TFMAP: Optimizing MAP for Top-N Context-aware Recommendation

Yue Shi[a] *, Alexandros Karatzoglou[b], Linas Baltrunas[b], Martha Larson[a], Alan Hanjalic[a], Nuria Oliver[b]
[a]Delft University of Technology, Netherlands; [b]Telefonica Research, Spain
[a]{y.shi, m.a.larson, a.hanjalic}@tudelft.nl, [b]{alexk, linas, nuriao}@tid.es

ABSTRACT

In this paper, we tackle the problem of top-N context-aware recommendation for implicit feedback scenarios. We frame this challenge as a ranking problem in collaborative filtering (CF). Much of the past work on CF has not focused on common evaluation metrics that lead to good top-N recommendation lists in designing recommendation models. In addition, previous work on context-aware recommendation has mainly focused on explicit feedback data, *i.e.*, ratings. We propose TFMAP, a model that directly maximizes Mean Average Precision in aim of creating an optimally ranked list of items for individual users under a given context. TFMAP uses tensor factorization to model *implicit feedback* data (*e.g.,* purchases, clicks) with contextual information.

The optimization of MAP in a large data collection is computationally too complex to be tractable in practice. To address this computational bottleneck, we present a fast learning algorithm that exploits several intrinsic properties of average precision to improve the learning efficiency of TFMAP, and to ensure its scalability. We experimentally verify the effectiveness of the proposed fast learning algorithm, and demonstrate that TFMAP significantly outperforms state-of-the-art recommendation approaches.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

Keywords

Collaborative filtering, context-aware recommendation, mean average precision, implicit feedback, tensor factorization

1. INTRODUCTION

Collaborative Filtering (CF) methods are at the core of most recommendation engines. Most of the data traces left by online users come in the form of implicit feedback, i.e., we know which items a user interacted, e.g., purchased, used, or clicked, etc., and possibly also the count of each interaction, however, we do not have an explicit rating, i.e., a relevance score, that represents the strength of the user's interest in that item [13]. Learning the suggestion function from implicit feedback data, such as purchase or usage logs, can either be considered a classification problem, where items are classified to relevant or irrelevant, or a ranking problem where an optimal list of items is to be computed.

Top-N recommendation has recently attracted increased research interest because it generates a ranked list of results, which directly links to the end-user satisfaction [9]. Conventionally, recommender systems have been optimized to produced scores. A predicted score reflects the system's hypothesis of the strength of a particular user's preference for a particular item. In an overwhelmingly large number of recommender system use scenarios, users do not want preference strength information on all the items in the collection, but rather a compact list of top recommended items.

Although ranking-oriented CF approaches have been proposed for explicit feedback domains, e.g., EigenRank [19] and CoFiRank [34], those approaches are hardly applicable to implicit feedback domains, since they require training examples that are derived from the users ratings on various items. We notice that in the implicit feedback domains, the quality of a recommendation list that contains items of binary relevance can be quantified using Mean Average Precision (MAP), a well known evaluation measure in the information retrieval (IR) community. MAP provides a single-figure measure of quality across recall levels, has especially good discrimination and stability properties, and roughly corresponds to the average area under the precision-recall curve [21]. It is thus a good measure of performance when a short list of the most relevant items is shown to users [27]. A state-of-the-art approach, Bayesian Personalized Ranking (BPR) [24], has been recently proposed to train recommendation models by optimizing the measure of the Area Under the ROC Curve (AUC), which is based on pairwise comparisons between items. Note that in the AUC measure mistakes at the top of the list carry equal weight to mistakes in the bottom of the recommendation list. In contrast to AUC, MAP is a list-wise measure, for which mistakes in the recommended items at the top of the list carry a higher penalty than mistakes at the bottom of the list [10, 39]. Users typically consider only few (5 -10) top-ranked items in the recommendation list, it is thus particularly important to get the recommendations at the top of the list right. The top-heavy bias of MAP is thus particularly important in the recommendation problem. For this reason, we propose a recommendation model for implicit feedback data domains by directly optimizing the MAP measure.

*This work was conducted when the first author was an intern at Telefonica Research, Barcelona.

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

Typically, recommender systems have access to additional information about the user-item interactions, such as the *context* that is associated with the user-item interaction [1]. The context could be the location where the user listened to a song on his/her mobile phone or the time of the user-item interaction. Context-aware recommendations (CARs) are a new paradigm that can significantly improve the recommendation relevance and quality, compared to conventional recommendations solely based on user-item interactions [1, 4, 14, 25]. In this paper we present a generic CF model that is based on a generalization of matrix factorization to address context-aware recommendations. We extend the concept of matrix factorization to *tensor factorization*. A tensor is a generalization of the matrix concept to multiple dimensions. In the example above the user-item two-dimensional matrix is converted into a three-dimensional tensor of user-item-location interactions (see Figure 1).

Two key issues need to be considered in CARs: (1) *Context Integration*. The contextual information needs to be integrated in the recommendation model to be able to benefit the quality of the recommendation; and (2) *Optimization Function*. The recommendation model needs to be optimized under an objective function that corresponds to the recommendation quality for each user under each given context. Previous work in CARs has extensively studied the *context integration* issue, such as using tensor factorization [14] (TF) and factorization machines [25]. However, the second issue (*optimization function*) has only been addressed in a simplistic way. In the work of [14] and [25], the objective function in the recommendation model consists of minimizing the rating prediction error. This is an effective strategy where explicit feedback data is available from users, however, optimizing this objective is infeasible for scenarios with only implicit feedback data. In these scenarios, the quality of a recommendation list for a user is solely dependent on the positions of the relevant items in the list under the given context.

Here we propose a new context-aware recommendation approach based on tensor factorization for MAP maximization (TFMAP) that is designed to work with implicit feedback datasets. Taking insights from the area of *learning to rank*, TFMAP directly optimizes MAP for learning the model parameters, *i.e.*, latent factors/features of users, items and context types, which are then used to generate item recommendations for users under different types of context.

Directly optimizing MAP across all the users in a data collection is an expensive and non-trivial task. Therefore, we also propose a fast learning algorithm which exploits several properties of the average precision (AP) measure. We show that the computational complexity of the fast learning algorithm for TFMAP is linear to the number of observed items in a given data collection. Our contributions in this paper can be summarized as: 1) We propose a new generalized CF approach, TFMAP, that directly optimizes for MAP and leverages contextual information when available. We demonstrate that TFMAP outperforms state-of-the-art context-aware and context-free approaches. We observe significant improvements not only in MAP but also in precision at the top-N ranked recommendations. 2) To the best of our knowledge, TFMAP is also the first approach that can exploit datasets with implicit user feedback and contextual information. 3) We propose a fast learning algorithm that ensures the scalability of TFMAP and that exploits several properties of the AP measure.

The paper is organized as follows: in Section 2 we discuss the most relevant previous work and position our paper with respect to it. The research problem and the terminology used throughout the paper are presented in Section 3. In Section 4, we present the detail of TFMAP and the fast learning algorithm. Our experimental evaluation is reported in Section 5. Finally, Section 6 summarizes our main contributions and highlights a few areas of future work.

2. RELATED WORK

The work in this paper closely relates to three research areas: CF with implicit feedback, context-aware recommendation, and learning to rank. In the following, we present the most relevant related work in each of them.

CF with Implicit Feedback. Most CF approaches in the literature deal with the rating prediction problem, as defined in the Netflix prize competition[1]. A common approach to CF is to fit a latent factor model to the data, *e.g.*, latent semantic models [12, 28], and matrix factorization models, which learns a latent feature/factor vector for each user and item in the dataset such that the inner product of these features minimizes an explicit or implicit loss function [5]. Factor models have been shown to perform well in terms of predictive accuracy and scalability [2, 18, 26].

One of the first studies that used latent factor models for large implicit feedback datasets was introduced in [13]. It uses a least squares loss function and exploits the structure of the data (dominated by zero entries that correspond to negative preference), such that observed user-item interactions are weighted proportionate to the count of the interactions. Some extensions following this approach are introduced in [23] and [29]. In [24] a factorization approach based on the optimization of a smoothed pairwise ranking objective function was proposed. Optimizing the proposed objective function corresponds to maximizing the AUC. In this paper, we propose to learn a recommendation model by optimizing MAP, whose top-bias property is a significant advantage over AUC for recommender systems, as discussed in Section 1. In addition, our work is substantially different from the aforementioned work, since various types of contextual information are exploited for the recommendation.

Context-aware recommendation (CAR). Early work in CAR was done by utilizing contextual information for pre-processing, where context drives data selection, or post-processing, where context is used to filter recommendations [1, 4]. Recent work has focused on building models that integrate contextual information with the user-item relations and model the user, item and context interactions directly. Two state-of-the-art approaches have been proposed to date, one based on tensor factorization [14, 35] and the other on factorization machines [25] (FM). However, both approaches have been designed for the rating prediction problem, *i.e.* for explicit feedback.

In this paper we utilize a tensor factorization approach, *i.e.*, the CANDECOMP/PARAFAC (CP) model [15], to represent the interactions among the user, the item and the context type. Our approach includes two substantial innovations, compared to the state-of-the-art in CARs: (1) It targets recommendation scenarios with implicit feedback; and (2) it takes the evaluation metric (MAP) into account for learning the recommendation model.

Note that recommendation approaches have been proposed to take into account additional information (also referred as metadata, side information, or attributes) about users or

[1]http://www.netflixprize.com/

items, *e.g.*, collective matrix factorization [30], localized factor models [3] and graph-based approaches [16]. However, this type of information would go beyond our definition of "context", since we refer to context as information that is associated with both the user and the item at the same time. Finally, note that a recommended item set from a recommender is regarded as the "context" of user choice in the work of [38]. However, this type of context is still extracted from the user-item relations, thus, not in the scope of the context studied in this paper.

Learning to Rank. Learning to rank has been an attractive research topic in both the machine learning and the information retrieval communities [20]. Our work in this paper is closely related to recent research where proxies for common IR evaluation measures, such as NDCG and MAP, are used as the objective functions. The main difficulty of directly optimizing evaluation measures lies in their non-smoothness [7], *i.e.*, they are dependent on the rank values of ranked documents/items but not directly on the predicted relevance scores.

Ranking approaches can be broadly classified in two categories, those that implicitly optimize the IR measure and those that formulate an explicit approximation of the measure. LambdaRank [7] is a popular implicit optimization method which was proposed to apply gradient decent on an implicit loss function which is related to IR measures. Methods that explicitly optimize IR measures include structured estimation techniques [32] that minimize convex upper bounds of loss functions based on evaluation measures [37], *e.g.*, SVM-MAP [39] and AdaRank [36]. In the case of CF, CoFiRank [34] introduced a matrix factorization method where structured estimation was used to minimize over a convex upper bound of NDCG. SoftRank [31] was the first approach that proposed an explicit smoothed version of an evaluation measure, in which a rank distribution was employed, resulting in the expected values of document ranks that are smooth to the predicted relevance scores. In addition, a more general extension of SoftRank was presented by Chapelle et al. [8].

In this paper, we also employ an explicit approximation of MAP, which is a smooth function of model parameters. Our work is different from aforementioned research, since we target context-aware recommendation rather than query-document search, and we propose a fast learning algorithm, which is critical for large-scale recommender systems.

3. PROBLEM AND TERMINOLOGY

The research problem studied in this paper is stated as follows: *Given implicit feedback and contextual information on user-item interactions, recommend to each user and under a given context, an optimal (from a MAP perspective) item list.*

We denote the implicit feedback data from M users to N items under K types of context as a binary tensor Y, *i.e.*, a 3-dimensional tensor, with $M \times N \times K$ entries which are denoted with y_{mik}: (1) $y_{mik} = 1$ indicates that user m has interacted (*i.e.* purchased, used) with item i under context type k. We can thus assume that the user has a preference for this item; and (2) $y_{mik} = 0$ indicates the absence of an interaction and thus the preference of user m to item i under context type k is unknown. $|Y|$ denotes the number of nonzero entries in Y. Y_{mk} denotes a binary vector that indicates the user m's preference on all the items under context type k.

As mentioned in Section 2, the main idea behind factor

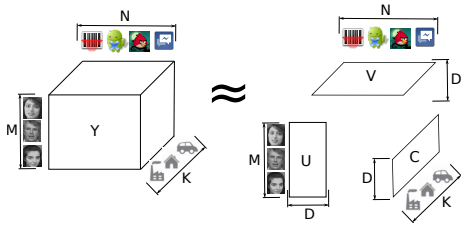

Figure 1: CP tensor factorization model.

models is to fit the original user-item interaction matrix with a low rank approximation. In this work we use tensor factorization (TF) as a generalization of the classical matrix factorization methods that accommodates for the contextual information. The latent features are stored in three matrices $U \in \mathcal{R}^{M \times D}$, $V \in \mathcal{R}^{N \times D}$ and $C \in \mathcal{R}^{K \times D}$ that correspond to users, items, and context types, respectively. We use U_m to denote a D-dimensional row vector, which represents the latent features for user m. Similarly, V_i represents the latent features of item i, and C_k represents the latent features of context type k.

We use the CP model [15], as illustrated in Fig. 1, for tensor factorization, in which user m's preference to item i under context type k is factorized as the inner product of the latent feature vectors, as shown below:

$$f_{mik} = \langle U_m, V_i, C_k \rangle = \sum_{d=1}^{D} U_{md} V_{id} C_{kd} \tag{1}$$

Based on user's m preference over all the items under context type k, we can then generate a recommendation list by ranking all the items in a descending order of the computed scores. Then, the AP of this list is defined as:

$$AP_{mk} = \frac{1}{\sum_{i=1}^{N} y_{mik}} \sum_{i=1}^{N} \frac{y_{mik}}{r_{mik}} \sum_{j=1}^{N} y_{mjk} \mathbb{I}(r_{mjk} \leq r_{mik}) \tag{2}$$

where r_{mik} denotes the rank of item i in the list of user m under context type k and $\mathbb{I}(\cdot)$ is an indicator function, which is equal to 1 if the condition is satisfied, and otherwise 0.

The MAP is defined as the average of AP across all the users and all the context types, as shown below:

$$MAP = \frac{1}{MK} \sum_{m=1}^{M} \sum_{k=1}^{K} \frac{\sum_{i=1}^{N} \frac{y_{mik}}{r_{mik}} \sum_{j=1}^{N} y_{mjk} \mathbb{I}(r_{mjk} \leq r_{mik})}{\sum_{i=1}^{N} y_{mik}} \tag{3}$$

4. TFMAP

In this section, we present the main technical contributions of this paper: (1) our proposed *smooth approximation of MAP*, its *optimization* and associated complexity analysis; and (2) a novel *fast learning algorithm* for optimizing over the smooth MAP measure in a context-aware setting.

4.1 Smoothed Mean Average Precision

It is apparent from Eq. (2) and (3), that AP (or MAP) depends on the rankings of the items in the recommendation lists. The rankings of the items change in a non-smooth way with respect to the predicted user preference scores and thus, the AP measure ends up being a non-smooth function with respect to the latent features of users, items and context types. We thus cannot use any of the standard optimization methods that require smoothness in the objective function.

As previously mentioned, significant progress has been made in the area of learning to rank regarding the explicit

optimization of evaluation metrics, such as MAP. The key issue is to approximate r_{mik} and $\mathbb{I}(r_{mjk} \leq r_{mik})$ in Eq. (2) and (3) by smoothed functions with respect to the model parameters, *i.e.*, U, V, and C.

Based on insights in [8], we approximate $\mathbb{I}(r_{mjk} \leq r_{mik})$ by the following logistic function:

$$\mathbb{I}(r_{mjk} \leq r_{mik}) \approx g(f_{mjk} - f_{mik}) = g(\langle U_m, V_j - V_i, C_k \rangle) \tag{4}$$

where $g(x) = 1/(1+e^{-x})$. The basic assumption is that the condition of item j being ranked higher than item i is more likely to be satisfied, if item j has relatively higher relevance score than item i. The authors in [8] also proposed a sophisticated approximation for r_{mik}, which, to the best of our knowledge, has not been deployed in practice. In the case of MAP, we argue it is not necessary to approximate r_{mik}, since only $1/r_{mik}$ is in use. For this reason, we propose to directly approximate $1/r_{mik}$ with another logistic function:

$$\frac{1}{r_{mik}} \approx g(f_{mik}) = g(\langle U_m, V_i, C_k \rangle) \tag{5}$$

Note that the larger the predicted relevance score f_{mik} the closer $g(f_{mik})$ gets to 1, resulting in a low value of r_{mik}. Reversely, the lower f_{mik}, the larger is r_{mik}. Substituting Eq. (4) and (5) into Eq. (3), we obtain a smoothed approximation of MAP:

$$\begin{aligned} MAP = & \frac{1}{MK} \sum_{m=1}^{M} \sum_{k=1}^{K} \frac{1}{\sum_{i=1}^{N} y_{mik}} \sum_{i=1}^{N} y_{mik} g(\langle U_m, V_i, C_k \rangle) \\ & \times \sum_{j=1}^{N} y_{mjk} g(\langle U_m, V_j - V_i, C_k \rangle) \end{aligned} \tag{6}$$

4.2 Optimization

Since Eq. (6) is smooth with respect to U_m, V_i, and C_k, we can now optimize it using standard methods, such as gradient ascent. In order to avoid overfitting , we add the Frobenius norms of the latent factors for regularization. Hence, the resulting TFMAP objective function is given by:

$$\begin{aligned} L(U,V,C) = & \sum_{m=1}^{M} \sum_{k=1}^{K} \frac{1}{\sum_{i=1}^{N} y_{mik}} \sum_{i=1}^{N} y_{mik} g(\langle U_m, V_i, C_k \rangle) \\ & \times \sum_{j=1}^{N} y_{mjk} g(\langle U_m, V_j - V_i, C_k \rangle) \\ & - \frac{\lambda}{2} (\|U\|^2 + \|V\|^2 + \|C\|^2) \end{aligned} \tag{7}$$

Note that we neglect the constant coefficient in MAP, since it has no influence on the optimization. Given a set of training data Y, a local maxima of Eq. (7) can be obtained by alternatively performing gradient ascent on one of the latent feature vectors at each step, while keeping the other latent vectors fixed. The gradients with respect to U, C, and V are given by Eq. (8∼10). Note that for notation convenience, we have performed the following substitutions:
$f_{mik} := \langle U_m, V_i, C_k \rangle$, $\quad f_{m(j-i)k} := \langle U_m, V_j - V_i, C_k \rangle$,
$\delta_A := g'(f_{mik}) \sum_{j=1}^{N} y_{mjk} g(f_{m(j-i)k}) - g(f_{mik}) \sum_{j=1}^{N} y_{mjk} g'(f_{m(j-i)k})$,
$\delta_B := g(f_{mik}) \sum_{j=1}^{N} y_{mjk} g'(f_{m(j-i)k})$.

$$\begin{aligned} \frac{\partial L}{\partial U_m} = & \sum_{k=1}^{K} \frac{1}{\sum_{i=1}^{N} y_{mik}} \sum_{i=1}^{N} y_{mik} \big[\delta_A (V_i \odot C_k) \\ & + \delta_B (V_j \odot C_k) \big] - \lambda U_m \end{aligned} \tag{8}$$

$$\begin{aligned} \frac{\partial L}{\partial C_k} = & \sum_{m=1}^{M} \frac{1}{\sum_{i=1}^{N} y_{mik}} \sum_{i=1}^{N} y_{mik} \big[\delta_A (U_m \odot V_i) \\ & + \delta_B (U_m \odot V_j) \big] - \lambda C_k \end{aligned} \tag{9}$$

$$\begin{aligned} \frac{\partial L}{\partial V_i} = & \sum_{m=1}^{M} \sum_{k=1}^{K} \frac{y_{mik} (U_m \odot C_k)}{\sum_{i=1}^{N} y_{mik}} \sum_{j=1}^{N} y_{mjk} \big[g'(f_{mik}) g(f_{m(j-i)k}) \\ & + \Big(g(f_{mjk}) - g(f_{mik}) \Big) g'(f_{m(j-i)k}) \big] - \lambda V_i \end{aligned} \tag{10}$$

where $\odot$ denotes element-wise product, and $g'(x)$ denotes the derivative of $g(x)$. Note that since either U_m or C_k is not coupled with other latent feature vectors as in Eq. (6), the derivation of Eq. (8) and (9) is straightforward. However, V_i is coupled with other latent feature vectors in Eq. (6), resulting in a more complicate derivation of Eq. (10). We leave the detailed derivation of Eq. (10) in the Appendix.

In order to understand the practical utility of TFMAP, we analyze the complexity of the learning process for one iteration. Given the data sparseness in the tensor Y and the fact that we usually have $|Y| >> M, K$, the computational complexity of calculating the gradients in Eq. (8) and (9) is $O(D|Y|)$, which is linear to the number of observed user-item interactions in the given tensor. Hence, the computation of the gradients with respect to the latent user features and latent context features is tractable, and able to scale up for large-scale use cases. However, the complexity of Eq. (10) is $O(DN|Y|)$. Considering that we usually have $|Y| >> N$, this complexity is even larger than quadratic to the number of items in the given collection. Thus, the computation of gradients regarding latent item features could be intractable in practice.

In the next section, we propose a novel *fast learning algorithm* to address the computational bottleneck in Eq. (10), reducing its complexity to $O(D|Y|)$.

4.3 Fast Learning

The proposed fast learning algorithm is outlined in Algorithm 1. Note that according to the definition of AP in Eq. (2), it is not necessary to optimize the latent features of all the items in order to maximize AP (as explained below).

The key idea of speeding up the learning process is to optimize, for each fixed pair of user m and context type k, the latent features of only a *set of representative* items, denoted as a buffer B_{mk}.

The gradient of the objective in Eq. (7) with respect to the latent features of item i in B_{mk} can be computed as:

$$\begin{aligned} \frac{\partial L}{\partial V_i} = & \sum_{m=1}^{M} \sum_{k=1}^{K} \frac{y_{mik} (U_m \odot C_k)}{\sum_{i \in B_{mk}} y_{mik}} \sum_{j \in B_{mk}} y_{mjk} \big[g'(f_{mik}) g(f_{m(j-i)k}) \\ & + \Big(g(f_{mjk}) - g(f_{mik}) \Big) g'(f_{m(j-i)k}) \big] - \lambda V_i \end{aligned} \tag{11}$$

The computational complexity then depends on the size of the buffer, *i.e.*, the number of items selected for each pair of user-context type. When all items are included in the buffer, Eq. (11) is equal to Eq. (10), while selecting fewer items in the buffer results in lower complexity.

The key issue with this approach is finding the *right* items to include in the buffer, as the quality of the learning process and hence the resulting model directly depends on the items included in the buffer. The buffer needs to be constructed in

such a way that it both reduces the computational complexity of the learning algorithm and conserves the necessary information to yield a high quality model.

4.3.1 Representative Item Selection

Relevant Items. For each user in a given context, we first include in the buffer all the items that have been observed by the user in that context, *i.e.*, for which we have the user's implicit feedback. These items are the basis for the computation of AP. Note that AP is defined based on the ranks of relevant items. Updating the latent features of relevant items should improve (*i.e.*, reduce) their rankings, thus, resulting in improved AP.

Irrelevant Items. Note that the ranking of irrelevant items influences AP indirectly, since their rankings are relative to the rankings of relevant items. Updating the latent features of irrelevant items will also improve (*i.e.*, raise) their rankings, thus, resulting in overall improved AP.

However, in practice, there are many more irrelevant items than relevant items for a user under a given context. The quantity of irrelevant items thus becomes the computational bottleneck in the learning algorithm of TFMAP.

For this reason, we choose to select only a relatively small number of irrelevant items in the buffer, n_{mk}, for user m and context type k. AP is a top-heavy list-wise ranking measure such that the lower the ranking of an item (the closer it is to the top of the list), the higher its influence in the final score. Top-ranked irrelevant items are the most influential items for AP optimization, yielding the following lemma:

LEMMA 1. *If we try to improve the AP of a ranking list by optimizing (*i.e.*, raising) the ranks of n irrelevant items, then raising the ranks of the top n irrelevant items should yield the largest improvement in AP.*

The proof in the case of $n = 1$ is provided in the Appendix. The proof for the case of $n > 1$ can be obtained in a similar way. Note that we could first sort all the irrelevant items for user m under context k in a descending order, according to the preference scores computed by the current model, *i.e.*, U_m, V and C_k in current iteration, and then select the top-ranked n_{mk} irrelevant items into the buffer.

In this work, we choose the set of irrelevant items in the buffer, n_{mk}, to be equal to the number of observed/relevant items for user m under context k, resulting in a total of $2n_{mk}$ items in the buffer.

We now optimize Eq. (7) for the latent features of the items within the buffer only. The complexity of Eq. (11) over each iteration is $O(2\tilde{n}^2MKD)$, where $\tilde{n}$ denotes the average number of observed items per user and context type. Note that we have $\tilde{n}MK = |Y|$ and $|Y| >> \tilde{n}$. Therefore, the complexity of Eq. (11) is $O(D|Y|)$, which is linear to the number of observed items in the given collection.

4.3.2 Efficient Buffer Construction

In order to select the top-ranked irrelevant items in each iteration, we need to make a prediction for each item and sort them according to the current predicted scores. Considering that most recommender systems contain large amounts of items, the computational cost for the prediction and sorting process would be very high. For this reason, we propose to sample a small set of irrelevant items and to select the top-ranked irrelevant items within the sampled set into the buffer.

We can maintain the representativeness of the top-ranked irrelevant items from the sampled set by using a key property of AP: *The items below the last relevant item in a ranked list have no contribution to AP.* This property can be easily understood from the definition of AP (see Equation 2).

Therefore, for each user under a given context type, we first find the relevant item with the lowest score. This operation is computationally cheap since the number of of relevant items is usually very small. We then sample n_s irrelevant items from those irrelevant items that have higher predicted relevance scores than the minimum predicted relevance score of the relevant items. This sampled set has higher probability to contain the globally top-ranked irrelevant items than a randomly sampled set. Note that the relevance scores are calculated by the model in each iteration. We illustrate the buffer construction for user m under context type k in an single iteration in Fig. 2.

In addition, since the model will become more accurate with each iteration, the minimum predicted score of the relevant items will also increase gradually. In other words, the position of the last relevant item in the ranked list will gradually move to the top of the list. As another by-product, this effect also helps to reduce the buffer construction time with each iteration. An experimental analysis confirming this property will be presented in Section 5.

ALGORITHM 1: Fast Learning TFMAP

Input: Training set Y, regularization parameter λ, sampling size n, learning rate γ, and the maximal number of iterations $itermax$.
Output: The learned latent features U, V, and C.
Initialize $U^{(0)}$, $V^{(0)}$, and $C^{(0)}$ with random values, and $t = 0$;
$p_0 = MAP$ based on Y and $U^{(0)}$, $V^{(0)}$, $C^{(0)}$;
repeat
  for $m = 1, 2, \ldots, M$ **do**
    $U_m^{(t+1)} = U_m^{(t)} + \gamma \frac{\partial L}{\partial U_m^{(t)}}$ based on Eq. (8);
  for $k = 1, 2, \ldots, K$ **do**
    $C_k^{(t+1)} = C_k^{(t)} + \gamma \frac{\partial L}{\partial C_k^{(t)}}$ based on Eq. (9);
  for $m = 1, 2, \ldots, M$ **do**
    for $k = 1, 2, \ldots, K$ **do**
      $B_{mk} = \textbf{BufferConstruct}(Y_{mk}, U_m^{(t)}, V, C_k^{(t)}, n)$;
      for $i \in B_{mk}$ **do**
        $V_i^{(t+1)} = V_i^{(t)} + \gamma \frac{\partial L}{\partial V_i^{(t)}}$ based on Eq. (11);
  $t = t + 1$;
  $p = MAP$ based on Y and $U^{(t)}$, $V^{(t)}$, $C^{(t)}$;
  if $p - p_0 \leq 0$ **then**
    break ;
  $p_0 = p$;
until $t \geq itermax$;
$U = U^{(t)}$, $V = V^{(t)}$, $C = C^{(t)}$

ALGORITHM 2: BufferConstruct

Input: User m's preference on all the items under context type k, i.e., Y_{mk}, and U_m, V, C_k, and sampling size n.
Output: B_{mk}.
$B_{mk} = \emptyset$;
$B_{mk} = B_{mk} \cup \{i | y_{mik} = 1\}$;
$n_{mk} = cardinality(B_{mk})$;
$p = \min_{i, y_{mik}=1} \langle U_m, V_i, C_k \rangle$;
$S = \{i | y_{mik} = 0\} \cap \{i | \langle U_m, V_i, C_k \rangle > p\}$;
Randomly sample n items from S as Q;
Descendingly sort items in Q, according to $\langle U_m, V_i, C_k \rangle, i \in Q$;
Set top-ranked n_{mk} items in Q as B^-;
$B_{mk} = B_{mk} \cup B^-$;

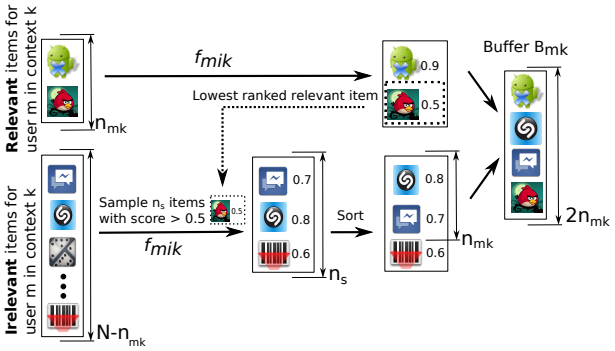

Figure 2: Illustration of buffer construction.

Note that the sampling size for the irrelevant items does not only influence the buffer construction time, but also the quality of the learned latent item features. We investigate this tradeoff between buffer size (i.e., computational cost) and performance in Section 5.

4.3.3 Termination Criterion

Since Eq. (6) is an approximation of MAP for the training data Y, we can use MAP (given by Eq. 3) as another termination criterion apart from conventional criteria, such as the number of iterations or the convergence rate. We stop the optimization process when we observe deteriorating values of MAP. Degrading values of MAP on the training data indicates that further optimizing the approximation of MAP as in Eq. (6) may not contribute to raising the true MAP.

5. EXPERIMENTAL EVALUATION

In this section we present a collection of experiments that evaluate the proposed TFMAP. We first give a detailed description of the datasets and setup that are used in the experiments. Then, we investigate the impact of several parameters in the proposed fast learning algorithm that are critical for TFMAP, as mentioned in Section 4.3. Finally, we evaluate the recommendation performance of TFMAP, compared to several baselines, and analyze its scalability.

The experiments were designed to address the following research questions: 1) Does the proposed fast learning algorithm benefit TFMAP in achieving MAP maximization? 2) Does TFMAP outperform state-of-the-art context-aware and context-free approaches? 3) Is TFMAP scalable for large-scale context-aware recommendation?

5.1 Experimental Setup

5.1.1 Dataset

The main dataset we use in this paper is from the *Appazaar* project[2] [6]. *Appazaar* recommends mobile applications to users from the Android Market. The application usage information from *Appazaar* users is recorded in the form of implicit feedback since *Appazaar* logs which apps are run by each user. In addition, *Appazaar* also tracks available contextual information from the phone sensors, such as motion sensor and GPS. We use two contextual factors in the experiments, i.e., motion (unavailable, slow, fast) and location (workplace, home, elsewhere). Note that both of the contextual factors were inferred from a GPS trace. Hence, the context variable has 9 possible types that take into account all the combinations of the two contextual factors, *i.e.*, $K = 9$ for C in Eq. (1). For example, context type "1" denotes that implicit feedback about a user running an application was observed when the user was at work and his/her motion status was unavailable. Finally, we represent one observation in the dataset as a triplet $(UserID, ItemID, ContextTypeID)$. The dataset contains 300469 triplets, 1767 users, 7701 items, 9 combinations of contextual features. On average, there are 18.9 app usage events per user and context type. A more detailed description of the dataset and its collection procedure can be found in [6].

Note that conventional CF benchmark datasets, *e.g.*, Netflix dataset, are not enriched with contextual information. Although the *Appazaar* dataset is not as large as these benchmark datasets, it is still much larger than datasets that have been used in previous context-aware recommendation work [14, 25]. Moreover, the datasets previously used in the CAR literature are all based on explicit ratings rather than implicit feedback, thus, not suitable for our study.

[2]http://appazaar.net/

5.1.2 Experimental Protocol

We separate the dataset into a training set and a test set, according to the timestamps. The training set consists of the first 80% implicit feedback data, while the test set contains the remaining 20% data. The target is to use the training set to learn a recommendation model, *i.e.*, U, V and C, which is then used to generate recommendation lists for each user under each type of context.

We use the MAP measure as in Eq. (3) to evaluate on the testset $\tilde{Y}$. Note that in order to have fair comparison with context-free approaches, we only preserve one context type for each user in the test set, *i.e.*, we randomly select one context type for each user in the test set and preserve the user's feedback within the selected context type, while excluding all the user's feedback data under other context types. To further clarify this design choice, we give a negative example in which a user in the test set has implicit feedback on the items under two different types of context. In this case, the MAP of context-aware approaches, such as TFMAP, should be measured according to AP under the two different types of context, while context-free approaches would only calculate AP based on the items and ignore the context. For this reason, our choice is necessary in order to attain fair comparative results to other context-free approaches.

In addition, note that since we only have implicit feedback from users, we cannot treat all the items that have no feedback in the test set as irrelevant/negative ones, in which case the recommendation performance could be severely underestimated. For this reason, we adopt a conventional widely-used evaluation strategy [9, 17], in which we randomly select 1000 items that have no feedback as irrelevant ones for each user in the test set. The performance is measured according to the recommendation list that only contains these 1000 items together with relevant items, *i.e.*, the items for which there is implicit feedback for that user.

In order to carry out our validation experiments, we randomly select 10% of all the implicit feedback data available in the training set. In our validation experiments we investigate the impact of the parameters and the fast learning algorithm in TFMAP.

Finally, note that we empirically tune the following conventional parameters so they yield the best performance in the validation test: regularization parameter λ=0.001, the latent dimensionality D=10, and the learning rate γ=0.001.

5.1.3 Setup for Comparison to FM

As mentioned in Section 2, the state-of-the-art context-aware approaches, such as FM [25], are designed to tackle the rating prediction problem (explicit feedback), and hence they are difficult, if not impossible, to apply to implicit feedback data. For this reason, we use another dataset, *Food* dataset [22], which was also used in the work of FM [25]. This dataset contains ca. 6K 5-scale ratings from 212 users on 20 menus/items, and each rating is associated with 2 contextual factors, *i.e.* one factor about whether the user's feeling about hunger is real or virtual (2 values: real, virtual) when she rated a menu, and the other factor about the user's hunger degree (3 values: normal, hungry and full). By taking into account all the combinations of the two contextual factors, we obtain 6 types of context in the Food dataset.

In our experiments, we randomly select 80% of the ratings as the training set and the remaining ratings as the test set. Items with a rating higher than 3 in the test set are considered to be relevant. Note that a different rating threshold could be set to define the relevant items. Under this setting, we use FM approach to first predict the ratings of the users on the items under each context type, and then generate the recommendation list according to the predicted ratings. For TFMAP, we train the model by converting the training set to an *implicit feedback* dataset, in which each rated item is regarded as an indicator of implicit feedback (i.e. the user tried the food item).

5.2 Validation: Impact of Fast Learning

We investigate the properties of the fast learning algorithm in TFMAP, presented in Section 4.3. The experimental results reported in this subsection are measured on the validation set previously described.

5.2.1 Impact of Sampling Size

By varying the sampling size in the fast learning algorithm of TFMAP, we investigate the buffer construction time and the performance variation in terms of MAP in the validation set, *i.e.*, an issue discussed in Section 4.3.2. We measure the buffer construction time cumulatively across all the users under all the context types in the training set over one iteration. The result is shown in Fig. 3.

Note that the buffer construction time increases almost linearly as the sampling size increases. Hence, with a relatively small sampling size, we could significantly reduce the buffer construction time compared to the case where all the irrelevant items for each user under a given context need to be ranked. For example, in the *Appazaar* dataset we have over 7000 items, which means that a sampling size of 200 could save over 50% of the buffer construction time, as illustrated in Fig. 3. Also note that the recommendation performance in terms of MAP increases sharply as the sampling size increases up to 200, and then saturates. Therefore, even with a relatively small size of irrelevant items, *e.g.,* 200, (compared to all the irrelevant items), the top-ranked irrelevant items within the sampled set are sufficiently representative to be used for MAP optimization.

In sum, these results empirically verify the selection of a small set of irrelevant items to create the buffer in the fast learning algorithm of TFMAP and justify our algorithm design choices. For the remaining experiments we will keep a sampling size of 200.

5.2.2 Impact of Representative Irrelevant Items

Here we aim to understand the the effectiveness of choosing the representative irrelevant items in the buffer. Rather than selecting representative irrelevant items to construct the buffer, an alternative is to use randomly selected irrelevant items. To test the random procedure we abandoned the ordering step of the algorithm and instead we randomly selected n_{mk} irrelevant items from the sampled set of size 200. In this case the accuracy yields a MAP of 0.083, dropping by 18.6% compared to the case where top-ranked irrelevant items are selected, i.e., MAP of 0.102 as shown in Fig. 3. Increasing the sampling size further emphasizes the benefit of carefully selecting the representative items. When we choose to sample 5000 irrelevant items, the benefit over the random strategy is 21.7%. This experiment validates the benefit of using representative irrelevant items in the buffer, as discussed in Section 4.3.1.

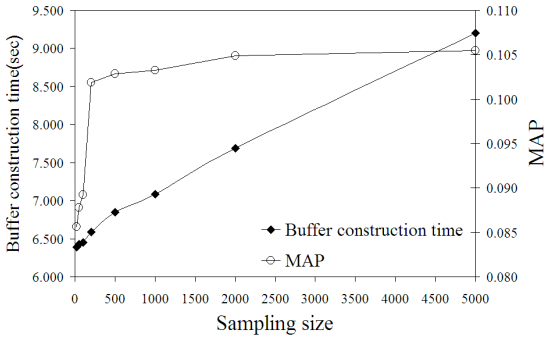

Figure 3: The impact of sampling size on buffer construction time and MAP of TFMAP.

5.2.3 Effect of the Lowest-ranked Relevant Item

As discussed in Section 4.3.2, it is not necessary to sample from all the irrelevant items in order to construct the buffer for a user in a given context, since the items ranked below the lowest-ranked relevant item have no influence on AP. Thus, the sampling process could be more efficient by neglecting the items ranked below the lowest-ranked relevant item.

Here, we present an experimental study that examines the change of the position of the lowest-ranked relevant item, *i.e.*, the maximal rank of relevant items in a recommendation list, during iterations, and also the change in the corresponding buffer construction time, as shown in Fig. 4. Note that this experiment is conducted on the validation set, with sampling size of 200 in TFMAP, and the results shown in Fig. 4 are the average values across all the users under all context types in each iteration.

We observe that the maximal rank of relevant items decreases with each iteration as the model is gradually optimized, *i.e.*, the model is more likely to rank relevant items higher in the list along iterations. This observation provides empirical evidence that exploiting the lowest-ranked relevant item in the sampling process does contribute to improving the quality of the representative irrelevant items, and also the efficiency of the buffer construction with each iteration. For example, the buffer construction time reduces by over 10% in the second iteration, compared to the first iteration.

5.2.4 Effectiveness of the Termination Criterion

Our final validation experiment investigates the effectiveness of the proposed termination criterion for the fast learning algorithm, as discussed in Section 4.3.3. We show the MAP measured in both the training (excluding the validation set) and the validation sets across the iterations, as in Fig. 5. Both MAP measures gradually improve towards an

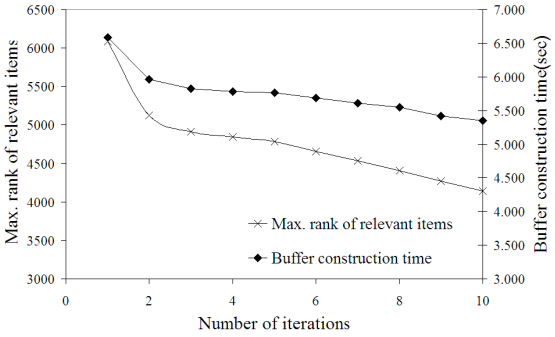

Figure 4: The average maximal rank of relevant items and the buffer construction time along iterations.

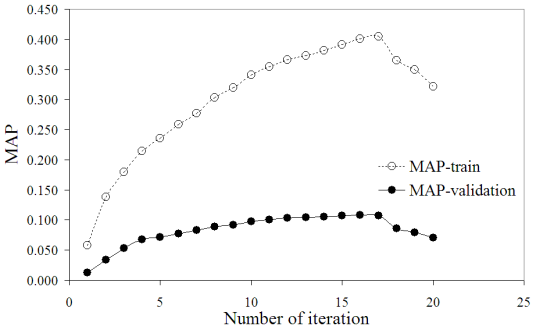

Figure 5: The MAP of the training set and the validation set in the learning process

optimal value with only a few iterations (less than 20), indicating that TFMAP effectively learns latent features for users, items and context types for MAP optimization. Also note that both MAP measures start dropping consistently after a few iterations, indicating that it is effective to use the MAP measured in the training set as a termination criterion for the learning process to avoid model overfitting.

From all the findings described in this section, we can give a positive answer to our first research question.

5.3 Performance Comparison

We now compare the performance of TFMAP with that of 5 baseline algorithms, according to the recommendation performance measured on the test set. The baseline approaches involved in this comparative experiment are listed below:

- **Pop**. A naive baseline that recommends items in terms of their popularity (*i.e.*, the number of observations from all the users) under the given context.
- **iMF**. A state-of-the-art CF approach proposed by Hu et al [13] for implicit feedback data.
- **BPR-MF**. Bayesian personalized ranking (BPR) represents another state-of-the-art optimization framework of CF for implicit feedback data [24]. BPR-MF represents the choice of using matrix factorization (MF) as the learning model with BPR optimization criterion. Note that the implementation of this baseline is done with the publicly available software MyMediaLite [11]. Although various learning models are available to be used with BPR, we find BPR-MF gives the best performance.
- **TFMAP-noC**. A variant of the proposed TFMAP, in which contextual information,*i.e.*, C, is not involved in the learning algorithm. Note that iMF, BPR-MF and TFMAP-noC are context-free methods, *i.e.*, the contextual information has no influence on the recommendations to individual users.
- **FM**. Factorization machine (FM) is a state-of-the-art context-aware approach [25]. As mentioned in Section 5.1.3, the comparison between FM and TFMAP is conducted on the *Food* dataset, due to the applicability of FM. Note that the implementation of FM is done with the publicly available software libFM[3].

Table 1: Performance comparison of TFMAP and context-free baselines on Appazaar dataset

	MAP	P@1	P@5	P@10
Pop	0.090	0.312	0.292	0.227
iMF	0.577	0.698	0.642	0.583
BPR-MF	0.612	0.800	0.712	0.602
TFMAP-noC	0.629	0.834	0.720	0.602
TFMAP	**0.659**	**0.879**	**0.732**	**0.611**

Table 2: Performance comparison of TFMAP and FM on Food dataset

	MAP	P@1	P@5	P@10
FM	0.152	0.036	0.050	0.055
TFMAP	**0.219**	**0.089**	**0.075**	**0.059**

Based on the Appazaar dataset, the recommendation performances of TFMAP and all the baselines except FM are shown in Table 1, from which we obtain three observations.

First, the context-free version of the proposed TFMAP, *i.e.*, TFMAP-noC significantly outperforms the other baselines in terms of MAP. Note that in our experiments, statistical significance is measured based on AP and precision values of all the users in the test set, according to Wilcoxon signed rank significance test with $p<0.01$. This result indicates that in the case that contextual information is unavailable, directly optimizing MAP as proposed in TFMAP could still lead to substantial improvement over state-of-the-art context-free approaches, such as iMF and BPR-MF.

Second, we can see that both BPR-MF and TFMAP-noC attain dramatic improvement in MAP over the other two baselines, Pop and iMF. As mentioned in Section 2, BPR is designed to optimize the evaluation metric AUC. The superior performance of BPR-MF and TFMAP-noC suggests that directly optimizing an evaluation metric that measures the recommendation performance in implicit feedback systems would yield significant improvements in the recommendation performance. In addition, note that TFMAP-noC achieves a significant improvement in MAP of 3% over BPR-MF, and 4% improvement of P@1. This result indicates that optimizing MAP is a better choice for recommender systems than optimizing AUC, since the top-heavy bias in MAP is a critical factor that provides substantial benefit for the recommendation performance.

Third, as can be seen, TFMAP achieves an additional significant improvement over TFMAP-noC, *e.g.*, 5% in MAP and P@1. This result indicates that TFMAP succeeds in utilizing contextual information together with user-item implicit feedback for maximizing MAP. In addition, the exploitation of context could greatly improve implicit feedback recommenders, a similar conclusion reached by previous work on CAR with explicit feedback [25, 14].

As mentioned before, we compare TFMAP with FM using the Food dataset, according to the protocol described in Section 5.1.3. The results are shown in Table 2. As can be observed, TFMAP significantly improves over FM to a large extent, i.e., by more than 40% in MAP, 100% in P@1 and

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

50% in P@5 and 8% in P@10, showing a great competitiveness for top-N context-aware recommendation.

From all the experimental results presented in this section, we confirm a positive answer to our second research question.

5.4 Scalability

The last experiment investigates the scalability of TFMAP by measuring the model training time against the amount of data used for training the model. We use from 10% to 100% of the training data (the observed implicit feedback data in the training set) for learning the latent features, and the corresponding training times are shown in Fig. 6. Note that we have normalized the training time by the time that is required for training the model with all the data in the training set. It can be observed that the training time increases almost linearly with the amount of the training data, empirically verifying the property of linear computational complexity. This finding also allows us to answer our last research question positively.

6. CONCLUSIONS AND FUTURE WORK

We have presented TFMAP, a novel top-N context-aware recommendation approach for implicit feedback domains. This approach utilizes tensor factorization to model each user's preference to each item under each type of context, and the factorization model is learned by directly optimizing MAP. We also propose a fast learning algorithm that exploits several properties of AP to keep the complexity of TFMAP linear to the number of implicit feedback data in a given collection, thus, making TFMAP scalable. Our experimental results verify the effectiveness of the proposed fast learning algorithm for TFMAP, and demonstrate that TFMAP could outperform several state-of-the-art context-aware and context-free recommendation approaches.

Taking insights from recent statistical analysis on evaluation measures [33], one line of our future work is to investigate the potential of optimizing other measures for context-aware recommendation, since different measures may represent different aspects of the recommendation quality. Another interesting topic of future work is to integrate contextual information together with metadata of users and items, as discussed in Section 2, to further advance the state-of-the-art in recommender systems.

7. ACKNOWLEDGMENTS

We are very grateful to Matthias Böhmer for making the Appazzar data available. This work is funded as part of a Marie Curie Intra European Fellowship for Career Development (IEF) awards held by Alexandros Karatzoglou (CARS, PIEF- GA-2010-273739).

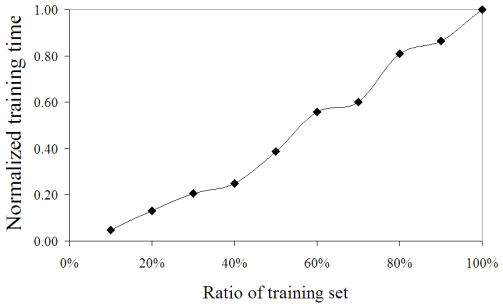

Figure 6: Scalability analysis of TFMAP

8. REFERENCES

[1] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23:103–145, January 2005.

[2] D. Agarwal and B.-C. Chen. Regression-based latent factor models. KDD '09, pages 19–28, 2009.

[3] D. Agarwal, B.-C. Chen, and B. Long. Localized factor models for multi-context recommendation. KDD '11, pages 609–617, 2011.

[4] L. Baltrunas and F. Ricci. Context-based splitting of item ratings in collaborative filtering. RecSys '09, pages 245–248, 2009.

[5] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proc. Intl. Conf. Machine Learning*, pages 65–72, 2004.

[6] M. Böhmer, B. Hecht, J. Schöning, A. Krüger, and G. Bauer. Falling asleep with angry birds, facebook and kindle - a large scale study on mobile application usage. In *Proc. of Mobile HCI '11*, 2011.

[7] C. J. C. Burges, R. Ragno, and Q. V. Le. Learning to Rank with Nonsmooth Cost Functions. In *NIPS*, pages 193–200, 2006.

[8] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *Inf. Retr.*, 13:216–235, June 2010.

[9] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. RecSys '10, pages 39–46, 2010.

[10] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. ICML '06, pages 233–240, 2006.

[11] Z. Gantner, S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Mymedialite: a free recommender system library. RecSys '11, pages 305–308, 2011.

[12] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004.

[13] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. ICDM '08, pages 263–272, 2008.

[14] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. RecSys '10, pages 79–86, 2010.

[15] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51:455–500, August 2009.

[16] I. Konstas, V. Stathopoulos, and J. M. Jose. On social networks and collaborative recommendation. SIGIR '09, pages 195–202, 2009.

[17] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. KDD '08, pages 426–434, 2008.

[18] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42:30–37, August 2009.

[19] N. N. Liu and Q. Yang. Eigenrank: a ranking-oriented approach to collaborative filtering. SIGIR '08, pages 83–90, 2008.

[20] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[21] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Cambridge Univ. Press, Cambridge [u.a.], 1. publ. edition, 2008.

[22] C. Ono, Y. Takishima, Y. Motomura, and H. Asoh. Context-aware preference model based on a study of difference between real and supposed situation data. UMAP '09, pages 102–113, 2009.

[23] R. Pan and M. Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. KDD '09, pages 667–676, 2009.

[24] S. Rendle, C. Freudenthaler, Z. Gantner, and S.-T. Lars. Bpr: Bayesian personalized ranking from implicit feedback. UAI '09, pages 452–461, 2009.

[25] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. SIGIR '11, pages 635–644, 2011.

[26] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. volume 20 of *NIPS '08*, 2008.

[27] G. Shani and A. Gunawardana. Evaluating recommendation systems. In F. Ricci, L. Rokach, and B. Shapira, editors, *Recommender Systems Handbook*, pages 257–298. 2010.

[28] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML*, pages 704–711, 2003.

[29] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. ICDM '10, pages 1055–1060, 2010.
[30] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. KDD '08, pages 650–658, 2008.
[31] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. WSDM '08, pages 77–86, 2008.
[32] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. Mach. Learn. Res.*, 6:1453–1484, 2005.
[33] J. Wang and J. Zhu. On statistical analysis and optimization of information retrieval effectiveness metrics. SIGIR '10, pages 226–233, 2010.
[34] M. Weimer, A. Karatzoglou, Q. Le, and A. Smola. Cofirank - maximum margin matrix factorization for collaborative ranking. NIPS '07, pages 1593–1600, 2007.
[35] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM'10*, pages 211–222, 2010.
[36] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. SIGIR '07, pages 391–398, 2007.
[37] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma. Directly optimizing evaluation measures in learning to rank. SIGIR '08, pages 107–114, 2008.
[38] S.-H. Yang, B. Long, A. J. Smola, H. Zha, and Z. Zheng. Collaborative competitive filtering: learning recommender using context of user choice. SIGIR '11, pages 295–304, 2011.
[39] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. SIGIR '07, pages 271–278, 2007.

APPENDIX

A. DERIVATION OF EQ. (10)

Note that in the following derivation, we leave out the derivative of the regularization term, i.e., $-\lambda V_i$, due to the space limit.

$$\frac{\partial L}{\partial V_i} = \sum_{m=1}^{M}\sum_{k=1}^{K} \frac{y_{mik}(U_m \odot C_k)}{\sum_{i=1}^{N} y_{mik}} \Big(g'(f_{mik}) \sum_{j=1}^{N} y_{mjk} g(f_{m(j-i)k}) - g(f_{mik}) \sum_{\substack{j=1 \\ j\neq i}}^{N} y_{mjk} g'(f_{m(j-i)k}) + \sum_{\substack{p=1 \\ p\neq i}}^{N} y_{mpk} g(f_{mpk}) g'(f_{m(i-p)k}) \Big)$$

Since we have $g'(-x) = g'(x)$, we obtain:

$$\frac{\partial L}{\partial V_i} = \sum_{m=1}^{M}\sum_{k=1}^{K} \frac{y_{mik}(U_m \odot C_k)}{\sum_{i=1}^{N} y_{mik}} \Big(g'(f_{mik}) \sum_{j=1}^{N} y_{mjk} g(f_{m(j-i)k}) + \sum_{\substack{p=1 \\ p\neq i}}^{N} y_{mpk} g(f_{mpk}) g'(f_{m(p-i)k}) - g(f_{mik}) \sum_{\substack{j=1 \\ j\neq i}}^{N} y_{mjk} g'(f_{m(j-i)k}) \Big)$$

By replacing index "p" by "j", we obtain:

$$\frac{\partial L}{\partial V_i} = \sum_{m=1}^{M}\sum_{k=1}^{K} \frac{y_{mik}(U_m \odot C_k)}{\sum_{i=1}^{N} y_{mik}} \Big(g'(f_{mik}) \sum_{j=1}^{N} y_{mjk} g(f_{m(j-i)k}) + \sum_{\substack{j=1 \\ j\neq i}}^{N} y_{mjk} \big(g(f_{mjk}) g'(f_{m(j-i)k}) - g(f_{mik}) g'(f_{m(j-i)k}) \big) \Big)$$

Since the term in the last summation is 0 when $j = i$, we obtain:

$$\frac{\partial L}{\partial V_i} = \sum_{m=1}^{M}\sum_{k=1}^{K} \frac{y_{mik}(U_m \odot C_k)}{\sum_{i=1}^{N} y_{mik}} \Big(g'(f_{mik}) \sum_{j=1}^{N} y_{mjk} g(f_{m(j-i)k}) + \sum_{j=1}^{N} y_{mjk} \big(g(f_{mjk}) g'(f_{m(j-i)k}) - g(f_{mik}) g'(f_{m(j-i)k}) \big) \Big)$$

$$= \sum_{m=1}^{M}\sum_{k=1}^{K} \frac{y_{mik}(U_m \odot C_k)}{\sum_{i=1}^{N} y_{mik}} \sum_{j=1}^{N} y_{mjk} \Big(g'(f_{mik}) g(f_{m(j-i)k}) + \big(g(f_{mjk}) - g(f_{mik}) \big) g'(f_{m(j-i)k}) \Big)$$

B. PROOF OF LEMMA 1

For the case of $n = 1$, considering that $r = [r_1, r_2, \ldots, r_N]$ denote the current ranks of all N items in a ranked list and r' denote their ranks after some optimization. We can optimize the latent features of a single irrelevant item a using a very small step size until its rank would increase from $r_a = q$ to $r'_a = q + 1$, consequently the rank of item b that was ranked $q + 1$ would now be ranked q, i.e., $r_b = q + 1$ and $r'_b = q$. Now the Lemma can be proved by proving that $\Delta AP = AP_{r'} - AP_r$ is a non-increasing function of q. It follows that the largest improvement ΔAP can be achieved by raising the rank of the irrelevant item with the lowest rank. Note that this proof is not related to a user or a context, so we simplify the notations. The ΔAP can be expressed as:

$$\Delta AP = \frac{1}{N_R} \sum_{i=1}^{N} \frac{y_i}{r'_i} \sum_{j=1}^{N} y_j \mathbb{I}(r'_j \leq r'_i) - \frac{1}{N_R} \sum_{i=1}^{N} \frac{y_i}{r_i} \sum_{j=1}^{N} y_j \mathbb{I}(r_j \leq r_i)$$

where N_R denotes the number of relevant items. Since we have the condition: $r_i = r'_i$, if $i \neq a, b$, we can obtain

$$\Delta AP = \frac{1}{N_R} \Big(\frac{y_a}{r'_a} \sum_{j=1}^{N} y_j \mathbb{I}(r'_j \leq r'_a) + \frac{y_b}{r'_b} \sum_{j=1}^{N} y_j \mathbb{I}(r'_j \leq r'_b) - \frac{y_a}{r_a} \sum_{j=1}^{N} y_j \mathbb{I}(r_j \leq r_a) - \frac{y_b}{r_b} \sum_{j=1}^{N} y_j \mathbb{I}(r_j \leq r_b) \Big)$$

Since we also have $y_a = 0$ as known, we further obtain:

$$\Delta AP = \frac{1}{N_R} \Big(\frac{y_b}{r'_b} \sum_{j=1}^{N} y_j \mathbb{I}(r'_j \leq r'_b) - \frac{y_b}{r_b} \sum_{j=1}^{N} y_j \mathbb{I}(r_j \leq r_b) \Big)$$

Substituting $r_b = q + 1$ and $r'_b = q$, we have:

$$\Delta AP = \frac{1}{N_R} y_b \sum_{j=1}^{N} y_j \Big(\frac{1}{q} \mathbb{I}(r'_j \leq q) - \frac{1}{q+1} \mathbb{I}(r_j \leq q + 1) \Big)$$

Again, when $j \neq a, b$, we have $r_j = r'_j$, and:

$$\mathbb{I}(r'_j \leq q) = \mathbb{I}(r_j \leq q + 1) = 1, \quad if \quad r'_j \leq q$$
$$\mathbb{I}(r'_j \leq q) = \mathbb{I}(r_j \leq q + 1) = 0, \quad if \quad r'_j > q + 1$$

Accordingly, we obtain:

$$\sum_{j=1, j\neq a,b}^{N} y_j \Big(\frac{1}{q} \mathbb{I}(r'_j \leq q) - \frac{1}{q+1} \mathbb{I}(r_j \leq q + 1) \Big) = N_q \Big(\frac{1}{q} - \frac{1}{q+1} \Big)$$

where N_q denotes the number of relevant items within top-q items. Finally, we obtain:

$$\begin{aligned} \Delta AP &= \frac{1}{N_R} y_b \Big(\sum_{j=1, j\neq a,b}^{N} y_j \Big(\frac{1}{q} \mathbb{I}(r'_j \leq q) - \frac{1}{q+1} \mathbb{I}(r_j \leq q + 1) \Big) \\ &\quad + y_a \Big(\frac{1}{q} \mathbb{I}(r'_a \leq q) - \frac{1}{q+1} \mathbb{I}(r_a \leq q + 1) \Big) \\ &\quad + y_b \Big(\frac{1}{q} \mathbb{I}(r'_b \leq q) - \frac{1}{q+1} \mathbb{I}(r_b \leq q + 1) \Big) \Big) \\ &= \frac{1}{N_R} y_b \Big(N_q \Big(\frac{1}{q} - \frac{1}{q+1} \Big) + y_b \Big(\frac{1}{q} - \frac{1}{q+1} \Big) \Big) \\ &= \frac{1}{N_R} \frac{N_q y_b + y_b^2}{q(q+1)} \end{aligned}$$

Note that $N_q \leq q$ and $y_b \in \{0, 1\}$. We complete the proof with ΔAP is a non-increasing function of q. A similar proof can be deduced for the case of $n > 1$.