of the held out attractions belong to any existing clusters. This setting is analogous to the second prediction task in Section 2.1 (where the goal is to predict $\perp$), and is arguably the most interesting task since it directly relates to predicting which attractions have maximal novel information (to the user).

Table 1: Test set accuracy for various prediction tasks. Latent Collaborative Clustering (LCC) performs the best in all cases. Note that conventional feature-based approaches are ill-suited for the HOLD ONE CLUSTER task (see Section 4.1.3).

MODEL	HOLD 50%	HOLD 25% PER CLUSTER	HOLD ONE CLUSTER
Random	23.6%	19.6%	24.7%
Largest Cluster	30.1%	45.4%	0.0%
Feature 1	25.2%	26.1%	0.0%
Feature 2	35.1%	31.9%	0.1%
Transformed Feature 1	46.3%	56.1%	19.8%
Transformed Feature 2	31.1%	32.1%	0.7%
LCC	**55.7%**	**61.9%**	**36.7%**
Augmented LCC	53.8%	61.3%	34.1%

Table 2: Comparing test set accuracy of Latent Collaborative Clustering while performing model selection for different tasks (via hyperparameter selection in validation set). Note that the LCC model in Table 1 corresponds to optimizing for HOLD 50%.

OPTIMIZATION CRITERION	HOLD 50%	HOLD 25% PER CLUSTER	HOLD ONE CLUSTER
HOLD 50%	**55.7%**	61.9%	36.7%
HOLD 25% PER CLUSTER	55.6%	**62.1%**	37.7%
HOLD ONE CLUSTER	54.8%	61.1%	**42.9%**

4.1.2 Baselines

We generated two feature representations for training feature-based baseline models (see Section 3.2).

- **Feature 1**. The first feature representation was constructed using TF-IDF vectors [23] of the corresponding Wikipedia articles. This resulted in 3,403 features, with each feature corresponding to the TF-IDF value of a stem word.
- **Feature 2**. The second feature representation was constructed using a crowd-sourced tagging task on Amazon Mechanical Turk. This resulted in 39 binary features, which correspond to whether human labelers considered each attraction to be associated with the 39 tags (e.g., "Museum" or "Nature"). See Section 5.2 for more details.

In addition, we also compare against two simpler baselines:

- **Random**. Predict an existing cluster or nonexistent cluster uniformly and random.
- **Largest Cluster**. Always predict the largest existing cluster.

4.1.3 Static Prediction Results

Table 1 shows our main test results. For these results, we performed hyperparameter selection for all approaches based on HOLD 50% prediction accuracy in the validation set. We see that our Latent Collaborative Clustering (LCC) models uniformly outperform all baseline approaches. The Transformed Feature 1 Model was the most competitive baseline, although it still performed significantly worse than our LCC model, while being over 50 times slower to train.[6] We also note that the augmented LCC model (15) (trained using Feature 2) actually performs worse compared to the standard LCC model, which may be due to the augmented LCC model requiring significantly more training data is required to learn a superior model. These results suggest that directly learning a latent item representation can be an effective approach for capturing the variability of similarity preferences within a user population.

[6]The disparity in training time is due to our LCC modeling having efficiently computable closed form solutions (see Appendix A.4), whereas the Transformed Feature-Based model must resort to using gradient descent (see Appendix A.6).

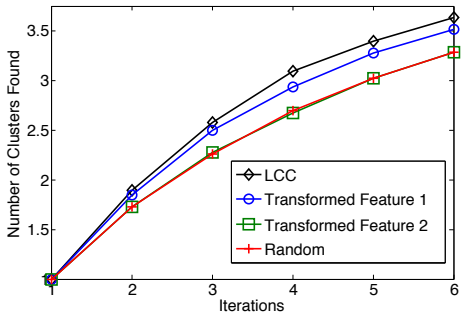


Figure 3: Showing the number of clusters created in the Sequential Prediction Experiment (see Section 4.2).

The feature-based models perform especially poorly on the HOLD ONE CLUSTER task (none of them outperformed the random baseline). We hypothesize three reasons for their poor performance. The first feature representation is very high dimensional, and thus likely requires significantly more training data to learn a good model (the Transformed Feature 1 model alleviates this issue somewhat by projecting item representations into a low-dimensional space). The second feature representation is likely not expressive enough to capture users' notions of dissimilarity, and as a consequence almost always predicts that held-out items should belong to *some* existing cluster (and hence rarely predicts $\perp$). Finally, training data for individual users is sparse, so it may be more beneficial to identify useful features of users rather than items (although such data was not collected in our user study).

Table 2 shows the test accuracy of our LCC model as we vary which prediction task we optimize for (via hyperparameter selection) in the validation set. We observe that the HOLD 50% and HOLD 25% PER CLUSTER tasks are reasonably aligned, whereas one can significantly boost performance for the HOLD ONE CLUSTER task at a slight expense to the other tasks.

4.2 Sequential Prediction Experiments

We also conducted a simulated **sequential prediction task**:

- **Maximize Number of Clusters**. For each test user, the goal is to iteratively recommend (unclustered) items in order to maximize the number of clusters created by that user (according to that user's collected training labels). Intuitively, this task measures a model's ability to recommend interesting items that the current user has not yet seen, and is related to the static prediction task HOLD ONE CLUSTER. .

Procedurally, for a given model and user m, the following happens:

- The model is initialized with an empty clustering.
- The model iteratively selects the unclustered item i with the lowest predicted affinity to any existing cluster, i.e., the item i with the lowest $F(m, i, \bar{c}_{mi})$ (4) (which can also be interpreted as predicting the item i with the highest likelihood that $\texttt{predict}(i|m, \mathcal{Y}_m) = \perp$ (5)). Ties are broken arbitrarily.
- The selected item is clustered according to the user's training labels (either added to an existing cluster or made into a new cluster).

Using the models selected in Section 4.1, we tested on users in the held-out test set using the same setup as in Section 4.1.1. We compared our LCC model against the following approaches:

- **Transformed Feature 1**. This model had the highest HOLD ONE CLUSTER performance.
- **Transformed Feature 2**. This model had the second highest HOLD ONE CLUSTER performance.
- **Random**. Randomly recommend an unclustered result.

Figure 3 shows the results. We observe that our LCC model is able to adaptively help the user create more clusters than the baselines. We further observe that the Transformed Feature 2 model actually does not outperform random guessing. Like in the HOLD ONE CLUSTER static prediction task, these results suggest that our LCC approach of directly learning a latent item representation can be an effective yet simple approach to making personalized recommendations to help the user discover novel (i.e., previously unseen) clusters.

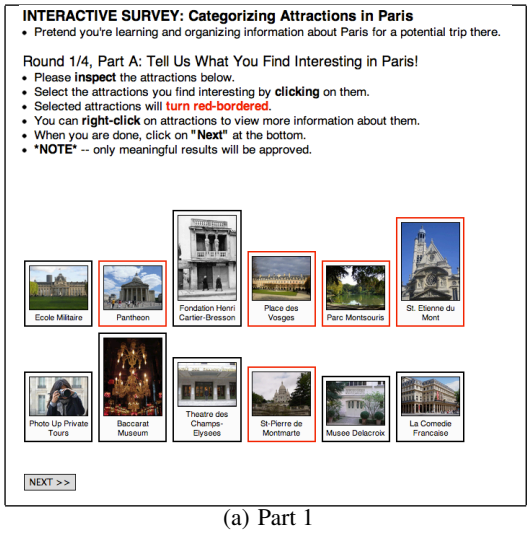
5. USER STUDY DETAILS

We collected 250 attractions in Paris from the TripAdvisor website.[7] We then constructed two datasets based on these attractions using human intelligence workers on Amazon Mechanical Turk.[8] The first dataset is the clustering data $\mathcal{Y}$ (1) used for training and evaluating our model. The second dataset is used to generate the second feature representation described in Section 4.1.2.

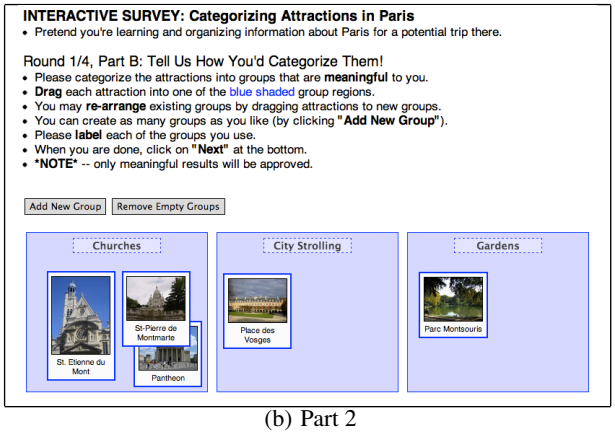
5.1 Cluster Feedback

To collect clustered feedback $\mathcal{Y}_m$ (2) from any given user, we ask that user to use a clustering interface that we developed. Figure 4(a) shows the first screen shown to the user. The user is asked to imagine that he or she is learning about and organizing attractions in Paris for a hypothetical trip there, and to use our clustering interface accordingly. The user study proceeds in four rounds, with each round comprising the following two parts:

- **Part 1**. Shown in Figure 4(a), the first part asks the user to select which of twelve randomly recommended attractions the user finds interesting in the context of planning a hypothetical trip to Paris.[9] More popular attractions (i.e., those ranked higher by TripAdvisor) have higher probability of being recommended.
- **Part 2**. Shown in Figure 4(b), the second part asks the user to organize the selected items from Part 1 into clusters. Clusters created in previous rounds are carried over.



(a) Part 1



(b) Part 2

Figure 4: Showing the two phases of clustering interface developed for the goal-oriented data browsing task of organizing attractions in Paris.

In total, 48 attractions are recommended to each user. Since our focus is on learning from clustering feedback, we only retain items that the users found interesting (and thus clustered).[10] On average, there are 18.7 clustered items and 4.5 clusters created per user.

Upon completion, we asked each user to self-report how well they understood the instructions and how useful they thought their generated clusterings would be for planning a hypothetical trip to

[7] http://www.tripadvisor.com
[8] https://www.mturk.com/

[9] Although we recommended results at random for data collection purposes, it is straightforward to modify our user study to evaluate adaptive recommendations algorithms (which were beyond the scope of this work) that reason about the entire interactive session and learn on-the-fly from user feedback.
[10] It would be interesting to develop models that jointly model both interest and similarity preferences. Note that modeling user interest is a conventional collaborative filtering problem.

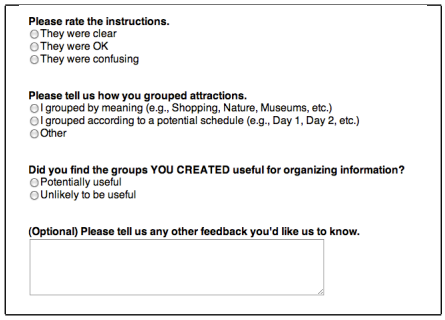


Figure 5: Showing the questionnaire given to users after they completed the clustering task.

Figure 6: Showing the tagging task for generating the second feature representation described in Section 4.1.2.

Paris. Figure 5 shows our closing questionnaire. Since our goal is to collect high-quality usage data from engaged users, we discarded any results if the user reported that the instructions were unclear or that the clusterings were useless. Overall, we retained approximately 80% of the user-generated clusterings for a total of 218.

5.2 Feature Tagging

We developed a tagging task to construct the second feature representation described in Section 4.1.2. Figure 6 shows our tagging interface. For each of the 250 attractions, we asked five human annotators to select which of 39 pre-specified tags (shown in Figure 6) should be associated with that attraction. Annotators were asked to select all tags that apply. We considered allowing users to specify their own tags, but that setup would dramatically increase the complexity of the data processing due to matching tags with similar meanings or spelling deviations.

We used this tagging data to construct a 39-dimensional binary feature representation of the 250 attractions (with each dimension corresponding to a tag). For each attraction, any tag that was selected by at least 3/5 annotators received a positive value in the corresponding binary feature, or otherwise a zero value.

6. RELATED WORK

Our work is motivated by recent advancements in the HCI community studying how to incorporate machine learning with rich user interactions. In particular, we focused on learning from clustering interactions [9, 2, 5]. In contrast to previous work, we aim to develop a systematic approach to model the variability of similarity functions contained within a user population.

The modeling approach most similar to LCC is Bayesian "crowd-clustering" [13]. One key difference is that [13] assumes there is a *global* (or consensus) set of atomic clusters (which different users may merge into varying higher-level clusters). As such, [13] focuses on recovering these atomic clusters from many higher-level partial clusterings. In contrast, we focus on more subjective user tasks, which are unlikely to yield agreed-upon atomic clusterings (e.g., organizing attractions in Paris based on personal interests).

Another related modeling approach is Bayesian clustered tensor factorization (BCTF) [27]. One key difference is that, for BCTF, pairwise relationships are not modeled symmetrically, which results in non-metric per-task transform matrices. In contrast, our collaborative clustering problem is naturally modeled using symmetric pairwise interactions that can be personalized to individual users using a metric transform.

The actual term "collaborative clustering" is not new, and has been used to refer to other clustering problems. For instance [14] studied the problem where the input data is distributed across many machines, and the machines must "collaborate" to arrive at a consensus clustering. Another example is [12], who studied how to combine ensembles of clusterings to make more robust predictions. In contrast, we use the term as an analogue to collaborative filtering. Another related work is [19], which uses latent representations to predict multiple non-redundant clusterings (for one task). In contrast, we focus on learning latent representations to capture the clustering variability of a user population.

6.1 Connection to Tensor Factorization

Our approach (6) can be viewed as a tensor factorization problem with missing values [1]. We can represent our training data $\mathcal{Y}$ (1) as a 3-tensor Y,

$$Y_{mij} = \begin{cases} y_{mij} & \text{if } (i,j) \in \bar{\mathcal{Y}}_m \\ ? & \text{otherwise} \end{cases}, \tag{17}$$

where ? denotes a missing value (i.e., user m did not cluster item i and/or item j).

Analogous to low-rank matrix (2-tensor) factorization approaches for collaborative filtering, our problem can be viewed as finding a low-rank 3-tensor factorization for collaborative clustering that has minimal reconstruction error on Y. In particular, our model can be viewed as a restricted form of the PARAFAC decomposition [1]:

$$Y_{mij} \approx \sum_{d=1}^{D} \gamma_d u_{md} x_{id} x_{jd} + b,$$

where each x_i and u_m are unit vectors, and γ_d are positive weights. Each x_i corresponds to an item representation, and each u_m corresponds to the diagonal of a user transform U_m. In our model, rather than constraining x_i and u_m to be unit vectors and controlling for magnitude via γ, we instead control the magnitudes of x_i and u_m (or U_m) via regularization penalties R_x and R_u.[11] We also enforce $u_m \geq 0$ to enforce each user model to be a metric transform.

6.2 Connection to Metric Learning

The problem of estimating user transforms U_m and V_m is related to (multi-task) metric learning problems under pairwise constraints

[11]The relationship between our latent factor model and the PARAFAC decomposition is analogous to that of bi-Gaussian latent factor models and the SVD in collaborative filtering [26, 22].

[29, 31, 24, 10, 32, 20]. Indeed, our feature-based baseline models (see Section 3.2) are instances of multi-task metric learning using a group regularization term [11, 20] (i.e., preferring all users to be similar to the "average" user $\bar{V} = (1/M)\sum_m V_m$).

Although one could, in principle, employ more sophisticated multi-task regularization penalties (e.g., [32]), we argue that the more important limiting factor for many applications is in building effective feature representations. Most multi-task metric learning approaches offer only modest improvements in performance (cf. [20, 32]). As we show in our empirical results, *directly learning a good latent representation* can lead to dramatic improvements in performance, while still being relatively efficient to train. We finally note that most multi-task metric learning approaches are typically designed for only a few tasks (typically less than 10), whereas our approach was applied to hundreds of tasks (i.e., each user is a task).

7. CONCLUSIONS & DISCUSSION

We have presented a new learning problem tailored to learning from clustering feedback called *collaborative clustering*, which can be viewed as a clustering analogue to collaborative filtering. We proposed a latent factor model to learn the space of clustering variability within a user population. We conducted empirical evaluations based on usage data collected from a clustering interface developed for a sensemaking task of exploring and organizing attractions in Paris. Our results show that our approach significantly outperforms conventional feature-based approaches on several realistic use cases.

In a sense, our approach for collaborative clustering is the simplest one that inherits the benefits of both tensor factorization as well as metric learning. It may be interesting to incorporate other advancements in collaborative filtering, such as localized latent embeddings, implicit feedback, and temporal dynamics [16, 15].

Another limitation of our approach is the inability to boost performance by using a joint model of both latent and observed features. Having a feature-based model is important for tackling the so-called cold-start problem for brand-new items with no feedback. This limitation may be due to the relatively simple linear content-based model we employed. For instance, recent work in collaborative filtering has demonstrated the ability to achieve the "best of both worlds" by effectively combining a latent factor model with a content-based (non-linear) topic model [30], and a similar approach may be fruitful for collaborative clustering. It may also be that user features are more useful than item features since training data per user is low.

Although we developed and validated our LCC approach for learning personalized clustering models, we did not formally model the full interactive setting where the system adaptively adjusts its recommendations based on user feedback. This full interactive setting can be considered as a type of interactive clustering problem. The most relevant related work on interactive clustering are [5, 9], although neither addressed how to model the end-to-end interaction sequence.[12] Other work such as [3] does provide guarantees for the interactive setting, but assumes a very different (and less realistic) interaction model where users must provide feedback on full clusterings of all items. Other adaptive clustering work include learning a single similarity function via crowdsourcing [28].

The broader goal is to design personalization frameworks that learn from multiple types of rich interactions (e.g., dynamic rankings [8] and zoomable metro maps [25]), as well as reason about (and optimize for) long-term user utility over entire interactive sessions. Progress towards this goal will require a confluence of progress in interface design, interpreting implicit feedback, structured prediction, and reinforcement learning.

[12]E.g., learning personalized models, deciding whether to make exploratory recommendations [17], and general reasoning about long-term user utility over the entire interaction sequence.

Algorithm 1 Solving the learning problem (11) for LCC

input: $S = \{\mathcal{Y}_m\}_{m=1}^M, \lambda_x, \lambda_u, \sigma_u, D$
$N \leftarrow$ number of items
$\forall m: U_m \leftarrow I_D, \vec{u}_m \equiv \texttt{diag}(U_m)$
$\forall i: x_i \leftarrow \texttt{random vector} \in \Re^D$
while not converged **do**
  $b \leftarrow$ (33)
  for $i = 1, \ldots, N$ **do**
    $x_i \leftarrow$ (18)
  end for
  for $m = 1, \ldots, M$ **do**
    $\rho \leftarrow const$ *//e.g.,* $\rho = 0.01$
    $\vec{\theta} \leftarrow \vec{0}$
    $\vec{\alpha} \leftarrow \vec{0}$
    while $\vec{u}_m \leftarrow$ (23) has negative components **do**
      $\vec{\theta} \leftarrow \max(\vec{u}_m + \vec{\alpha}, 0)$
      $\vec{\alpha} \leftarrow \vec{\alpha} + \vec{u}_m - \vec{\theta}$
      $\rho \leftarrow \gamma\rho$ *//e.g.,* $\gamma = 1.1$
    end while
  end for
end while
return: $(\mathbf{x}, \mathbf{U}, b)$

Acknowledgements. This research was supported in part by ONR (PECASE) N000141010672 and ONR Young Investigator Program N00014-08-1-0752. The authors also thank Niki Kittur, Dafna Shahaf and Jing Xiang for valuable discussions and feedback.

A. LEARNING PROCEDURE

We now show how to solve (11) for item represention **x** and user transforms **U** by solving a sequence of alternating constrained convex optimization problems. Note that solving for the user transforms on observed features **V** can be solved analogously to **U**.

Each item representation x_i can be solved in closed form when all other components are fixed. Since each user model U_m must lie in the positive orthant $U_m \geq 0$, we solve for each U_m via a sequence of closed form solutions (that iteratively converge to $U_m \geq 0$). Algorithm 1 shows our learning procedure. We also show how to speed up computation under special cases in Appendix A.4.

A.1 Estimating Item Representations

We estimate each x_i using closed-form coordinate descent. We can write $\partial\mathcal{L}/\partial x_i$ as ,

$$\lambda_x x_i + \sum_m \sum_{j \in \bar{\mathcal{Y}}_m, j \neq i} \beta_{mij} U_m x_j (x_j^\top U_m x_i + b - y_{mij}).$$

Setting the above to 0 gives the optimal solution for x_i,

$$x_i = \left(\lambda_x I_D + \frac{1}{2}\sum_m \Phi_{m,i}\right)^{-1} \left(\frac{1}{2}\sum_m \phi_{m,i}\right) \quad (18)$$

$$\Phi_{m,i} = U_m \left(\sum_{j \in \bar{\mathcal{Y}}_m, j \neq i} \beta_{mij} x_j x_j^\top\right) U_m \quad (19)$$

$$\phi_{m,i} = U_m \sum_{j \in \bar{\mathcal{Y}}_m, j \neq i} \beta_{mij}(y_{mij} - b) x_j. \quad (20)$$

A.2 Estimating User Transforms

For estimating a user's transform U_m, we first observe that:

$$x_i^\top U_m x_j = \vec{u}_m^\top (x_i \circ x_j),$$

Algorithm 2 Efficiently compute $\Phi_{m,i}$ (19) and $\phi_{m,i}$ (20).

input: item i
input: user m
$G_m \leftarrow \sum_{j\in\bar{\mathcal{Y}}_m} x_j x_j^\top$ *//cache this globally*
$g_m \leftarrow \sum_{j\in\bar{\mathcal{Y}}_m} x_j$ *//cache this globally*
$\Phi_{m,i} \leftarrow V_m \left((s-t_m)\sum_{j\sim_m i} x_j x_j^\top + t_m(G_m - x_i x_i^T)\right) V_m$ *//see* (27)
$\kappa_1 \leftarrow ((1-b)s + (1+b)t_m)\sum_{j\sim_m i} x_j$
$\kappa_2 \leftarrow (1+b)t_m(g_m - x_i)$
$\phi_{m,i} \leftarrow V_m(\kappa_1 - \kappa_2)$ *//see* (28)
return: $(\Phi_{m,i}, \phi_{m,i})$

Algorithm 3 Efficiently compute Ψ_m (24) and ψ_m (25).

input: user m
Define H_m according to (30) *//cache this globally*
Define h_m according to (32) *//cache this globally*
$\Psi_m \leftarrow (s-t_m)\left(\sum_{j\sim_m i}(x_i \circ x_j)(x_i \circ x_j)^\top\right) + t_m H_m$ *//see* (29)
$\psi_m \leftarrow ((1-b)s + (1+b)t_m)\sum_{j\sim_m i}(x_i \circ x_j) - (1+b)t_m h_m$ *//see* (31)
return: (Ψ_m, ψ_m)

where $\vec{u}_m \equiv \texttt{diag}(U_m)$ denotes the diagonal of U_m, and $\circ$ denotes the Hadamard product.[13] For each individual user (indexed by m), we can write the corresponding component in (11) as

$$\begin{aligned}&\bar{\mathcal{L}}(\vec{u}_m) \equiv \lambda_u \|\vec{u}_m - \sigma_u \vec{1}\|^2 \\ &+\tfrac{1}{2}\textstyle\sum_{i,j\in\bar{\mathcal{Y}}_m, i\neq j} \beta_{mij}\left(\vec{u}_m^\top(x_i \circ x_j) + b - y_{mij}\right)^2.\end{aligned} \tag{21}$$

Using the Alternating Direction Method of Mulitipliers optimization approach [7], we solve $\operatorname{argmin}_{\vec{u}_m \geq 0} \bar{\mathcal{L}}(\vec{u}_m)$ by iteratively solving a sequence of optimization problems of the form

$$\begin{aligned}&\bar{\mathcal{L}}^{(k)}(\vec{u}_m) = \lambda_u \|\vec{u}_m - \sigma_u \vec{1}\|^2 + \rho^{(k)}\|\vec{u}_m - \vec{\theta}^{(k)} + \vec{\alpha}^{(k)}\|^2 \\ &+\tfrac{1}{2}\textstyle\sum_{i,j\in\bar{\mathcal{Y}}_m, i\neq j} \beta_{mij}\left(\vec{u}_m^\top(x_i \circ x_j) + b - y_{mij}\right)^2,\end{aligned} \tag{22}$$

where $\rho^{(k)}$ is an increasing scalar sequence, $\vec{\theta}^{(k)}$ is an intermediate (always feasible) solution that converges to the constrained global optimum, and $\vec{\alpha}^{(k)}$ can be interpreted as a "dual" variable for the constraint $\vec{u}_m - \vec{\theta}^{(k)} = 0$. We can write $\partial\bar{\mathcal{L}}^{(k)}/\partial\vec{u}_m$ as

$$\begin{aligned}&\lambda_u(\vec{u}_m - \sigma_u\vec{1}) + \rho^{(k)}(\vec{u}_m - \vec{\theta}^{(k)} + \vec{\alpha}^{(k)}) \\ &+ \frac{1}{2}\sum_{i,j\in\bar{\mathcal{Y}}_m, i\neq j} \beta_{mij}(x_i \circ x_j)\left((x_i \circ x_j)^\top \vec{u}_m + b - y_{mij}\right)\end{aligned}$$

which yields an optimal solution for $\vec{u}_m$ as:

$$\vec{u}_m = A^{-1}c \tag{23}$$

$$A = \left(\lambda_u + \rho^{(k)}\right) I_{D^2} + \frac{1}{2}\Psi_m$$

$$c = \lambda_u\sigma_u\vec{1} + \frac{1}{2}\psi_m + \rho^{(k)}\left(\vec{\theta}^{(k)} - \vec{\alpha}^{(k)}\right)$$

$$\Psi_m = \sum_{i,j\in\bar{\mathcal{Y}}_m, i\neq j} \beta_{mij}(x_i \circ x_j)(x_i \circ x_j)^\top \tag{24}$$

$$\psi_m = \sum_{i,j\in\bar{\mathcal{Y}}_m, i\neq j} \beta_{mij}(y_{mij} - b)(x_i \circ x_j). \tag{25}$$

[13] In the case where U_m is not restricted to be diagonal, we can redefine $\vec{u}_m = \texttt{vec}(U_m)$ as the vectorized (or "flattened") $D^2 \times 1$ representation of U_m, and replace $\circ$ with the Kronecker product $\otimes$.

We then update $\rho^{(k)}$, $\vec{\theta}^{(k)}$, and $\vec{\alpha}^{(k)}$ according to Lines 15-17 in Algorithm 1 until convergence (which is guaranteed [7]).[14] Note that each computation of (23) is typically very fast since the latent dimensionality D is typically not very large.

A.3 Balancing Must-Links vs Non-Links

Following [30], we balanced links and non-links via:

$$\beta_{mij} = \begin{cases} s, & \text{if } y_{mij} = 1, \\ t_m, & \text{if } y_{mij} = -1, \end{cases} \tag{26}$$

where s and t_m are balancing parameters satisfying $s, t_m > 0$. From preliminary experiments, we set:

$$s = 1 \quad \text{and} \quad t_m = \frac{\text{number of links for user } m}{\text{number of total possible pairs for user } m}.$$

This structure allows efficient learning for special cases of our model (see Appendix A.4).

A.4 Efficient Learning

Efficiently estimating item representations. In order to efficiently compute (18), we must efficiently compute:

$$\Phi_{m,i} = \sum_{j\in\bar{\mathcal{Y}}_m, j\neq i} \beta_{mij} x_j x_j^\top$$

$$\phi_{m,i} = \sum_{j\in\bar{\mathcal{Y}}_m, j\neq i} \beta_{mij}(y_{mij} - b)x_j,$$

for each user m. We assume that each β_{mij} is set according to Section A.3. For each item i and user m, we can write $\Phi_{m,i}$ as:

$$(s-t_m)\sum_{j\sim_m i} x_j x_j^\top + t_m \sum_{j\in\bar{\mathcal{Y}}_m, j\neq i} x_j x_j^\top, \tag{27}$$

where $i \sim_m j$ indicates that item i and item j are grouped together by user m. Thus, we can cache $G_m \equiv \sum_{j\in\bar{\mathcal{Y}}_m} x_j x_j^\top$ to reduce the computational cost of (27) to be proportional to the actual number links seen in the clusters (which is typically much smaller than the total number of possible links). Similarly, we can write $\phi_{m,i}$ as:

$$((1-b)s + (1+b)t_m)\sum_{j\sim_m i} x_j - (1+b)t_m \sum_{j\in\bar{\mathcal{Y}}_m, j\neq i} x_j. \tag{28}$$

Again we can cache $g_m \equiv \sum_{j\in\bar{\mathcal{Y}}_m} x_j$ to reduce the computational complexity of (28) to be proportional to the actual number links seen in the clusters. Algorithm 2 describes procedurally how to leverage (27) and (28) to efficiently compute (19) and (20), respectively.

Efficiently estimating user transforms. In order to efficiently compute (23), we must efficiently compute:

$$\Psi_m = \sum_{i,j\in\bar{\mathcal{Y}}_m, i\neq j} \beta_{mij}(x_i \circ x_j)(x_i \circ x_j)^\top$$

$$\psi_m = \sum_{i,j\in\bar{\mathcal{Y}}_m, i\neq j} \beta_{mij}(y_{mij} - b)(x_i \circ x_j).$$

Similar to computing the item representations as described above, for each user m, we have:

$$\begin{aligned}\Psi_m =&(s-t_m)\sum_{j\sim_m i}(x_i \circ x_j)(x_i \circ x_j)^\top \\ &+ t_m \sum_{i,j\in\bar{\mathcal{Y}}_m, j\neq i}(x_i \circ x_j)(x_i \circ x_j)^\top.\end{aligned} \tag{29}$$

[14] In practice, we iterate until $\vec{u}_m \geq -\epsilon$ for some small $\epsilon > 0$, and then project $\vec{u}_m \leftarrow \max(\vec{u}_m, 0)$.

Define $X_m = [x_i]_{i \in \bar{\mathcal{Y}}_m}$. Note that:

$$H_m \equiv \sum_{i,j \in \bar{\mathcal{Y}}_m, j \neq i} (x_i \circ x_j)(x_i \circ x_j)^\top$$
$$= \left(X_m X_m^\top\right) \circ \left(X_m X_m^\top\right) - \sum_{j \in \bar{\mathcal{Y}}_m} (x_j x_j^\top) \circ (x_j x_j^\top), \quad (30)$$

which can be computed in time linear in number of items clustered by user m. Thus, (29) can be computed in time linear in the number of observed links of user m. We also have for ψ_m:

$$\psi_m = ((1-b)s + (1+b)t_m) \sum_{j \sim_m i} (x_i \circ x_j)$$
$$- (1+b)t_m \sum_{i,j \in \bar{\mathcal{Y}}_m, j \neq i} (x_i \circ x_j). \quad (31)$$

Similarly, we note that:

$$h_m \equiv \sum_{i,j \in \bar{\mathcal{Y}}_m, j \neq i} (x_i \circ x_j)$$
$$= \left(\sum_{j \in \bar{\mathcal{Y}}_m} x_j\right) \circ \left(\sum_{j \in \bar{\mathcal{Y}}_m} x_j\right) - \sum_{j \in Y_m} (x_j \circ x_j). \quad (32)$$

Algorithm 3 describes procedurally how to leverage (29) and (31) to efficiently compute (24) and (25), respectively.

A.5 Estimating Global Offset

Since the global offset b is not subject to regularization, we simply estimate b as the mean deviation of each pairwise (dis-)similarity scores in the training objective:

$$b = \texttt{mean}\left(\sum_m \sum_{i,j \in \bar{\mathcal{Y}}_m} \beta_{mij}(y_{mij} - x_i^\top U_m x_j)\right). \quad (33)$$

A.6 Estimating Feature Transform

We now describe how to estimate the feature transform S for the transformed feature-based model described in Section 3.2.

We can write the gradient $\partial\mathcal{L}/\partial S$ as

$$\lambda_s S + \frac{1}{2} \sum_m \sum_{i,j \in \mathcal{Y}_m, i \neq j} \beta_{mij}(F(m,i,j) - y_{mij})(z_i z_j^\top + z_j z_i^\top) S V_m,$$

which does not yield a close-form solution due to $F(m,i,j)$ having quadratic interactions of S (unless each item has disjoint features such that $z_i^\top z_j = 0$ when $i \neq j$, which is equivalent to our LCC model). As such, we perform gradient descent using $\partial\mathcal{L}/\partial S$.

B. REFERENCES

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations with missing data. In *SIAM Conference on Data Mining (SDM)*, 2010.

[2] S. Amershi, J. Fogarty, and D. Weld. Regroup: Interactive machine learning for on-demand group creation in social networks. In *ACM Conference on Human Factors in Computing Systems (CHI)*, 2012.

[3] M. Balcan and A. Blum. Clustering with interactive feedback. In *International Conference on Algorithmic Learning Theory (ALT)*, 2008.

[4] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.

[5] S. Basu, D. Fisher, S. Drucker, and H. Lu. Assisting users with clustering tasks by combining metric learning and classification. In *National Conference on Artificial Intelligence (AAAI)*, 2010.

[6] J. Blitzer and J. Weston. Latent structured ranking. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012.

[7] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternatiing direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[8] C. Brandt, T. Joachims, Y. Yue, and J. Bank. Dynamic ranked retrieval. In *ACM Conference on Web Search and Data Mining (WSDM)*, 2011.

[9] D. H. Chau, A. Kittur, J. I. Hong, and C. Faloutsos. Apolo: Making sense of large network data by combining rich user interaction and machine learning. In *ACM Conference on Human Factors in Computing Systems (CHI)*, 2011.

[10] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *International Conference on Machine Learning (ICML)*, 2007.

[11] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2004.

[12] G. Forestier, P. Gançarski, and C. Wemmert. Collaborative clustering with background knowledge. *Journal of Data & Knowledge Engineering*, 69(2):211–228, 2010.

[13] R. Gomes, P. Welinder, A. Krause, and P. Perona. Crowdclustering. In *Neural Information Processing Systems (NIPS)*, 2011.

[14] K. Hammouda and M. Kamel. Collaborative document clustering. In *SIAM Conference on Data Mining (SDM)*, 2006.

[15] Y. Koren and R. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

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

[17] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *World Wide Web Conference (WWW)*, 2010.

[18] N. Nello Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[19] D. Niu, J. Dy, and M. Jordan. Multiple non-redundant spectral clustering views. In *International Conference on Machine Learning (ICML)*, 2010.

[20] S. Parameswaran and K. Weinberger. Large margin multi-task metric learning. In *Neural Information Processing Systems (NIPS)*, 2010.

[21] D. M. Russell, M. J. Stefik, P. Pirolli, and S. K. Card. The cost structure of sensemaking. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, 1993.

[22] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Neural Information Processing Systems (NIPS)*, 2008.

[23] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[24] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Neural Information Processing Systems (NIPS)*, 2003.

[25] D. Shahaf, J. Yang, C. Suen, J. Jacobs, H. Wang, and J. Leskovec. Information cartography: Creating zoomable, large-scale maps of information. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

[26] N. Srebro. *Learning with Matrix Factorizations*. PhD thesis, Massachusetts Institute of Technology, 2004.

[27] I. Sutskever, R. Salakhutdinov, and J. Tenenbaum. Modelling relational data using Bayesian clustered tensor factorization. In *Neural Information Processing Systems (NIPS)*, 2009.

[28] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A. T. Kalai. Adaptively learning the crowd kernel. In *International Conference on Machine Learning (ICML)*, 2011.

[29] K. Wagstaff and C. Cardie. Clustering with instance-level constraints. In *National Conference on Artificial Intelligence (AAAI)*, 2000.

[30] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2011.

[31] E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Neural Information Processing Systems (NIPS)*, 2002.

[32] Y. Zhang and D. Yeung. Transfer metric learning by learning task relationships. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.