Toward insightful and efficient sequential recommendation with metric models based on frequent sequences

Corentin Lonjarret[♯] Marc Plantevit [‡] Céline Robardet[*] Roch Auburtin[†]

Abstract

Modeling user preferences (long term history) and user dynamics (short term history) is of greatest importance to build efficient sequential recommender systems. The challenge lies in the successful combination of the whole users' history and their recent actions (sequential dynamics) to provide personalized recommendations. Existing methods capture the sequential dynamics of a user using fixed-order Markov chains regardless the user, which limits the personalization of the user dynamics. In this paper, we propose to use frequent sequences to identify the part of user history which is the most relevant to the recommendation. The most salient items are then used in a unified metric model that embeds items based on user preferences and histories. To this end, we take advantage of the latest techniques that have shown their performance in recent methods. Experiments reported on 13 datasets demonstrate that our method outperforms state-of-the-art methods. In addition, we show that sequences provide explanations when recommending an item, as well as information about users and datasets.

1 Introduction

The number of situations for which a user is facing a huge number of possibilities, such that it is impossible for her to make the most appropriate choices by evaluating each of the individual cases, increases at the same rate as the digitalization of the society. Offering the possibility to access to a virtually infinite set of digital contents, or digital descriptions of real objects, called hereafter items, is therefore only possible if, at the same time, the search process is equipped with powerful recommendation tools. Recommender systems have received much attention over the last two decades [1, 6, 16]. From the digital trace left by a user during her partial exploration of the set of items, these methods aim to select a small number of elements of interest for her.

In this paper, we tackle the problem of sequential recommendation that aims to predict/recommend for a user the next item from collaborative data. This is a challenging problem whose importance have been gradually recognized by researchers. Indeed, the user preferences (i.e., the long term dynamics) and the user sequential dynamics (i.e., the short term dynamics) need to be fruitfully combined to account for both personalization and sequential transitions. Early approaches sought to group users with similar traces through a matrix decomposition process that reveals latent factors as typical user profiles on which new users are positioned [4, 11, 12]. Another direction is to decompose a similarity matrix between items to better account for transitive relationships between items [10]. More recently, and especially since the traces left by the users become longer, a growing body of literature has investigated the integration of the user sequential dynamics [5, 9, 15] through fixed order Markov chain estimations. It supposes that the conditional probability of an item depends on a fixed number L of past items. However, the number of parameters to be estimated increases exponentially with L, making the fitting of high-order models intractable. Furthermore, Markov chains consider exactly the same number of items whatever the users and their actions. To cope with these problems, we propose a new model, **REBUS**, that uses frequent sequences to identify the part of user history that is the most relevant for recommendation. The most salient items are then used in a unified metric model that embeds items based on user preferences and histories. By doing so, we also take advantage of the latest techniques that have shown their performance in recent methods, such as the use of distances instead of inner product [5] to normalize the embedded vectors and make the method more robust.

Figure 1 illustrates how **REBUS** (a.k.a. Recommendation Embedding Based on freqUent Sequences) works. A representation of the user preferences is computed by embedding all items of the user history. Similarly, frequent sequences are used as a proxy to learn conditional distributions of item appearance in user sequences and thus capture sequential dynamics. To embed user preferences and dynamics in a single Euclidean space and make them comparable, we do not embed user sequences (whose number is much larger than the number of items) but the items making them, weighted by a temporal damping factor. The items that are the nearest of the vector resulting of the sum of user preference and sequential dynamics are chosen for recommendation.

[*]INSA Lyon, CNRS, LIRIS UMR5205, F-69621 France
[†]Visiativ, France
[‡]Université Lyon 1, CNRS, LIRIS UMR5205, F-69622 France

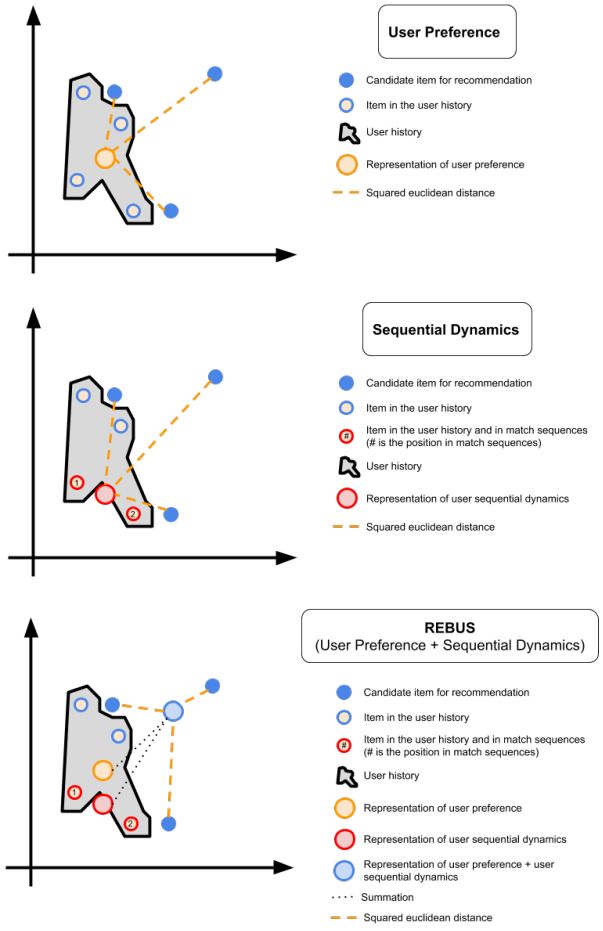

Figure 1: Overview of **REBUS** that accommodates user preferences and sequential dynamics through Euclidean distances. **User preferences** are represented by the embedding of items history. **Sequential Dynamics** is represented by the embedding of the context items, i.e. items that match a frequent sequential pattern. The order of the items within the context is taken into account with a temporal damping factor. **REBUS** takes the nearest item of the sum of user preference and sequential dynamics for recommendation.

Our contributions are summarized as follows: We develop a new method, **REBUS**, that integrates and unifies distance-based methods to capture both user preferences and sequential dynamics. Especially, personalized order Markov chains, thanks to sequential patterns, makes possible to select the part of the recent user history that is of most interest for the recommendation. In an empirical study over 13 datasets, we demonstrate that **REBUS** outperforms state-of-the-art algorithms. Furthermore, we show that the sequences we use to model user sequential dynamics can provide additional insights on the recommendations.

2 Notations

In this paper, U and I are used to denote the sets of users and items. Symbols u and i stand for individual users and items. The trace left by a user u is the sequence of items $s_u = \langle i_1, \ldots, i_p \rangle$, with $i_\ell \in I$, $\ell = 1 \ldots p$, and i_p being the most recent item. Let $s_u^{[1,t]}$ be the sub-string of s_u starting at the 1^{st} item and ending at the t^{th} one. We use I_{s_u} to denote the items that appear in s_u and $I_{s_u^{[1,t]}}$ to those that belong to $s_u^{[1,t]}$. Notations used throughout this paper are summarized in the table 1, as long as some examples.

Table 1: Notations.

U	User set.
I	Item set.
S	Set of user sequences.
F	Set of frequent substrings of S. $F = \{\langle 1\rangle, \langle 2\rangle, \langle 4\rangle, \langle 5\rangle, \langle 4,5\rangle\}$
s_u	Sequence of items associated to user u. $s_u = \langle 1,2,3,2,2,4,5\rangle$
I_{s_u}	Set of items that appear in s_u. $I_{s_u} = \{1,2,3,4,5\}$, $I_{s_u^{[1,4]}} = \{1,2,3\}$
m_{s_u}	Sequence of F that best matches s_u. $m_{s_u}\langle 4,5\rangle$
s_u^t, $m_{s_u}^t$	The item that appears in s_u or m_{s_u} at position t. $s_u^5 = 2$, $m_{s_u}^2 = 5$
$s_u^{[1,t]}$	The substring of s_u starting at position 1 and ending at position t. $s_u^{[1,4]} = \langle 1,2,3,2\rangle$
$m_u^{[1,t]}$	Sequence of F that best matches $s_u^{[1,t]}$. $m_u^{[1,4]} = \langle 2\rangle$
$\widehat{p}_{u,i,t}$	The prediction that measures the susceptibility for user u to choose item i while only considering $s_u^{[1,t]}$.
$>_{u,t}$	Personalized total order of user u at time step t.

3 Related Work

Recommendation methods consist in estimating values $\widehat{p}_{u,i}$ to measure the interest of user u for item i using real observations $\mathbf{D}$. To do that, a loss function $\mathcal{L}(\mathbf{D}, \widehat{\mathbf{P}})$ is optimized, that evaluates the proximity between $\mathbf{D}$ and $\widehat{\mathbf{P}} = (\widehat{p}_{u,i})$. Several approaches have been proposed so far, and we present below the recent approaches that are related to our proposal.

User preferences (Long Term Dynamics). Recommendation processes aim at identifying user preferences. In many state-of-the-art recommender systems Matrix Factorization (MF) [12] is used to model the interactions between users and items. It consists in decomposing a matrix $\mathbf{D}_{user \times item} = (d_{u,i})$ with $d_{u,i} = 1 \Leftrightarrow i \in I_u$ in a product of k-ranked matrices: $\mathbf{D}_{user \times item} \approx R \times Q$. The prediction $\widehat{p}_{u,i}$, that the user u chooses the item i, is estimated by the inner product

$$\widehat{p}_{u,i} = \langle \mathbf{R}_u, \mathbf{Q}_i \rangle,$$

with $\mathbf{R}_u$ and $\mathbf{Q}_i$ the latent vectors associated to user

u and item i. However, as usually users give implicit feedback on few items of I, the matrix $\mathbf{D}$ is sparse and these approaches suffer from loss of precision when the number of users and items grows.

In order to overcome these problems, other approaches such as FISM [10] decompose an implicit item-to-item similarity matrix in two k-ranked matrices $\mathbf{P}$ and $\mathbf{Q}$ so that

$$\widehat{p}_{u,i} \propto \beta_i + \beta_u + \frac{1}{|I_{s_u} \setminus \{i\}|^\alpha} \sum_{j \in I_{s_u} \setminus \{i\}} \langle \mathbf{P}_j, \mathbf{Q}_i \rangle,$$

where β_i, β_u are biases and $\frac{1}{|I_{s_u} \setminus \{i\}|^\alpha}$ is used to control the degree of agreement between items. Hence, the more i is similar to items already chosen by user u, the more likely i will be a good choice for u. Taking into account these transitive relations between items makes it possible to increase the quality of the recommendation.

Sequential Dynamics (Short Term Dynamics). Another trend in recommender system design is to use temporal information to consider how user preferences evolve through time. Short term dynamics can be modeled using Markov Chains of order one. It consists in using the probability $p(i \mid j)$ to predict i given that the last item in the sequence of the user is j. The transition matrix $\{p(i \mid j)\}_{ij}$ is decomposed in the product of two k-ranked matrices $\mathbf{M}_j$ and $\mathbf{N}_i$ that represent the latent/embedding vectors of item j and i, so that the probability of having i after j is estimated by the following inner product:

$$\widehat{p}_{i|j} \propto \langle \mathbf{M}_j, \mathbf{N}_i \rangle.$$

Unifying user preferences and sequential dynamics. Current recommendation methods accommodate user preferences and sequential dynamics as it has been observed that it increases their performances. FPMC [15] is one of the first method that uses both Matrix Factorization and first-order factorized Markov Chains. The probability that user u chooses item i just after taken item j is estimated by the sum of the following inner products:

$$\widehat{p}_{u,i|j} \propto \langle \mathbf{R}_u, \mathbf{Q}_i \rangle + \langle \mathbf{M}_j, \mathbf{N}_i \rangle$$

More recently, Fossil [9] proposes to associate methods based on the similarity between items like FISM with Markov Chains of order L:

$$\begin{aligned} \mathbf{M}_{s_u^{[t-L+1,t]}} &= \frac{1}{|I_{s_u} \setminus \{i\}|^\alpha} \sum_{j \in I_{s_u} \setminus \{i\}} \mathbf{P}_j + \sum_{k=1}^{L} (\eta_k + \eta_k^u) \mathbf{P}_{s_u^{t-k+1}} \\ \widehat{p}_{u,i|s_u^{[t-L+1,t]}} &\propto \beta_i + \langle \mathbf{M}_{s_u^{[t-L+1,t]}}, \mathbf{N}_i \rangle \end{aligned}$$

where t is the time-stamped of the last item in the user sequence. η_k and η_k^u are relative weights that control the long and short dynamics.

On top of that, PRME [5] improved FPMC and Fossil by replacing the inner products with Euclidean distances. As argued in [3, 5], metric embedding model brings better generalization ability than Matrix Factorization to represent Markov chains. The probability that user u takes the item i after item j is estimated by the sum of the following Euclidean distances:

$$\widehat{p}_{u,i|j} \propto -(\alpha \cdot ||\mathbf{R}_u - \mathbf{Q}_i||_2^2 + (1-\alpha) \cdot ||\mathbf{M}_j - \mathbf{M}_i||_2^2)$$

with α a weight that controls the long and short dynamics. TransRec [8], one of the most recent methods, is based on a novel metric embedding model that unifies user preferences and sequential dynamics with translation. To achieve this, items are embedded as points P_i in a latent transition space and users are modeled as translation vectors T_u in the same space:

$$\widehat{p}_{u,i|j} \propto \beta_i - d(P_j + T_u, P_i)$$

with $d()$ a distance function (L_1 or squared L_2), T_u a translation vector representing u, and P_i, P_j points in the transition space related to items i and j. The sequential dynamics is captured with first order Markov chains.

Summary and desiderata. Few approaches combines user preferences and sequential dynamics into a metric embedding model. The proposed approaches consider Markov chains of fixed small order L. In this paper, we aim to personalize the sequential dynamics of the users by identifying most salient items from user history, and using them in an unify way with user preferences in an embedding metric model.

4 The REBUS model

REBUS (a.k.a. Recommendation Embedding Based on freqUent Sequences) is a metric embedding model in which only items are projected. Their corresponding embedding vectors are influenced by both the user's preferences and by their dynamics. In order to take into account the part of the sequence that is the most characteristic of each user, we consider frequent sequential patterns that are shared by several users to increase the generalizing power of the approach. Once the most important items identified for a user, their temporal influence on the recommendation model is elegantly integrated thank to a temporal damping factor which makes possible to integrate, in a unified way, sequences of variable lengths. Embedding vectors are then learned using pairs of items (i, j), in such a way to maximize the probability that the prediction, that

item i is chosen by u at time-stamped t, is greater than the one of any other items j, if i is the item observed at position t in the sequence of u. This is done by instantiating the Bayesian personalized ranking optimization method [14] on our model.

4.1 Metric-based user preferences derived from user's past. To model the user preferences, we follow the way paved by FISM [10] and embed items into latent vectors so that the prediction for a user u to choose an item i varies in the opposite way to its distance to the items already chosen by the user:

$$\widehat{p}_{u,i} = -||\frac{1}{|I_{s_u^{[1,t]}} \setminus \{i\}|^\alpha} \sum_{j \in I_{s_u} \setminus \{i\}} P_j - P_i||_2^2.$$

In the learning phase, when one tries to estimate the prediction associated with the choice of an item i for a user u at a time-stamped t, it seems rightful to restrict the set of items to be considered to those prior to t. This choice, which is not the one usually made in the literature, is however useful for improving the prediction:

$$\widehat{p}_{u,i|s_u^{[1,t]}} = \widehat{p}_{u,i,t} = -||\frac{1}{|I_{s_u^{[1,t]}} \setminus \{i\}|^\alpha} \sum_{j \in I_{s_u^{[1,t]}} \setminus \{i\}} P_j - P_i||_2^2$$

4.2 Sequential dynamics evaluated by personalized contexts. As explained in the previous section, it has been shown that taking into account short-term individual dynamics improves the recommendation process. Our claim in this paper is that taking into account longer dynamics further improves the accuracy of the recommendation, especially when the considered sequences are a bit long. Contrariwise, in Markov chain model, the sequences considered to evaluate the conditional probability of an item are of fixed length L and the number of parameters to be estimated increases exponentially with L. We propose to alleviate this problem by accommodating Variable-order Markov Models (VMM) with string algorithms to learn conditional distributions of item appearance in user sequences such that (1) the length of the context s_u considered for the prediction is adapted in response to the available measures derived from the training data, and (2) some items in s_u may not be taken into account for the prediction if they do not reflect an order Markov dependencies based on the observed data.

As any Markov models, VMMs require to count the number of occurrences of an item i after a context s in the training data, these counts providing the basis for generating the predictions. But, rather than estimating all contexts of fixed length L, the VMMs select a set of contexts of different lengths, determined by the training data. The approach we follow to find the most adapted context for a user u at a time t is made of two steps: (1) the identification of a set of globally relevant contexts for all the users and all the times-stamped of the training set and (2) the construction of a tailored context for a given u and position t in its sequence based on the previous set of contexts.

Identifying potentially relevant contexts. Contexts that can account for user dynamics are those that can be generalized to several users. We propose to identify them thanks to the extraction frequent substrings within the set S of user sequences. The frequent substrings we consider appear in at least ***minCount*** and have for maximum size L. They constitute the set F.

Personalized context for u at time t. Once all the frequent substrings computed, our objective is to find which frequent substring to use as personalized context for a user u. To this end, we identify the sequence of F whose intersection with $s_u^{[1,t]}$ is the longest, recent actions (items). We first identify the most recent items we can use according to the collection of frequent substrings and then we look for the longest subsequence that ends with this item. We adapt the "*Exact matching with wild cards*" algorithm [7] to solve the above problem. It makes possible to discard items thanks to wildcard symbols (and then to obtain sequences with non-consecutive items) while comparing two sequences, and favors recent items. Such a sequence is denoted $m_{s_u^{[1,t]}}$. For instance, let $F = \{\langle 0\rangle, \langle 1\rangle, \langle 2\rangle, \langle 3\rangle, \langle 4\rangle, \langle 0,1\rangle, \langle 1,3\rangle, \langle 0,1,3\rangle, \langle 1,2\rangle, \langle 2,4\rangle, \langle 1,2,4\rangle\}$ be the set of frequent substrings. Given the user sequence $s_u^{[1,t]} = \langle 0,1,2,3,4,5\rangle$, the most recent item that can be exploited is 4 (item 5 does not appear in F), the longest sequence that ends with 4 is $\langle 1,2,4\rangle$ (the substring $\langle 1,2,*,4\rangle$ matches to $s_u^{[2,5]}$). If none of the sequences of F matches the sequence $s_u^{[1,t]}$ (e.g., no item from F matches to $s_u^{[1,t]} = \langle 7,8,9\rangle$), we set $m_{s_u^{[1,t]}} = i^\star$, where $i^\star$ is a dummy item embedded by our model. This dummy item plays a major role for cold start recommendation (for new users). We can notice that the personalized context $m_{s_u^{[1,t]}}$ can be different for a given user depending on the considered time-stamped.

4.3 The long-term and short-term metric embedding model. The embedding of items into a metric space has two main advantages. First it brings better generalization as distances preserve the triangle inequalities. Second, it makes possible to fully integrate into a

single distance the long and short-term dynamics.

$$\widehat{p}_{u,i,t} \propto -||\Big(\overbrace{\sum_{j \in I_{s_u^{[1,t]}} \setminus \{i\}} P_j}^{\text{User preferences}} + \overbrace{\sum_{k \in m_{s_u^{[1,t]}}} P_k}^{\text{Sequential dynamics}}\Big) - P_i||_2^2$$

In order to not over-weight items that appear in long transactions, that is items chosen with many other items by a user, long-term dynamics are normalized by the inverse of the length of the sequence.

$$\widehat{p}_{u,i,t} \propto -||\Big(\frac{1}{|I_{s_u^{[1,t]}} \setminus \{i\}|^\alpha} \sum_{j \in I_{s_u^{[1,t]}} \setminus \{i\}} P_j + \sum_{k \in m_{s_u^{[1,t]}}} P_k\Big) - P_i||_2^2$$

α is used to weight the importance given to long-term dynamics relative to short-term ones. Finally we add a bias term and a temporal damping factor that reduces the importance of an item of $m_{s_u^{[1,t]}}$ as it appears far back in the past:

$$\widehat{p}_{u,i,t} \propto -(\beta_i + ||\Big(\frac{1}{|I_{s_u^{[1,t]}} \setminus \{i\}|^\alpha} \sum_{j \in I_{s_u^{[1,t]}} \setminus \{i\}} P_j + \sum_{r=0}^{|m_{s_u^{[1,t]}}|} \eta_r P_{m^r_{s_u^{[1,t]}}}\Big) - P_i||_2^2)$$

The value of η_r increases with r to give more weight to recent items and promote the most recent items. To that end, we make η_r to follow the softmax (normalized exponential) [2] of one minus the cumulative Weibull distribution function:

$$\eta_r = \frac{e^{1-(1-e^{-\left(\frac{|m_{s_u^{[1,t]}}|-r}{y}\right)^x})}}{\sum_{r'=0}^{|m_{s_u^{[1,t]}}|} e^{1-(1-e^{-(r'/y)^x})}},$$

with $x = 2$ and $y = 7$.

Learning **REBUS** model requires determines the parameters $\mathbf{P}$ and β. Hyperparameters, such as α, L, **mincount** (and others given below related to the optimization algorithm) are set using a grid search strategy.

4.4 BPR optimization criterion. The goal for a recommendation system is not so much to predict the next item exactly, but to propose a ranking in which the item actually chosen by the user is as high as possible. Bayesian Personalized Ranking (BPR) [14] formalizes this problem as maximizing the posterior probability of the model parameter values θ given the order relation on items: $p(\theta\ |>_{u,t})$. Using Bayes' rule, it is proportional to $p(>_{u,t}|\ \theta)p(\theta)$. Thus, the goal is to identify the parameters that maximize the likelihood of correctly ordering the user preferences, that is having $i >_{u,t} j$ if i is ranked higher than item j by user u at time step t, i.e. for $i = s_u^t$. Assuming independence of items, users and times-stamped, this leads to estimate the model parameters by the maximum a posteriori probability (MAP):

$$\begin{aligned}\arg\max_\theta \quad &\propto \prod_{u \in U} \prod_{t=1}^{|s_u|} \prod_{j \neq S_u^t} p(s_u^t >_{u,t} j|\theta)\ p(\theta) \\ &\propto \sum_{u \in U} \sum_{t=1}^{|s_u|} \sum_{j \neq S_u^t} \ln\big(p(s_u^t >_{u,t} j|\theta)\big) + \ln\big(p(\theta)\big) \quad (4.1)\end{aligned}$$

The parameters of the **REBUS** model are the embedded vectors P_i and the bias term β_i for $i \in I \cup \{i^\star\}$. $p(s_u^t\ >_{u,t}\ j|\theta)$ is the probability that i is correctly ranked with respect to j by the model, that is:

$$\begin{aligned}p(i >_{u,t} j \mid \theta) &= p(\widehat{p}_{u,i,t}(\theta) >_{u,t} \widehat{p}_{u,j,t}(\theta)) \\ &= p(\widehat{p}_{u,i,t}(\theta) - \widehat{p}_{u,j,t}(\theta) >_{u,t} 0)\end{aligned}$$

This quantity is approximate by $\sigma(\widehat{p}_{u,i,t}(\theta) - \widehat{p}_{u,j,t}(\theta))$, where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the differentiable logisitc sigmoid function. Taking as prior probability for θ a normal distribution with zero mean and $\lambda_\theta I$ (with I the identity matrix) as variance-covariance matrix, the criterion to optimize (eq. 4.1) becomes:

$$\sum_{u \in U} \sum_{t=1}^{|S_u|} \sum_{j \neq s_u^t} \ln(\sigma(\widehat{p}_{u,s_u^t,t} - \widehat{p}_{u,j,t}) - \lambda_\theta ||\theta||^2 \quad (4.2)$$

As equation 4.2 is differentiable, it can be optimized using a gradient descent approach where the gradient is defined as

$$\begin{aligned}\frac{\partial f}{\partial \theta} &= \frac{\partial}{\partial \theta} \ln(\sigma(\widehat{p}_{u,s_u^t,t} - \widehat{p}_{u,j,t}) - \lambda_\theta \frac{\partial}{\partial \theta} ||\theta||^2 \\ &\propto \big(1 - \sigma(\widehat{p}_{u,s_u^t,t} - \widehat{p}_{u,j,t})\big) \frac{\partial}{\partial \theta}(\widehat{p}_{u,s_u^t,t} - \widehat{p}_{u,j,t}) - \lambda_\theta \theta\end{aligned}$$

We adopt the Stochastic Gradient Descent (SGD) known to be more efficient on large training set [15, 9, 8, 5]. P_i and β_i, for $i \in I \cup \{i^\star\}$ are initialized with random values from a uniform distribution over the interval $[0, 1]$. Then, iteratively, it consists in uniformly sampling a user u, a time-stamped t, and a 'negative' item j (with $j \notin m_{s_u^{[1,t]}}$) and updating the parameters θ using:

$$\theta \leftarrow \theta + \epsilon\Big(1 - \big(\sigma(\widehat{p}_{u,s_u^t,t} - \widehat{p}_{u,j,t})\big)\ \frac{\delta}{\theta}(\widehat{p}_{u,s_u^t,t} - \widehat{p}_{u,j,t}) - \lambda_\theta \theta\Big),$$

where ϵ is the learning rate. The explicit computation of gradient $\frac{\delta}{\theta}(\widehat{p}_{u,s_u^t,t} - \widehat{p}_{u,j,t})$ is given in annex. The iteration stops when a convergence criterion is met.

5 Experiments

In this section, we present our experimental results on the performance of **REBUS** for sequential recommendation. We begin by describing the 13 real-world datasets we have used, as well as the questions that the experiments are intended to answer. Then, we provide a thorough comparison with 8 state-of-the-art algorithms for item recommendation and sequential prediction. Eventually, we provide a deeper analysis of our algorithm, especially how frequent sequences are actually used in sequential recommendations. For reproducibility purposes, the source code and the data are made available[1].

5.1 Datasets and aims. To evaluate the performance of **REBUS** on various datasets from different domains, we consider 4 well-known benchmarks and a new dataset: **Amazon** was introduced by [13]. It contains Amazon product reviews from May 1996 to July 2014 from several product categories. We have chosen to use the 5 following diversified categories: *Automotive, Cellphones, Office product, Toys* and *video games*. **MovieLens 1M**[2] is a popular dataset including 1 million movie ratings from 6040 users between April 2000 and February 2003. We used the datasets pre-processed in selection of the x most recent ratings for each user, $x \in \{5, 10, 20, 30, 50\}$. **Epinions**[3] describes consumer reviews for the website Epinions from January 2001 to November 2013. This dataset was collected by the authors of [18]. **Foursquare**[4] depicts a large number of user check-ins on Foursquare website from December 2011 to April 2012. **Visiativ**[5] is a new dataset that gathers the downloads of Computer-Aided-Design resources by engineers and designers from November 2014 to August 2018. This dataset seems relevant to us to evaluate temporal recommendation since the downloaded documents are tutorials that users read to train themselves and improve their comprehension on some specific software.

For each dataset, ratings are converted into implicit feedback and we only considered users and items that have at least 5 interactions. The main characteristics of these datasets are reported in Table 2.

To evaluate the performance and the limits of **REBUS**, we propose to answer the following questions: What are the performances of **REBUS** compared to the ones of state-of-the-art algorithms? Does **REBUS** take benefit from sequential behaviors in a better way than Markov chains of fixed order do? Does **REBUS** provide additional insights on sequential recommendation?

Table 2: Main characteristics of the datasets.

	Datasets	#Users	#Items	$\sum$#seq.	avg (#seq.)	avg (#item occ.)
	Epinions	5015	8335	26932	5.3703	3.2312
	Foursquare	43110	13335	306553	7.1109	22.9886
	Visiativ	1398	590	16417	11.7432	27.8254
Amazon	Ama-Auto	34316	40287	183573	5.3495	4.5566
	Ama-Cell	68330	60083	429231	6.2817	7.1440
	Ama-Office	16716	22357	128070	7.6615	5.7284
	Ama-Toy	57617	69147	410920	7.1319	5.9427
	Ama-Game	31013	23715	287107	9.2576	12.1066
MovieLens	ML-5	6040	2848	30175	4.9959	10.5952
	ML-10	6040	3114	59610	9.8692	19.1426
	ML-20	6040	3324	111059	18.3873	33.4113
	ML30	6040	3391	152160	25.1921	44.8717
	ML50	6040	3467	215676	35.7079	62.2082

5.2 Comparison methods. We consider as baselines 8 methods from the state-of-the-art in both item and sequential recommendation: Popularity (**POP**), the naive baseline that ranks items according to their popularity [1]; Bayesian Personalized Ranking (**BPR-MF**) [14], that recommends items on the solely user preferences with matrix factorization techniques; Factorized Markov Chains (**FMC**) [15] that is based on the factorization of the item-to-item transition matrix; Factorized Personalized Markov Chains (**FPMC**) [15] that considers both user preferences and user dynamics thanks to matrix factorization and first-order Markov chains; Personalized Ranking Metric Embedding (**PRME**) [5] that embeds user preferences and user dynamics into two Euclidean distances; Hierarchical Representation Model (**HRM**) [17] that extends FPMC by using aggregation operations to model more complex interactions. We compare against HRM with both max pooling and average pooling, respectively denoted by HRMmax and HRMavg; Factorized Sequential Prediction with Item Similarity Models (**Fossil**) [9] that accommodates personalized high-order Markov chains with FISM [10]; Translation-based Recommendation (**TransRec**) [8] that unifies user preferences and sequential dynamics with translations. We use both L_1 and squared L_2 distances;

We evaluate our method **REBUS** in two settings: once with sequential dynamics based on frequent sequences matching as explained in section4, and the other one, $\textbf{REBUS}_{1MC}$, where the sequential dynamic is evaluated thanks to first order Markov chains. Table 3 provides a comparison of the methods defined above according to several criteria: whether they are personalized, sequentially aware, metric-based, integrated into a unified model and based on personalized order Markov

[1] https://goo.gl/ojKj9J
[2] http://grouplens.org/datasets/movielens/1m/
[3] http://epinions.com/
[4] https://foursquare.com/
[5] https://www.visiativ.com/en-us/

Table 3: Overview of the different models. **P**: Personalized?, **S**: Sequentially-aware?, **M**: Metric-based?, **U**: Unified model?, O: personalized Order Markov chains?

Property	P	S	M	U	O
Pop	✗	✗	✗	✗	✗
BPR-MF	✓	✗	✗	✗	✗
FMC	✗	✓	✗	✗	✗
FPMC	✓	✓	✗	✗	✗
HRM	✓	✓	✗	✓	✗
PRME	✓	✓	✓	✗	✗
Fossil	✓	✓	✗	✓	✗
TransRec	✓	✓	✓	✓	✗
REBUS$_{1MC}$	✓	✓	✓	✓	✗
REBUS	✓	✓	✓	✓	✓

chains.

5.3 Experimental settings. For each dataset, the user sequences are divided into 3 parts: (1) the most recent item that is used for the test and named *ground-truth item*, (2) the second most recent item used for validating when learning the model and (3) the other items used for the training. The accuracy of the models is measured by the area under the ROC curve (AUC) and with the Hit Rate at position 50 (HIT50):

$$AUC = \frac{1}{|U|}\sum_{u\in U}\frac{1}{|I\setminus I_{s_u}|}\sum_{j\in I\setminus I_{s_u}}\mathbb{1}(\widehat{p}_{u,g_u,t} > \widehat{p}_{u,j,t}),$$

where g_u is the ground-truth item of user u at the most recent time-stamped t. The indicator function $\mathbb{1}(b)$ returns 1 if the argument b is *True*, *0* otherwise. The measure calculates for each user how highly the ground-truth item has been ranked. The HIT50 function returns the average number of times the ground-truth item is ranked in the top 50.

To make fair comparisons, we optimized the same criterion S-BPR for all models using SGD with a fixed learning rate of 0.05. All regularization hyperparameters were selected from $\{0, 0.001, 0.01, 0.1, 1\}$ and we took the combination of hyperparameters that gives the best AUC evaluation. Finally, the dimension of the learned latent vectors is set to 10. Similarly to [8], we did not use the dropout technique mentioned in the HRM paper to make it comparable to other methods. For Fossil, we returned the best results from experiments with Markov Chains of order 1 to 3. For PRME, we report the best results found with parameter α equal to 0.4. For TransRec, we used L_1 and squared L_2 distances.

5.4 Performance study. The performance of the different methods on every dataset are reported in Table 4. We can observe that **REBUS** outperforms other models on most of the datasets, having the best AUC value. **REBUS** performs also well in term of HIT50, with an average rank of 2 among the 11 approaches. PRME performs very well on dense datasets (i.e., ML30, ML50). However, this model is outperformed by several others on sparse datasets. It gives evidence that having independent latent vectors to model user preferences and sequential dynamics is not an advantage for sparse datasets. Using personalized order Markov Chains (i.e., **REBUS**) makes it possible to reach better performance than using first order Markov chains (i.e., **REBUS**$_{1MC}$). This demonstrates the need to consider several items instead of the most recent one to model the sequential dynamics.

5.5 Study of the personalized contexts. To show evidences that fixed-order Markov chains are not able to capture the same sequential dynamics as those apprehended by **REBUS**, we study the characteristics of the sequences of F used in the recommendation. Column (A) from Table 5 reports the number of times a sequence different to $i^\star$ is exploited in **REBUS** as personalized context, for 6 of the datasets. For most of user sequences, **REBUS** takes benefit from a sequence different to $i^\star$.

Figure 2 reports the distribution of the sequences used to model the dynamics (sequences equal to $i^\star$ are omitted). Three criteria are considered: (1) their size, (2) their occupation within the user sequence, and (3) the position of the most recent item used (the most recent item in $s_u \cap m_{s_u^{[1,t]}}$). In most of the cases, short sequences are exploited to model the sequential dynamics. Nevertheless, they are of different sizes ranging from 1 to 15. The occupation of the sequences is rather low. **REBUS** often uses *compact* sequences. However, **REBUS** sometimes makes use of an arbitrary number of wild-cards which cannot be done by fixed order Markov chains. **REBUS** often takes into account the most recent item within the user sequences (i.e., position equal to 0) but there are some cases for which older items are used which cannot be done by first order Markov chains. Additionally, Table 5 reports the number of personalized contexts used in **REBUS** that can be modeled as first order Markov chains (B) or fixed L order Markov chains (C) and the ones that cannot be caught by Markov chains (D). The added value of our approach can be assessed by (C) and (D). These experiments demonstrate that **REBUS**'s modeling of sequential dynamics cannot be done with fixed order Markov chains.

Table 4: AUC and HIT50 for the different models on all datasets. The last row, called *Improvement*, shows in percentage of improvement of our method according to the other best model (best obtained results in bold).

Models	Metric	Epinions	Foursq	Visiativ	Auto	Cell	Office	Toy	Video	Avg (Amazon)	ML-5	ML-10	ML-20	ML-30	ML-50	Avg(ML)	Avg(All)
Pop	AUC	0.4575	0.9169	0.7864	0.5870	0.6959	0.6428	0.6240	0.7497	0.6599	0.7352	0.7722	0.7919	0.7981	0.8032	0.7801	0.7201
	HIT50	3.42%	55.65%	48.24%	3.84%	4.43%	1.66%	1.69%	5.17%	3.36%	12.65%	12.93%	12.82%	12.72%	13.13%	12.85%	14.49%
BPRMF	AUC	0.5359	0.9527	0.8384	0.6342	0.7611	0.7054	0.7280	0.8562	0.7370	0.7817	0.8309	0.8446	0.8514	0.8576	0.8332	0.7829
	HIT50	3.58%	66.65%	56.85%	3.80%	5.49%	4.33%	3.65%	12.47%	5.95%	17.80%	18.43%	17.39%	17.06%	16.54%	17.44%	18.77%
FMC	AUC	0.5476	0.9471	0.8341	0.6442	0.7548	0.6865	0.6943	0.8423	0.7244	0.7234	0.8012	0.8341	0.8441	0.8550	0.8116	0.7699
	HIT50	2.88%	63.47%	57.13%	2.49%	7.14%	3.02%	4.30%	14.30%	6.25%	14.44%	18.99%	20.55%	21.49%	22.22%	19.54%	19.42%
FPMC	AUC	0.5518	0.9480	0.8247	0.6415	0.7376	0.6859	0.7194	0.8524	0.7274	0.7405	0.8286	0.8476	0.8654	0.8802	0.8325	0.7787
	HIT50	2.93%	64.64%	55.34%	2.24%	2.81%	2.97%	4.42%	11.95%	4.88%	12.20%	19.32%	18.25%	25.83%	26.89%	20.50%	19.22%
HRM_max	AUC	0.5627	0.9523	0.8364	0.6562	0.7656	0.6983	0.7262	0.8566	0.7406	0.7693	0.8381	0.8538	0.8611	0.8706	0.8386	0.7882
	HIT50	2.53%	61.72%	56.99%	3.72%	6.30%	4.19%	3.84%	12.60%	6.13%	17.54%	21.78%	24.92%	22.42%	23.65%	22.06%	20.17%
HRM_avg	AUC	0.6033	0.9560	0.8483	0.6703	0.7891	0.6985	0.7581	0.8779	0.7588	0.7719	0.8448	0.8617	0.8664	0.8761	0.8442	0.8017
	HIT50	3.07%	60.87%	60.93%	4.46%	8.64%	5.50%	5.22%	15.25%	7.81%	21.26%	**25.82%**	23.40%	23.41%	24.14%	23.61%	21.69%
PRME	AUC	0.6138	0.9604	0.8618	0.6523	0.7987	0.7155	0.7411	0.8763	0.7568	0.7761	0.8354	0.8681	**0.8786**	**0.8884**	0.8493	0.8051
	HIT50	2.88%	65.94%	64.01%	3.77%	7.03%	6.43%	5.25%	16.29%	7.75%	16.81%	23.48%	**26.49%**	**27.84%**	**28.18%**	**24.56%**	22.65%
Fossil	AUC	0.5974	0.9607	0.8429	0.6887	0.7980	0.7219	0.7620	0.8776	0.7696	0.7916	0.8478	0.8624	0.8677	0.8699	0.8479	0.8068
	HIT50	3.51%	63.21%	55.63%	5.21%	7.50%	5.44%	4.77%	13.87%	7.36%	19.01%	23.36%	22.80%	22.27%	21.64%	21.82%	20.63%
TransRec	AUC	0.6030	0.9632	0.8649	0.6767	0.7997	0.7230	0.7478	0.8755	0.7646	0.7856	0.8430	0.8645	0.8689	0.8743	0.8473	0.8069
L1	HIT50	3.02%	65.99%	64.23%	5.14%	7.88%	**6.78%**	4.80%	15.39%	8.00%	20.30%	23.60%	23.86%	24.08%	24.26%	23.22%	22.26%
TransRec	AUC	0.6084	0.9621	0.8639	0.6902	0.8049	0.7258	0.7646	0.8844	0.7740	0.7936	0.8488	0.8711	0.8750	0.8791	0.8535	0.8132
L2	HIT50	**5.02%**	67.15%	63.08%	**5.56%**	**8.99%**	5.12%	**5.60%**	15.98%	8.25%	20.85%	24.67%	24.09%	24.42%	23.94%	23.60%	22.65%
REBUS	AUC	**0.6365**	0.9674	0.8676	**0.7028**	**0.8278**	**0.7557**	0.7787	0.8904	**0.7911**	**0.8092**	**0.8516**	**0.8732**	0.8756	0.8811	0.8581	**0.8244**
	HIT50	4.72%	69.37%	63.15%	5.29%	8.96%	6.64%	4.94%	16.93%	8.55%	21.89%	25.24%	24.52%	23.99%	24.90%	24.11%	**23.12%**
REBUS$_{1MC}$	AUC	0.6358	**0.9681**	**0.8688**	0.7026	0.8264	0.7464	**0.7793**	**0.8905**	0.7890	0.8082	0.8507	0.8729	0.8748	0.8812	0.8575	0.8235
	HIT50	4.39%	**69.71%**	**64.37%**	5.27%	**8.99%**	6.45%	5.48%	**16.94%**	**8.62%**	**21.96%**	24.69%	24.01%	21.96%	25.60%	23.64%	23.06%
Rank	AUC	1	1	1	1	1	1	1	1	1	1	1	1	2	2	1.4	1.14
	HIT50	2	1	1	2	1ex	2	2	1	1.6	1	2	3	5	3	2.8	2
Improvement	AUC	3.70%	0.51%	0.45%	1.83%	2.85%	4.12%	1.92%	0.69%	2.21%	1.90%	0.33%	0.24%	-0.34%	-0.81%	0.54%	1.38%
vs Best	HIT50	-5.98%	3.81%	0.22%	-4.86%	0.00%	-2.06%	-2.14%	3.99%	4.48%	3.29%	-2.25%	-7.44%	-13.83%	-9.16%	-1.83%	2.08%

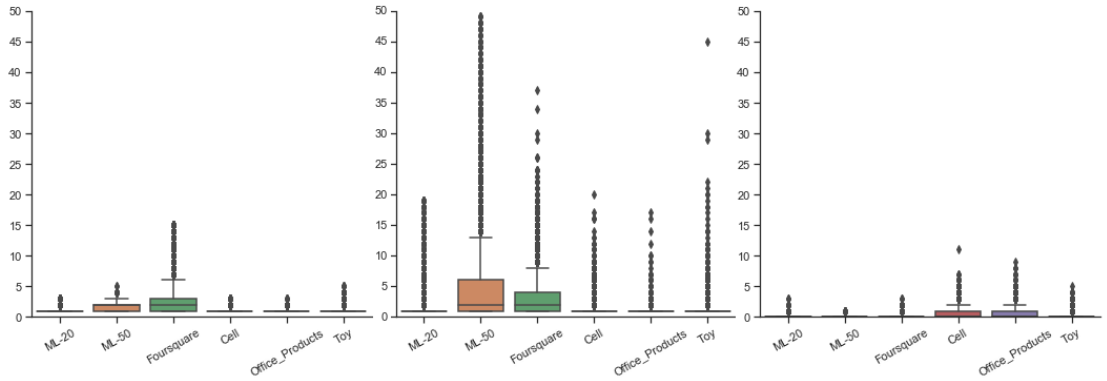

Figure 2: Boxplots: (1) size of the context; (2) Occupation of the context (i.e., # wild cards used + # items between the first and the last item of the context), (3) First position of the context in the user sequence (i.e.,# items before the occurrence of $m_{s_u^{[1,t]}}$).

Table 5: # user contexts equal to $i^\star$(A) equivalent to a first order Markov chain(B) or a L order Markov chain ($L > 1$)(C), that cannot be modeled by Markov chains(D).

Dataset	**(A)** $\#m_{s_u} = i^\star$	**(B)** $\#m_{s_u} \in$F MC1	**(C)** $\#m_{s_u} \in$F MCL	**(D)** $\#m_{s_u} \in$F VMC
Foursquare	25	11275	18905	12905
Ama-Cell	1674	47458	1006	18192
Ama-Office	804	10255	162	5495
Ama-Toys	532	44637	1328	11120
ML-20	0	4886	364	790
ML-50	0	2134	1265	2641

5.6 Importance of the user preferences with respect to the sequential dynamics. The user preference modeling part is very important to provide accurate recommendations. Models that only focus on user preferences (e.g., BPRMF) outperforms models that only consider the sequential dynamics (e.g., FMC). The hyperparameter α makes possible to weight the importance given at each part of the model: $\alpha = 1$ means that the two parts have the same importance, whereas the lower α, the greater the importance of the user preferences in the recommendations. The best value of α determined by grid search ($\alpha \in \{0.1, 0.2, \ldots, 1\}$) ranges from 0.5 (Foursquare) to 1 (Epinions, Ama-Auto, Ama-Cell, Ama-Office). For the majority of datasets, α is lower than 1, which demonstrates that the user prefer-

Table 6: Best value of α obtained by grid search.

Dataset	**REBUS**	**REBUS**$_{1MC}$
Epinions	1	0.9
Foursquare	0.5	0.5
Visiativ	0.6	0.6
Amazon		
Ama-Auto	1	0.5
Ama-Cell	1	1
Ama-Office	1	0.8
Ama-Toy	0.8	0.8
Ama-Game	0.8	0.8
MovieLens		
ML-5	0.7	0.6
ML-10	0.8	0.7
ML-20	0.7	0.7
ML-30	0.8	0.7
ML-50	0.8	0.9

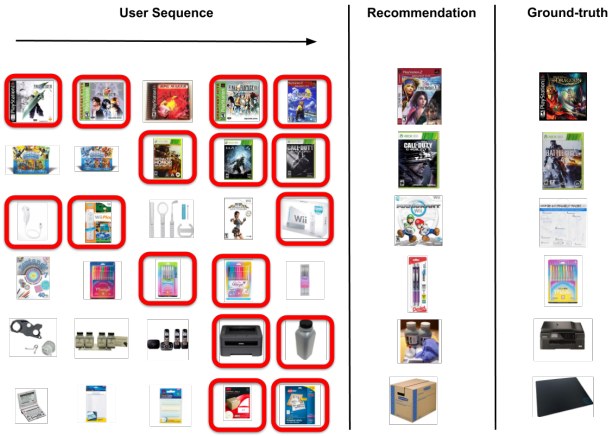

Figure 3: Recommendations for a random sample of users by **REBUS** on Amazon Games ([1-3] rows) and Office ([4-6] rows). The red squares are items in $m_{s_u^{[1,t]}}$.

ences have a bigger influence than the user short term history. Nevertheless, only the fruitful combinations of long term and short term user history makes it possible to provide accurate recommendations.

5.7 Recommendations. Figure 3 shows some recommendations made by **REBUS** on Amazon-Games and Amazon-Office datasets. As we can see on these examples, **REBUS** captures the sequential dynamics and recommended items that are similar to the ground truth. For instance, for the first user, **REBUS** recommends *Final Fantasy X-2* because the user sequentially bought the previous editions of the *Final Fantasy* game (the red squared items). The ground truth is a similar game: *The Legend of Dragoon*. On this example, the sequential dynamics is captured with 4-item sequence with a wild-card item within the user sequence. Some recommendations are based on smaller and more compact sequences, e.g., the two last recommendations use the two most recent items to capture the sequential dynamics. Based on the last two user actions – a laser printer then a toner and Adhesive Shipping Labels then Shipping Labels – **REBUS** respectively recommends refill kit for toner (instead of a new printer) and box mailers (instead of mouse pad).

5.8 Discussion. This experimental study demonstrates that our model **REBUS** provides the most accurate recommendations for most of the datasets. Frequent substrings allows to provide insights about the recommendations. We also tested **REBUS** with sequences instead of substrings. Results are very similar. However, the model based on frequent sequences is more costly to learn because the collection of frequent sequences is much greater than the collection of frequent substrings.

6 Conclusion

We introduce a new distance based method, **REBUS**, to address the problem of sequential recommendation. It provides personalized order Markov chains thanks to the use of sequential patterns which better model the sequential dynamics. Our model learns a representation of the user preferences and the sequential dynamics in the same Euclidean space. We performed thorough experiments on multiple real-world datasets and **REBUS** outperformed all the state-of-the-art models in most of the configurations. The sequences used to model the sequential dynamics also provide additional insights with the recommendations. This works opens up several avenues for further research such as the investigation of more sophisticated pattern mining techniques to both better model the user preferences and the sequential dynamics and provide explanation on the recommendations.

Acknowledgment. We thank R.He and J.McAuley [9, 8] who made available their codes and data.

References

[1] Aggarwal, C. Recommender Systems. *Springer US* (2016).

[2] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[3] Chen, S., Moore, J., Turnbull, D., and Joachims, T. Playlist prediction via metric embedding. *SIGKDD* (2012).

[4] Ding, Y., and Li, X. Time weight collaborative filtering. In *CIKM* (2005), ACM, pp. 485–492.

[5] Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y. M., and Yuan, Q. Personalized ranking metric embedding for next new POI recommendation. *IJCAI* (2015).

[6] Gomez-Uribe, C. A., and Hunt, N. The netflix recommender system: Algorithms, business value, and innovation. *ACM TMIS 6*, 4 (2016), 13.

[7] Gusfield, D. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, 1 ed. Cambridge University Press, May 1997.

[8] He, R., Kang, W., and McAuley, J. Translation-based Recommendation. *RecSys* (2017).

[9] He, R., and McAuley, J. Fusing similarity models with markov chains for sparse sequential recommendation. *ICDM* (2016).

[10] Kabbur, S., Ning, X., and Karypis., G. FISM: factored item similarity models for top-n recommender systems. *SIGKDD* (2013).

[11] Koren, Y. Collaborative filtering with temporal dynamics. *Commun. ACM 53*, 4 (2010), 89–97.

[12] Koren, Y., and Bell., R. Advances in collaborative filtering. *in Recommender Systems Handbook* (2011).

[13] McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. Image-based recommendations on styles and substitutes. *SIGIR* (2015).

[14] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme., L. BPR: Bayesian personalized ranking from implicit feedback. *Uncertainty in Artificial Intelligence* (2009).

[15] Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. Factorizing personalized markov chains for next-basket recommendation. *WWW* (2010).

[16] Resnick, P., and Varian, H. Recommender systems. *Communications of the ACM 40*, 3 (1997), 56–58.

[17] Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S., and Cheng, X. Learning hierarchical representation model for nextbasket recommendation. *SIGIR* (2015).

[18] Zhao, T., McAuley, J. J., and King, I. Leveraging social connections to improve personalized ranking for collaborative filtering. In *CIKM* (2014), pp. 261–270.