Translation-Based Sequential Recommendation for Complex Users on Sparse Data*

Hui Li, Ye Liu, Nikos Mamoulis, and David S. Rosenblum, *Fellow, IEEE*

Abstract—Sequential recommendation is one of the main tasks in recommender systems, where the next action (e.g., purchase, visit, click) of the user is predicted based on his/her past sequence of actions. Translating Embeddings is a knowledge graph completion approach which was recently adapted to a translation-based sequential recommendation (TransRec) method. We observe a flaw of TransRec when handling complex translations, which hinders it from generating accurate suggestions. In view of this, we propose a *translation-based recommender for complex users* (CTransRec), which utilizes *category-specific projection* and *temporal dynamic relaxation*. Using our proposed *Margin-based Pairwise Bayesian Personalized Ranking* and *Time-Aware Negative Sampling*, CTransRec outperforms state-of-the-art methods for sequential recommendation on extremely sparse data. The superiority of CTransRec, which is confirmed by our extensive experiments on both public data and real data obtained from the industry, comes from not only the additional information used in training but also the fact that CTransRec makes good use of this additional information to model the complex translations.

Index Terms—Sequential Recommendation, Sequential Behavior, Recommender Systems.

1 Introduction

Due to the availability of sequential data in recommender systems (e.g., user's click logs or purchasing orders with timestamps), sequential recommendation has become one of the core tasks for recommender systems [1]. Given a user's historical data, sequential recommendation aims at predicting his/her next action, e.g., next product to buy, next restaurant for lunch or next POI (i.e., point of interest) to visit. Unlike traditional *two-way* recommendation which predicts the user's preference over items, sequential recommendation models the *three-way* interaction among a user, an item he/she has selected and the next item he/she will select. Modeling three-way interactions raises new challenges due to the scale and inherent sparsity of real-world data [2].

Our industrial partner (anonymized as 'Alpha' in this paper) is a leading convenience store chain in the world. Alpha is setting up unmanned stores in the central business district (CBD) of several metropolitan areas and would like to incorporate sequential recommendation model into their unmanned shopping systems so that customers can get immediate recommendations at checkout. Historical data is extremely sparse in this application due to the large number of products in unmanned stores. On the other hand, user preference is traceable, as each store has its regular customers who are white-collar workers in CBD offices near the store. They can be identified from their mobile phones used to unlock the door and enter the unmanned store.

There are two directions in the study of sequential recommendation. Factorized Personalized Markov Chains (FPMC) [1] is the pioneering sequential recommendation method. FPMC utilizes tensor factorization and models three-way relations using two components: one is the interaction between the user and the next item to be selected and the other is the sequential history between previous items and the next item. Due to its success in handling large-scale data for sequential recommendation, FPMC has been widely studied and improved by follow up work [3, 4, 5]. Another line of work is using recurrent neural networks (RNNs) (e.g., Gated Recurrent Units (GRU) [6] and Long Short-Term Memory (LSTM) [7]) or convolutional neural networks (CNNs) (e.g., Convolutional Sequence Embedding [8]). However, techniques in these two categories cannot satisfy the requirements of companies such as Alpha [9, 10, 11]: MC based methods perform better on sparse data, though their accuracy is still low; Neural network based approaches may capture features well on denser data, but they are not suitable for fast online recommendation due to the long and complex training process.

Recently, He et al. [2] proposed a *translation-based recommender* (TransRec) for sequential recommendation. TransRec is inspired by the large body of work on *Knowledge Graph (KG) Completion* [12]. KG Completion models the user as a 'translation' from a previous action to the next action and it has higher accuracy compared to MC based models. On the other hand, TransRec is fast and scales well on large data compared to neural network based approaches, since it only considers transitions depending on the last action which is the most significant factor affecting user's next action (especially on sparse data) [13]. The aforementioned advantages make TransRec a good option for our client. However, TransRec inherits a flaw of TransE in sequential

* *Work supported in part by A*STAR SERC PSF under grant 15212000.*

- *Hui Li is with the Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University, Xiamen, Fujian, China. E-mail: hui@xmu.edu.cn. He is the corresponding author.*
- *Ye Liu is with School of Computing, National University of Singapore, Singapore. E-mail: liuye@comp.nus.edu.sg.*
- *Nikos Mamoulis is with Department of Computer Science and Engineering, University of Ioannina, Ioannina, Epirus, Greece. E-mail: nikos@cs.uoi.gr.*
- *David S. Rosenblum is with School of Computing, National University of Singapore, Singapore. E-mail: david@comp.nus.edu.sg.*

recommendation. Although TransE is known for its ability to model well 1-to-1 relations, it has been shown ineffective when handling 1-to-N, N-to-1, and N-to-N relations in KGs [12, 14]. As we will explain in Sec. 3.1, TransRec does not model well the so-called *complex user translation* and therefore it cannot distinguish complex embedding representations and give an accurate recommendation.

In this paper, we propose a *translation-based recommender for complex users* (CTransRec), which can model translations beyond 1-to-1. Recommender systems typically contain side information like item categories and timestamps in addition to user-item interaction data, which we utilize for solving the problem of complex translation. To sum up, the key contributions of this paper are:

- CTransRec maps item embeddings by performing item-category projection before translation, via which items have distinct representations when 'translated' by different (complex) users.
- CTransRec considers the influence of time, which is ignored by most previous work on sequential recommendation. We use temporal dynamic relaxation to relax the strict translation in TransRec. As we show, CTransRec with temporal dynamics is, in fact, an explicit way to deal with complex users.
- We analyze the relationship between CTransRec and existing KG completion methods and explain why CTransRec can outperform TransRec in theory.
- We propose *Margin-based Pairwise Bayesian Personalized Ranking* (MPBPR) and *Time-Aware Negative Sampling* (TNS) approaches, which optimize our translation based recommender.
- We conduct extensive experiments on a new real-world dataset Dianping and other real public datasets which show that CTransRec has superior performance compared to the state-of-the-art models on extremely sparse data. What is more, after testing CTransRec in several unmanned stores of Alpha during a nine-month period, we demonstrate its effectiveness in a real-world scenario.

Notation: We use lower-case fonts for scalars, bold lower-case fonts for vectors and bold upper-case font for matrices. For example, p is a scalar, $\mathbf{p}$ is a vector and $\mathbf{P}$ is a matrix.

2 PRELIMINARIES

We first introduce the sequential recommendation problem. Since translation-based recommendation is related to knowledge graph (KG) completion, we also briefly discuss this problem. Then, we revisit the TransRec model, which applies KG completion techniques to sequential recommendation.

2.1 Sequential Recommendation

The sequential recommendation problem can be defined as follows:

Definition 1 (Sequential Recommendation). Let $U = \{u_1, \dots, u_{|U|}\}$ be a set of users and $I = \{i_1, \dots, i_{|I|}\}$ be a set of items. The action sequence $\mathcal{S}^u$ for each user u is the sequence of items that the user has visited or selected in the past (i.e., clicks, purchases, check-ins): $\{a_1, \dots, a_{|\mathcal{S}^u|}\}$, where $a_s = \langle i_s, t_s \rangle$ denotes that u selected item i_s at time t_s and $t_{|\mathcal{S}^u|} \geq t_{|\mathcal{S}^u|-1} \geq \dots \geq t_1$. Given a user u and her action sequence $\mathcal{S}^u$, the goal of sequential recommendation is to predict the item(s) that u will next add to $\mathcal{S}^u$.

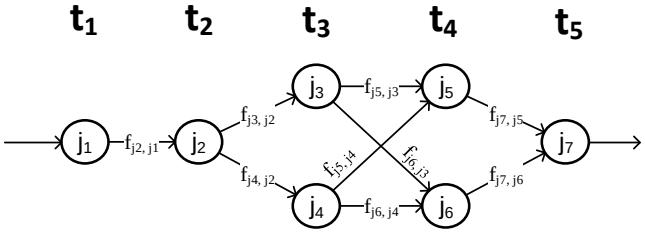


Fig. 1. Part of the Transition Graph for user u

Definition 1 is general enough to cover the *next-basket* sequential recommendation [1, 4, 5] where several items are contained in one action (e.g., two or more items are purchased together by u if they have the same timestamps in $\mathcal{S}^u$), the *session-based* sequential recommendation [6, 15, 16] where the action sequence for each user is divided into several short sequences and the *next-item* sequential recommendation [2, 3, 17] where each action involves just one item. Note that sequential recommendation only predicts the next target item(s), not the timestamp for the next action.

If we define the sequential recommendation problem using probabilities, then the goal is to use the last action in $\mathcal{S}^u$ to estimate the probability $Prob(j|u, i)$ that user u transits from the previous item i to the next item j (i.e., the predictor) [2]. Sequential recommendation models rank the possible next items j for the target user u according to $Prob(j|u, i)$ where i is the last item for u at the timestamp that the system needs to recommend items to u, and the action sequence $\mathcal{S}^u$ can be represented as a *transition graph*:

Definition 2 (Transition Graph). The Transition Graph for user u is formed by the action sequences $\mathcal{S}_u, u \in U$. From each sequence $\mathcal{S}_u$ we extract quadruples $\langle$previous item i, user u, next item j, time interval $f_{j,i}\rangle$, where $f_{j,i}$ indicates the time interval between the corresponding actions in $\mathcal{S}_u$ referring to items i and j and there are no items that u selected before j but after i. If there are several items that u picked at the same timestamp, they will form independent quadruples $\langle i, u, *, f_{*,i}\rangle$ with the same time interval $f_{*,i}$. Each quadruple is an edge of the graph that links the corresponding items.

Fig. 1 illustrates an example of part of the transition graph for user u. In the example, j_3 and j_4 were selected by u at the same timestamp t_3 and $f_{j_3,j_2} = f_{j_4,j_2}$. The transition graph models user actions in a sequential recommender. We denote two consecutive actions on items i and j by the same user u by either triple $\langle i, u, j\rangle$ or quadruple $\langle i, u, j, f_{j,i}\rangle$ if the time interval $f_{j,i}$ is needed.

2.2 Knowledge Graph Completion

Recent years have witnessed a rapid growth of knowledge graphs (KG) such as YAGO, Freebase, DBpedia and NELL [12]. A typical KG represents relationships between entities as triples $\langle$*head entity, relation, tail entity*$\rangle$, abridged as $\langle h, r, t\rangle$. For instance, $\langle$*Einstein, BornIn, Ulm*$\rangle$ represents

one fact in KG. KGs provide ways to retrieve, organize and manage human knowledge in structural forms, allowing AI systems to perform reasoning and facilitating tasks.

Due to data sparsity, knowledge graphs are usually far from complete. *Knowledge Graph completion* (KG completion) aims at predicting which triples not in a knowledge graph are likely to be true and fill in the missing piece of information into the KG. For example, given an incomplete triple $\langle$*DonaldTrump, PresidentOf, ?*$\rangle$, the system should be able to predict that the tail entity is *USA*. Translating Embeddings (TransE) [18] is the most representative model for KG completion. TransE represents both entities and relations as k-dimensional vectors (i.e., embeddings) in the same space $\mathbb{R}^k$. The relation is interpreted as the translation vector $\mathbf{r}$ and the embeddings for h and t can be bridged by $\mathbf{r}$ with low error: $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. For example, TransE models the above example as *DonaldTrump* + *PresidentOf* $\approx$ *USA*. TransE is motivated from word embedding [19], which represents words as vectors and then captures linguistic regularities, e.g., $king - man + woman \approx queen$. There is a large amount of work towards KG completion; readers can refer to [12] for a detailed survey

Sequential recommendation is similar to KG completion, since (i) it predicts values for incomplete triples of the form $\langle$previous item, user, ?$\rangle$ where ? is the next item and (ii) KG completion models multiple complex relations between entities in a similar manner as a sequential recommender models transitions between items.

2.3 TransRec

Motivated by TransE, TransRec learns a transition space $\Phi = \mathbb{R}^k$ where each item is represented by a k-dimensional vector $\mathbf{p} \in \Phi$ (i.e., embedding) and each user is expressed as a translation vector $\mathbf{q} \in \Phi$. Then, one triple $\langle$*previous item i, user u, next item j*$\rangle$ in user u's historical records $\mathcal{S}_u$ is modeled as $\mathbf{p}_i + \mathbf{q}_u \approx \mathbf{p}_j$, which means user u is the translation from item i to item j. According to the definition, $\mathbf{p}_j$ should be close to $\mathbf{p}_i + \mathbf{q}_u$ under some metric like Euclidean distance. Then a similar objective function like TransE is used as the predictor: $Prob(j|u,i) = \beta_j - ||\mathbf{p}_i + \mathbf{q}_u - \mathbf{p}_j||_2$, where β_j is the bias term for item j and $||...||$ indicates Euclidean distance.

3 CTransRec

KGs contain auxiliary information (e.g., types of entities, graph structure and textual information) that can help in the KG completion problem [12]. Analogously, user-item interactions in recommender systems have side information. In this section, we introduce our *translation-based recommender for complex users* (CTransRec), which utilizes item category and temporal dynamics. Due to the high relevance between translation based recommendation and KG completion, we also connect the techniques in CTransRec to some existing models for KG completion problem and explain why CTransRec is superior to TransRec, besides the intuitive reason that additional information is incorporated.

3.1 Complex User Translation

The superiority of TransRec over older sequential recommendation techniques that are not based on translation was shown in [2, 13]. However, TransRec inherits a flaw of TransE; TransE is well suited for 1-to-1 relations, but it fails when dealing with complex relations in knowledge graphs [12, 14]. Analogously, there are 1-to-N, N-to-1, and N-to-N user translations in sequential recommendation. For example, for user u in Fig. 1 and (previous) item j_2 there are more than one (simultaneous) next items in u's transition graph (i.e., j_3 and j_4).

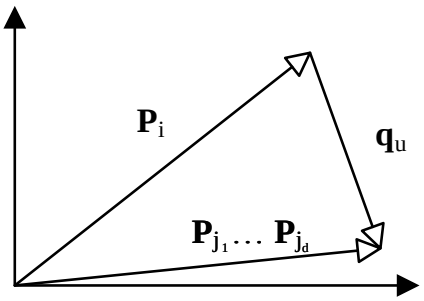


Fig. 2. TransRec cannot distinguish complex translations and the embeddings of $\mathbf{p}_{j_1}, ..., \mathbf{p}_{j_d}$ are close to each other.

We illustrate the flaw of TransRec via a more detailed example below. Before that, we formally define the concept of *complex user* as follows:

Definition 3 (Complex User). A user u, which is the translation between items that have more than one preceding or successive items via u, is a *complex user* or a *complex user translation*.

Example 1 (Flaw of TransRec). User u in Fig. 1 is complex, because there are 1-to-N ($t_2 \rightarrow t_3$), N-to-1 ($t_4 \rightarrow t_5$), and N-to-N ($t_3 \rightarrow t_4$) edges in the transition graph. TransRec enforces $\mathbf{p}_i + \mathbf{q}_u \approx \mathbf{p}_j$ for all $j = 1, ..., d$ such that $\langle \mathbf{p}_i, \mathbf{q}_u, \mathbf{p}_j \rangle$ exists; hence, $\mathbf{p}_{j_1} \approx ... \approx \mathbf{p}_{j_d}$ as shown in the graphical example of Fig. 2.

There are many such triples for the same customer in Alpha's shopping system (i.e., $\langle i, u, * \rangle$). For instance, consider triples <*cola, Alice, hamburger*> and <*cola, Alice, french fries*>, which mean that *Alice* first ordered cola then hamburger/french fries. The embeddings of hamburger and french fries will be close to each other in TransRec. Because of this, TransRec cannot distinguish items involved in complex user translations for sequential recommendation, which motivates the design of our CTransRec method.

3.2 Category-Specific Projection

To overcome the drawback of TransRec in dealing with complex user translation, we should allow an item to have distinct representations when chosen by different users. Our CTransRec model maintains, for each item and user, two vectors. The first vector is a classic embedding vector which encodes the features of an item or a user, while the second one is the *projection* vector which is used to map the item to different representation spaces.

Specifically, CTransRec introduces projection vectors $\mathbf{s}$ for each item and user-specific projection vectors $\mathbf{g}$ in addition to the embeddings $\mathbf{p}$ and $\mathbf{q}$. The embeddings of previous and next items are projected into different representation spaces by the following mapping matrices first:

$$\mathbf{M}_{u,i} = \mathbf{g}_u \mathbf{s}_i^T + \mathbf{I},\ \mathbf{M}_{u,j} = \mathbf{g}_u \mathbf{s}_j^T + \mathbf{I} \quad (1)$$

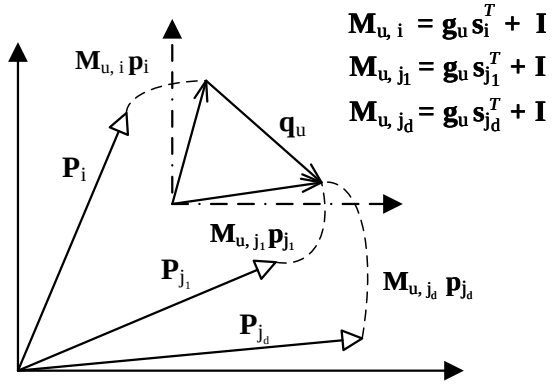


Fig. 3. Projection helps CTransRec distinguish $\mathbf{p}_{j_1}$ and $\mathbf{p}_{j_d}$ even they have same preceding item $\mathbf{p}_i$ and user translation $\mathbf{q}_u$.

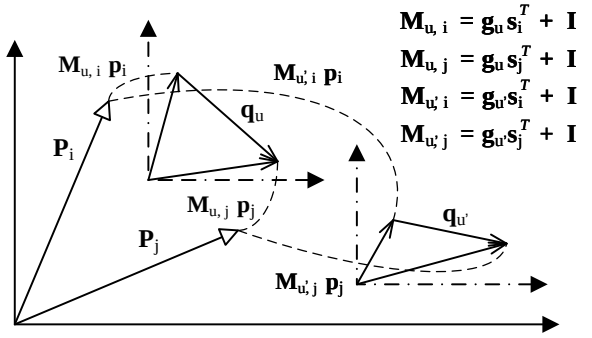


Fig. 4. Projection allows an item to have distinct representations when chosen by different users and even same embedding vectors will have different representations for different users.

where $\mathbf{I}$ is the identity matrix.

Then the translation of TransRec becomes:

$$\mathbf{M}_{u,i}\mathbf{p}_i + \mathbf{q}_u \approx \mathbf{M}_{u,j}\mathbf{p}_j. \quad (2)$$

CTransRec maps items into the user vector space via $\mathbf{M} = \mathbf{g}\mathbf{s}^T$ and then the *projected* item embedding $\mathbf{M}\mathbf{p}$ is translated in the user vector space via $\mathbf{q_u}$. By utilizing the projection, CTransRec allows each item to have multiple representations, which brings two benefits: (i) As shown in Fig. 3, the projected representations $\mathbf{M}_{u,j_1}\mathbf{p}_{j_1}$ and $\mathbf{M}_{u,j_d}\mathbf{p}_{j_d}$ are close to each other in the user vector space. But the different projection matrices $\mathbf{M}_{u,j_1}$ and $\mathbf{M}_{u,j_d}$ force the embedding $\mathbf{p}_{j_d}$ to be away from $\mathbf{p}_{j_1}$, even if they have the same preceding item and user translation. Thus, CTransRec addresses TransRec's flaw illustrated in Fig. 2. (ii) When involved in different translations (i.e., users), the same item has multiple distinct representations which capture the *personalized* user translations (see Fig. 4).

One may ask that why we do not use a simpler approach which maintains one projection $\mathbf{M}_u$ for each user. This way, the translation could be expressed as $\mathbf{M}_u\mathbf{p}_i + \mathbf{q}_u \approx \mathbf{M}_u\mathbf{p}_j$. The reason is two-fold. First, different items have different attributes and it is insufficient to reflect this diversity if all the items share the same projection matrix $\mathbf{M}_u$ for the same translation u. On the other hand, similar items should have similar projections for one user translation, which can be controlled by $\mathbf{s}_i$ in Eqs. 1 and 2. Eqs. 1 and 2 are a more fine-grained translation, as the projection $\mathbf{g}\mathbf{s}^T$ considers both user and item features from the specific item-user pair. Second, Eqs. 1 and 2 only involve vector-level operations (explained in Sec. 3.5) and they scale well to large data in practice, compared to adopting one projection matrix which requires matrix-vector products.

However, sequential records are typically very sparse in recommenders. Training one projection for every item is infeasible. Fortunately, item category information typically exists (e.g., genres of music in Yahoo! Music [20] and categories of POIs in Foursquare and Gowalla [21]) and daily human activities usually present category-level transition patterns [22]. Our solution to the sparsity problem is motivated by FPMC-LBPR [22], where instead of the user-item matrix the user-category matrix is utilized because it contains rich information and it is of smaller size. CTransRec adopts a category projection vector $\mathbf{s}$ in Eq. 1 (i.e., one projection vector for each category) instead of learning a projection for each item. By harnessing category-based projection, items of one category share the same projection which represents their homogeneity, while items in different categories have distinct mappings to reflect their diversity. Meanwhile, items with a small number of records can still be projected (i.e., their projection vectors can be learned using the records of other items in the same category). Finally, the probability that user u transits from the previous item i to the next item j (i.e., predictor) is:

$$Prob(j|u,i) \propto \beta_j - ||\mathbf{M}_{u,i}\mathbf{p}_i + \mathbf{q}_u - \mathbf{M}_{u,j}\mathbf{p}_j||_2, \quad (3)$$

where β_j is the bias term which captures the popularity of item j. CTransRec also enforces item embedding $||\mathbf{p}|| \leq 1$ and category-specific projection vector $||\mathbf{s}|| \leq 1$, which have been shown to be effective for KG completion [12, 14] and TransRec also adopts it for sequential recommendation. In this paper, we use Euclidean distance $||...||_2$ in the predictor, because it has been proved more interpretable and effective in translation based models [23]. Still, other metrics like Manhattan distance can also be utilized here.

In practice, one item may belong to multiple categories. For example, *Spicy Jalapeno Grilled Chicken Burger* is both *spicy food* and *fast food*. To model multiple categories for one item, we propose a general form of category projection vector: $\mathbf{s}_i$ for item i is the weighted summation of all category-specific vectors $\mathbf{o}$ wherein the item resides: $\mathbf{s}_i = \mathbf{o}_z/|C_i|, z \in C_i$ where C_i represents all categories that item i belongs to.

Connection to KG completion. TransRec is inspired by TransE, which models a relation in a knowledge graph as the translation from a head entity to a tail entity: $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$. Similarly, the prototype of the first predictor (Eq. 3) in CTransRec can be traced to TransD [14], one of the successors of TransE. In TransD, there are additional mapping vectors $\mathbf{m}_r$, $\mathbf{m}_h$ and $\mathbf{m}_t$. Head entity $\mathbf{h}$ and tail entity $\mathbf{t}$ are projected by matrices $\mathbf{M}_r^1 = \mathbf{m}_r\mathbf{m}_h^T + \mathbf{I}$ and $\mathbf{M}_r^2 = \mathbf{m}_r\mathbf{m}_t^T + \mathbf{I}$ first ($\mathbf{I}$ is the identity matrix). Then two projected head and tail vectors, together with the translation $\mathbf{r}$ are used in the same translation function as in TransE. Compared to TransD, CTransRec uses category-level projection vector for items instead of one projection vector for each vector, which helps to alleviate data sparsity; items without much information can benefit from other items in the same category.

3.3 Temporal Dynamic Relaxation

Modeling temporal changes in customer preferences is a traditional task in user modeling and many efforts have been made to incorporate temporal information into collaborative filtering [24, 25, 26]. For instance, Ding and Li [24] found appropriate time weights for items such that the items rated recently can contribute more to the prediction of the recommendation items.

On the other hand, temporal dynamics are rarely considered in KG completion [12]. For the sequential recommendation problem, as far as we know, most previous models consider the *order* of actions (e.g., *Bob first bought an MP3 player and then bought earphones*) and a few recent approaches [21, 27, 28] explicitly take time differences (i.e., *Bob bought earphones one day after he bought an MP3 player*) into consideration. Ignoring temporal information simplifies the model but makes recommendation less accurate.

Inspired by the work on temporal collaborative filtering [24, 26] which uses an exponential decay function to penalize the ratings made long ago, we use exponential growth to model the influence of time (i.e., the strength of translation) in sequential recommendation:

$$Prob(j|u,i) \propto \beta_j - e^{\lambda(t_j - t_i)}||\mathbf{M}_{u,i}\mathbf{p}_i + \mathbf{q}_u - \mathbf{M}_{u,j}\mathbf{p}_j||_2, \quad (4)$$

where λ is the growth rate and t_j represents the timestamp when user u performed an action on item j. By adopting an exponential growth function, the distance between the projected embeddings $\mathbf{p}_i$ and $\mathbf{p}_j$ increases with the time difference $t_j - t_i$.

The rationale behind Eq. 4 is that we want to drive the projected embedding of next item j close to $\mathbf{M}_{u,i}\mathbf{p}_i + \mathbf{q}_u$ during training if items i and j were visited within a short time interval. Items selected by a user in a short time should share some common properties in the corresponding user's translation space, which means that the distance between their representations is small. This is also known as the *temporal locality* of the user's behavior [26]. For instance, if a person enjoys watching one certain movie, she will search for related movies by the same directors/actors or of the same genre. Another example is the POI recommender where one users' check-ins within short time are close geographically.

Connection to KG completion. Even though there is no model for KG completion that considers temporal dynamics, we can find a method with similar format as what we used in the second predictor (Eq. 4). By simplifying Eq. 4, we have $\theta \cdot dist(\mathbf{p}_i + \mathbf{q}_u, \mathbf{p}_j)$ where $dist$ represents detailed distance function. In this way, we can treat θ (i.e., $e^{\lambda(t_j - t_i)}$) as a weight to relax the strict requirement $\mathbf{p}_i + \mathbf{q}_u \approx \mathbf{p}_j$. As demonstrated in Fig. 5, temporal information (i.e., θ) controls how close the projected embedding of next item j is to $\mathbf{M}_{u,i}\mathbf{p}_i + \mathbf{q}_u$, and therefore the predictor in Eq. 4 is called *temporal dynamic relaxation*. This is another way to distinguish items in complex user translations, besides allowing them to have distinct representations. Thus incorporating temporal dynamics into our translation based recommender is indeed solving the problem that TransRec cannot handle complex translations well. Coincidently, TransM [29], which is another successor of TransE, associates each fact $\langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle$ with a weight θ specific to $\mathbf{r}$ and the score function of TransE becomes $-\theta_r||\mathbf{h} + \mathbf{r} - \mathbf{t}||_2$.

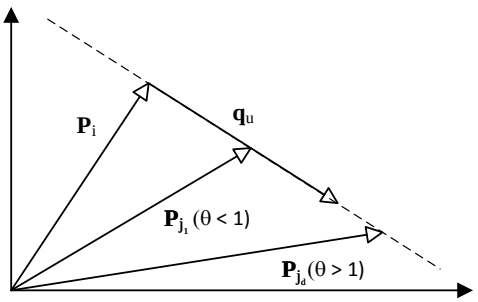


Fig. 5. Temporal information allows CTransRec to relax the strict requirement $\mathbf{p}_i + \mathbf{q}_u \approx \mathbf{p}_j$.

3.4 Learning Procedure

CTransRec follows the Pairwise Bayesian Personalized Ranking (PBPR) framework of FPMC [1]. We adopt margin-based optimization which is commonly used in training KG completion [23] and propose *Margin-based Pairwise Bayesian Personalized Ranking* (MPBPR). Together with MPBPR, *Time-Aware Negative Sampling* (TNS) which fits the temporal information used in CTransRec is utilized for sampling. The details of the procedure are as follows:

Ranking Optimization. The goal of PBPR is to rank the true item(s) higher than other items when making next-item prediction. In other words, $\mathbf{p}_j$ should rank higher than other items when making a prediction for $\langle \mathbf{p}_i, \mathbf{q}_u, ? \rangle$, if triple $\langle \mathbf{p}_i, \mathbf{q}_u, \mathbf{p}_j \rangle$ exists in u's history records. Margin-based optimization is the prevalent optimization method for translation based models in KG completion [12]. The core idea is to update only those embeddings for which the score for the corresponding true triple is not higher than the score for negative triples by a predefined margin γ. Margin-based optimization is a way to avoid overfitting since it stops training for those embeddings that are already good enough.

CTransRec combines PBPR and margin-based optimization, and uses Margin-based Pairwise Bayesian Personalized Ranking in the optimization function:

$$\hat{\Theta} = \arg\max_{\Theta} \sum_{u \in U} \sum_{j \in \mathcal{I}^u} \sum_{j' \notin \mathcal{I}^u} \min\Big(0, \sigma(\hat{w}_{u,i,j} - \hat{w}_{u,i,j'}) - \gamma\Big) - \Omega(\Theta), \quad (5)$$

where $\mathcal{I}^u$ represents the set of items that user u has visited, U indicates all users, γ is the margin separating positive and negative quadruples, σ is the sigmoid function, Θ is the parameter set, $\Omega(\Theta)$ is the $\mathcal{L}2$ regularizer and $\hat{w}_{u,i,j}$ is the predictor defined in Eq. 3 or Eq. 4.

Then, parameters can be updated via Stochastic Gradient Ascent (SGA) if the margin test can be passed. Due to space limitations, we do not give individual update rules for each parameter. Instead, we provide a general update rule, which can be used for all parameters:

$$\Theta \leftarrow \Theta + \eta \cdot \left(\sigma(\hat{w}_{u,i,j'} - \hat{w}_{u,i,j}) \frac{-\partial(\hat{w}_{u,i,j'} - \hat{w}_{u,i,j})}{\partial \Theta} - \omega \cdot \Theta \right), \quad (6)$$

Algorithm 1 Learning Procedure of CTransRec

1: Initialize each dimension of $\mathbf{s}, \mathbf{g}, \mathbf{p}, \mathbf{q}$ with uniform distribution in interval $(-\frac{6}{\sqrt{k}}, \frac{6}{\sqrt{k}})$. Set all β to zero.
2: **while** not converged **do**
3: **for** $i \in 1...|V|$ **do** ▷ V is all the training triples.
4: Uniformly sample one user u
5: Uniformly sample one positive item $j \in \mathcal{I}^u$
6: Uniformly sample one negative item $j' \notin \mathcal{I}^u$
7: $f_{j,i} \leftarrow \sigma(t_j - t_i)$ ▷ i is the direct preceding item of j.
8: $f_{j',i} \leftarrow \sigma\Big(\text{Rand}(\bar{f}_{j'}, +\infty)\Big)$
9: **if** $\sigma(\hat{w}_{u,i,j} - \hat{w}_{u,i,j'}) < \gamma$ **then**
10: Update all parameters according to Eq. 6
11: Normalize $\mathbf{s}$ and $\mathbf{p}$ for i, j, j' if $||\mathbf{s}|| > 1$ or $||\mathbf{p}|| > 1$

where η is the learning rate and ω is the regularization weight. Note that CTransRec can be updated using other methods like AdaGrad [30] and AdaDelta [31].

Time-Aware Negative Sampling. TransRec adopts the traditional negative sampling method for KG completion [12] which samples for each positive triple $\langle \mathbf{p}_i, \mathbf{q}_u, \mathbf{p}_j \rangle$ one negative triple $\langle \mathbf{p}_i, \mathbf{q}_u, \mathbf{p}_{j'} \rangle$ such that $j' \notin \mathcal{I}^u$. This method, however, cannot be adopted by CTransRec directly since CTransRec relies on the timestamps of both positive item j and negative item j' when computing the time intervals $f_{j,i} = t_j - t_i$ and $f_{j',i} = t_{j'} - t_i$ used for exponential growth in the predictor of Eq. 4. To obtain the time interval $t_{j'} - t_i$ for a negative triple, CTransRec counts for each item j the maximum time interval $\bar{f}_j$ between j and any of its direct preceding items. Then, the time interval for the negative triple with negative sample item j' is uniformly sampled within range $(\bar{f}_{j'}, +\infty)$. Here, we follow the *closed-world assumption* commonly used for knowledge representation [32]: what is not currently known to be true, is false. Since the maximum time interval between item j' and its direct preceding item is known to be $\bar{f}_{j'}$ in the existing data, we presume that any time interval longer than $\bar{f}_{j'}$ cannot be true for item j' and therefore the time interval for the negative triple can be sampled within this range. Finally, $f_{j',i}$ is passed through the sigmoid function before being used in the predictor, so that $e^{f_{j',i}}$ is not too large for the computer to process. Time intervals for positive triples also follow the same step.

Overall Procedure. Putting MPBPR and TNS together, the overall learning procedure of CTransRec is shown in Alg. 1. We follow the initialization method for embeddings used in TransRec. After the model is trained, CTransRec (like typical sequential recommenders [2, 4]) ranks the possible next items j for u according to the values of the predictor $Prob(j|u, i)$ where $j \in I$ and i is the last item for u at the timestamp that the system needs to give recommendations to u. The items with highest predictions are then recommended to u.

3.5 Model Complexity

The number of parameters in CTransRec is $2k|U| + 2k|I| + |I|$, since it requires two k-dimensional vectors (i.e., embedding vector and projection vector) for each user and item and one bias for each item. Compared to TransE which has $k|U| + k|I| + |I|$ parameters, observe that CTransRec does not require learning many more parameters, even though it uses additional information.

TABLE 1
Data statistics after cleaning

Data	# of Users	# of Items	# of Categories	# of Actions	Sparsity
Yelp	204,748	115,763	1,213	3,091,984	99.99%
GoogleLocal	346,238	501,115	7,141	2,571,085	99.99%
Dianping	616,331	10,979	247	3,868,306	99.94%

Since the time complexity depends on the number of positive/negative quadruples ⟨previous item, user, next item, time interval⟩ which can pass the margin test (line 9 in Alg. 1), we cannot evaluate the overall time complexity accurately. However, we can estimate the time complexity for updating one pair of positive/negative quadruples (lines 10-11 in Alg. 1), which depends on the update rule in Eq. 6. The projected representation in the predictor $\hat{w}_{u,i,j}$ can be computed using matrix associativity: $\mathbf{g}_u \mathbf{s}_i^T \mathbf{p}_i = \mathbf{g}_u(\mathbf{s}_i^T \mathbf{p}_i)$. Since $\mathbf{s}_i^T \mathbf{p}_i$ is indeed the inner product between two k-dimensional vectors $\mathbf{s}_i$ and $\mathbf{p}_i$, computing $\mathbf{g}_u \mathbf{s}_i^T \mathbf{p}_i$ actually contains two steps: compute inner product $\mathbf{s}_i^T \mathbf{p}_i$ and then do element-wise multiplication between the result (scalar) and $\mathbf{g}_u$. Therefore the computation only requires vector-level operations, which have a complexity of $O(2k)$. The time complexity for $\hat{w}_{u,i,j'} - \hat{w}_{u,i,j}$ is dominated by the computation of representation for CTransRec. After obtaining $\hat{w}_{u,i,j'} - \hat{w}_{u,i,j}$, CTransRec needs to update eight k-dimensional vectors (i.e., embeddings and projection vectors for u, i, j, j') and three item bias values for i, j, j' according to Eq. 6. Thus, CTransRec needs $O(2k + 8k + 3)$ time to update the corresponding parameters in one pair of positive/negative quadruples. Similarly, the complexity of TransRec consists of two parts: $\hat{w}_{u,i,j'} - \hat{w}_{u,i,j}$ and updating, i.e., $O(k + 4k + 3)$ in total.

In summary, CTransRec requires roughly two times of the number of parameters and time compared to TransRec. Specifically, its time complexity for one pair of positive/negative quadruples is linear to the dimensionality k. Assume the number of pairs which pass the margin test is $|B|$, then the complexity for one iteration is $O(k|B|)$ in CTransRec. Thus, CTransRec scales well for large data in practice.

4 Experiments

In this section, we conduct an experimental study using real datasets to answer the following questions:

- **Q1**: Does the proposed CTransRec model outperform the state-of-the-art sequential recommendation approaches?
- **Q2**: Is CTransRec sensitive to hyperparameters and update method?
- **Q3**: For next-category prediction, does CTransRec achieve better results than a baseline method which also considers category information?
- **Q4**: What is the performance of CTransRec when applied to the unmanned stores of Alpha?

We first present our experimental settings and then answer Q1, Q2, Q3 and Q4 in Secs. 4.2, 4.3, 4.4 and 4.5, respectively.

4.1 Experimental Settings

Data. We use three real datasets for our experiments. As far as we know, they are the largest datasets containing category and temporal information.

TABLE 2
Results on sequential recommendation ($k = 10, \eta = 0.5$)

Data	Metric	FPMC	PRME	HRM	Time-LSTM	Caser	TransRec	CTransRec$_c$	CTransRec$_{ct}$	Percentage
Yelp	AUC	0.8412	0.9134	0.9014	0.8261	0.8516	0.9238	0.9442	**0.9512**	2.97%
	Hit@50	5.32%	6.34%	6.12%	3.24%	5.47%	7.43%	8.17%	**8.45%**	13.73%
	Time (Secs)	84.92	59.64	56.73	62451.32	84654.15	108.68	229.19	248.75	28.88%
GoogleLocal	AUC	0.7211	0.7986	0.7321	0.7348	0.7612	0.8177	0.8512	**0.8664**	5.96%
	Hit@50	3.21%	8.52%	3.99%	3.45%	6.98%	10.98%	13.21%	**13.54%**	23.32%
	Time (Secs)	95.14	77.70	75.04	84798.42	12456.42	93.04	198.78	209.15	24.80%
Dianping	AUC	0.8023	0.7498	0.7531	0.7419	0.7849	0.7972	**0.8201**	0.8141	2.87%
	Hit@50	6.34%	4.45%	4.21%	4.41%	6.01%	6.26%	**7.01%**	6.97%	11.98%
	Time (Secs)	71.06	45.00	47.08	54781.35	79478.01	94.63	204.22	221.33	15.81%

- **Yelp**[1] dataset was provided for the tenth round of the *Yelp Dataset Challenge*. It contains 4,736,897 reviews and ratings from 1,183,362 users over 156,638 local businesses in 1,240 categories.
- **GoogleLocal**[2] from *Google* includes 10,487,697 reviews and ratings from 4,537,091 users on 3,061,290 local businesses in 48,013 categories across five continents.
- **Dianping**[3]. We constructed a new benchmark for evaluating sequential recommendation by crawling data from *Dianping*[4], which contains 772,697 customers' ratings and reviews (4,822,754 in total) for 11,723 restaurants in Shanghai from April 2003 to November 2013. The number of categories is 248.

Following [2], we discard users/items with fewer than 5 actions and categories with fewer than 5 items. Since our approaches focus on implicit feedback, we regard each purchase/rating/check-in as one action. Tab. 1 shows statistics for all datasets after cleaning. These three datasets are extremely spare (sparsity $= 99.9\%$).

Competitors. We compare CTransRec to seven approaches: FPMC [1], PRME [4], HRM [5], Time-LSTM [28][5], Caser [8][6], TransRec [2][7] and FPMC-LBPR [22]. FPMC-LBPR is a competitor for the next-category prediction task and the other approaches are used to evaluate sequential recommendation. In our experiments, average pooling for both two steps is used for HRM as it shows better performance in [2]. Detailed explanation of these models can be found in Sec. 5.2.

We test two versions of our CTransRec approach. CTransRec$_c$ indicates our method with category-specific projection (predictor in Eq. 3) and CTransRec$_{ct}$ represents our method using both category-specific projection and temporal dynamics relaxation (predictor in Eq. 4).

Environment. Experiments were conducted on a machine with two Intel(R) Xeon(R) CPU E5-2637 v4 @ 3.50GHz, 128 GB of main memory, three NVIDIA GeForce Titan X Pascal (12 GB memory for each) and Debian 9. All methods except for Time-LSTM and Caser are single-threading and implemented using standard C++ libraries. Time-LSTM and Caser are implemented using Theano and PyTorch, respectively.

Evaluation. We follow the evaluation settings in [2] and partition the edges of the transition graph for each user into three parts. The 10% most recent timestamps are used for testing, the second 10% most recent timestamps are used for validation and the remaining timestamps are used for training. If a user does not have enough timestamps (≤ 10), all the edges will be used for training. For instance, if t_4 in Fig. 1 is used for validation, edges $\langle j_3, u, j_5\rangle$, $\langle j_3, u, j_6\rangle$, $\langle j_4, u, j_5\rangle$ and $\langle j_4, u, j_6\rangle$ will be used for validation. The performance of different models are evaluated using *Area Under the ROC Curve* (AUC) and *Hit Rate at position 50* (Hit@50):

$$AUC = \frac{1}{|\mathcal{T}|} \sum_{<i,\,u,\,j,\,f_{j,i}>\in\mathcal{T}} \frac{1}{|\mathcal{I}\setminus\mathcal{S}^u|} \sum_{j'\in\mathcal{I}\setminus\mathcal{S}^u} \mathbf{1}(R_{u,j} < R_{u,j'}),$$
$$Hit@50 = \frac{1}{|\mathcal{T}|} \sum_{<i,\,u,\,j,\,f_{j,i}>\in\mathcal{T}} \mathbf{1}(R_{u,j} \leq 50), \tag{7}$$

where $\langle i,\ u,\ j,\ f_{j,i}\rangle$ means $\langle$*previous item* i, *user* u, *next item* j, *time interval* $f_{j,i}\rangle$, $R_{u,i}$ is the rank of item i for user u among all items, $\mathcal{T}$ is the test set and $\mathbf{1}(e)$ returns 1 if e is true and 0 otherwise. Note $\mathcal{S}^u$ in Eq. 7 only contains training records.

Hyperparameters. We set learning rate $\eta = 0.5$, dimensionality $k = 10$ and investigate the best hyperparameters for the regularization weight ω in {0, 0.001, 0.01, 0.1, 1}, margin γ in {0.01 0.05 0.1 0.5 0.8}, growth rate λ in {0.01, 0.1, 0.5, 1} and α[8] in {0.2, 0.5, 0.8} on the validation set for the corresponding models. For Caser, we carefully follows the instructions in [8] to tune all its hyperparameters[9]: The dropout rate is set to 0.5, the Markov order L ranges in $\{1, \cdots, 9\}$, the height h of horizontal filters ranges in $\{1, \cdots, L\}$, the target number T ranges in $\{1, 2, 3\}$, the activation functions ϕ_a and ϕ_c range in {*identity*, *sigmoid*, *tanh*, *relu*}. For each height H, the number of horizontal filters is in $\{4, 8, 16, 32, 64\}$ and the number of vertical filters is in $\{1, 2, 4, 8, 16\}$.

All methods are trained until convergence and we report results on the test set under the selected hyperparameters in Sec. 4.2. In addition to the above search ranges, we also report the effect of η and k when using different η in {0.05, 0.3, 0.5, 0.8} and k in {10, 30, 50, 80, 100} in Sec. 4.3.

4.2 Overall Performance (Q1)

Tab. 2 illustrates the overall performance of all methods on the three datasets when $k = 10, \eta = 0.5$ and other hyperparameters were tuned to their best values. The last column 'Percentage' shows the improvement percentage (or the overhead increase) of our models (either CTransRec$_c$ or CTransRec$_{ct}$) over the competitors with best AUC and

1. https://www.yelp.com/dataset/challenge
2. http://jmcauley.ucsd.edu/data/googlelocal
3. http://lihui.info/data/dianping.html
4. http://www.dianping.com
5. https://github.com/DarryO/time_lstm
6. https://github.com/graytowne/caser_pytorch
7. https://sites.google.com/view/ruining-he

8. α controls the balance between user-item interaction and item-item interaction in PRME. See Tab. 4 in Sec. 5.2 for explanation.

9. Please refer to [8] for detailed explanations of Caser's hyperparameters.

Hit@50. The best results on AUC and Hit@50 are reported in bold.

We can see that PRME and HRM have better performance than FPMC. As we will explain in Sec. 5.2, PRME and HRM are essentially extensions of FPMC. PRME replaces the inner product used in FPMC with Euclidean distance and the distance of two objects measures the strength of their sequential relation. Since the metric space obeys the *triangular inequality*, Euclidean distance is more suitable than inner product for modeling ranking. HRM uses aggregation operations to model more complicated interactions among different factors beyond the independence assumption, instead of the simple interactions used in FPMC. Time-LSTM and Caser do not show better performance compared to traditional MC-based methods, as the datasets are extremely sparse. The above observation confirms the conclusion in [9] that MC-based methods perform best in sparse datasets, where model parsimony is critical, while neural network-based models perform better in denser datasets where higher model complexity is affordable.

Compared to non-translation based models, translation-based models TransRec and CTransRec achieve considerable improvements. This demonstrates the advantage of incorporating translation-based methods for KG completion into models for sequential recommendation. Due to the ability to handle complex user translation, CTransRec is significantly better than TransRec. To be specific, CTransRec improves AUC by 2.87%-5.96% and Hit@50 by 11.98%-23.32% compared to TransRec. CTransRec$_c$ outperforms all previous work including TransRec on all three datasets, illustrating the impact of category-specific projection. Compared to CTransRec$_c$, CTransRec$_{ct}$ has better results on datasets Yelp and GoogleLocal, which shows that taking temporal information into account improves the quality of sequential recommendation. For dataset Dianping, CTransRec$_{ct}$ is marginally worse than CTransRec$_c$. A possible reason is that dataset Dianping contains fewer categories and there are many items in each category sharing similarity. Hence only using category-specific projection already deals with complex user translations well and using the temporal information in addition only introduces noise. In practice, we suggest deploying CTransRec$_{ct}$ for a system with more item categories while CTransRec$_c$ is more suitable for a system with fewer item categories where each category contains more homogeneous items.

The runtime cost of the translation based methods TransRec and CTransRec is higher than that of FPMC, PRME and HRM. Compared with TransRec, CTransRec has a 15%-28% time overhead. Considering the improvement of recommendation quality over competitors, we believe CTransRec is worth the additional cost. What is more, the sampling method in Algorithm 1 can be easily sped up using *HOGWILD!*-style parallelization [33] like many KG completion methods[10] (i.e., each thread samples independently) and the cost can be significantly reduced. Compared to methods without neural networks (single-threading in our experiments), Time-LSTM and Caser require much longer training time even though they utilize the power of GPUs. This shows that neural network-based methods are not suitable for the scenario where a fast recommendation is required (e.g., online recommendation).

10. https://github.com/thunlp/Fast-TransX

4.3 Sensitivity (Q2)

We explore the effect of dimensionality k, learning rate η, regularization weight ω, margin γ and growth rate λ on CTransRec. To evaluate the effect of ω, γ and λ, we report the results when different values are used while fixing $k = 10$, $\eta = 0.5$ and grid search is conducted to keep other hyperparameters optimal in their ranges. For example, we evaluate the sensitivity to ω by reporting the performance when ω is one of $\{0, 0.001, 0.01, 0.1, 1\}$, $k = 10$, $\eta = 0.5$ and other hyperparameters are tuned to be optimal. When evaluating the effect of k, we set $\eta = 0.5$ and tune other hyperparameters to be optimal. For the results of η, we set $k = 10$ and also tune other hyperparameters so that models show their best performance. Since we show AUC and HIT@50 for all the datasets in this section (i.e., Figs. 6, 7, 8, 9 and 10 in the following), we indicate the measure of the y-axis and the dataset at the top of each plot (e.g., "Yelp (AUC)") in order to save space.

Dimensionality k. We increase the dimensionality from 10 to 100 while tuning other parameters so that each model has its best performance at a specific k. Fig. 6 compares CTransRec$_{ct}$ with TransRec and shows that the performance of both methods improves as k increases and CTransRec$_{ct}$ consistently outperforms TransRec. On dataset Yelp, the AUC of CTransRec$_{ct}$ does not increase significantly as k rises. A possible explanation is that a small k already generates very high AUC (0.95) for CTransRec$_{ct}$ and increasing k does not benefit the model.

Learning rate η. Fig. 7 demonstrates the effect of η on TransRec and CTransRec$_{ct}$. We can conclude that η is relatively easy to choose as CTransRec performs best when η is close to 0.5 on all three datasets. CTransRec outperforms TransRec for all values of η.

Regularization weight ω. The influence of regularization weight ω on TransRec and CTransRec$_{ct}$ is evaluated in Fig. 8. Although TransRec performs similarly to CTransRec on dataset Dianping for some values of ω, in most cases CTransRec has better performance. ω is also easy to tune in practice since its changes are unimodal.

Margin γ. Margin γ affects MPBPR, as it controls how many parameters need to be updated under the current state of the training model. In fact, γ helps CTransRec to avoid overfitting. From the results shown in Fig. 9, we can find that a small γ (0.05) typically produces best results for CTransRec$_c$ while CTransRec$_{ct}$ requires a little larger γ (0.1). The larger γ is, the more possible that a pair of positive/negative quadruples can pass the margin test and the method will need to update the corresponding parameters. Since CTransRec$_{ct}$ additionally adopts the relaxing translation as shown in Fig. 5, it should have more flexibility to deal with complex translation and avoid overfitting compared to CTransRec$_c$. Therefore, although more pairs pass margin test and more parameters are updated under larger γ, CTransRec$_{ct}$ still produces good results. The only exception is dataset Dianping where the performance of CTransRec$_{ct}$ drops as γ increases. As explained in Sec. 4.2,

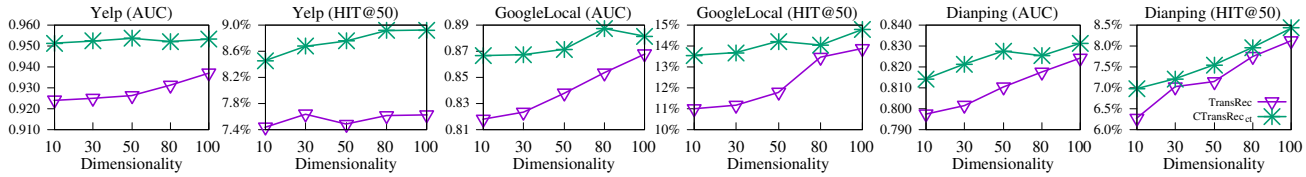


Fig. 6. Performance for different values of k ($\eta = 0.5$)

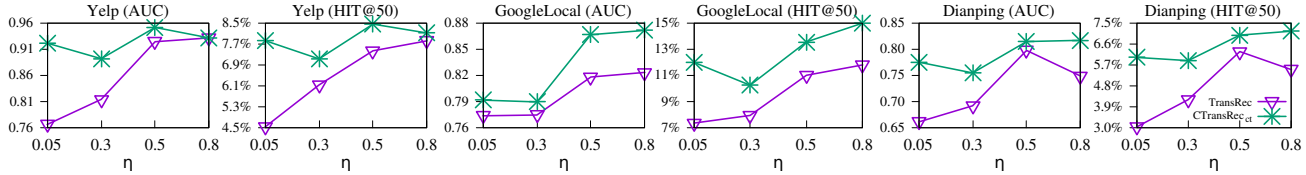


Fig. 7. Performance for different values of η ($k = 10$)

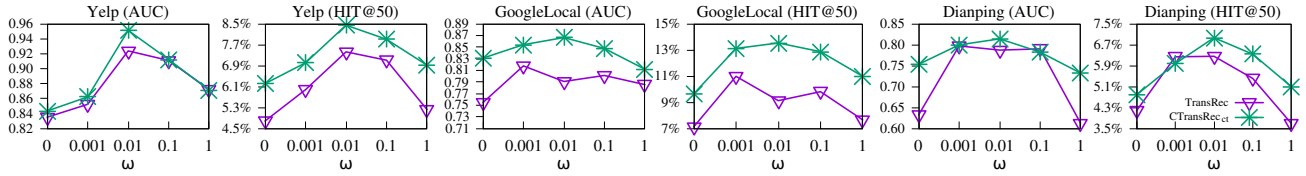


Fig. 8. Performance for different values of ω ($k = 10, \eta = 0.5$)

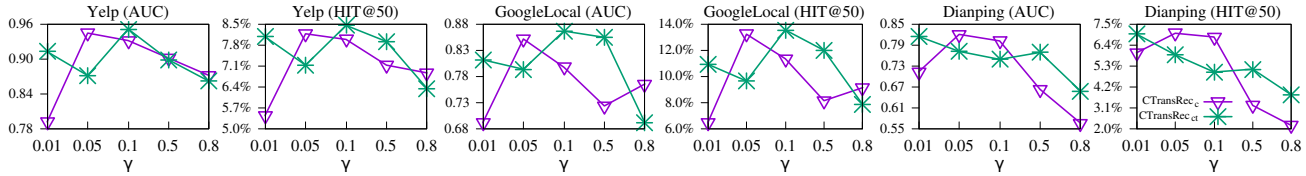


Fig. 9. Performance for different values of γ ($k = 10, \eta = 0.5$)

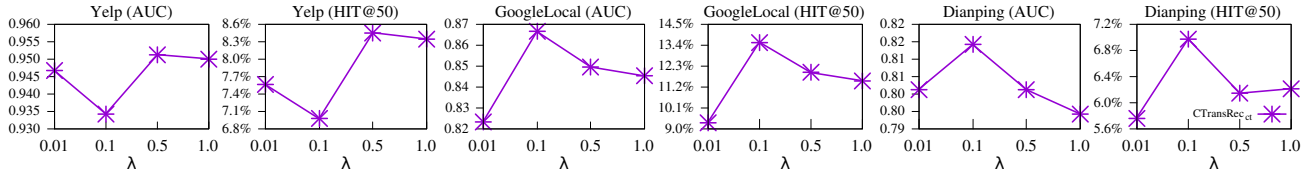


Fig. 10. Performance for different values of λ ($k = 10, \eta = 0.5$)

category based projection already models translation well and the additional temporal information introduces noise; more parameters are updated and the result becomes worse. In practice, we suggest a large γ for CTransRec$_{ct}$ while CTransRec$_c$ should use a smaller γ.

Growth rate λ. λ is related to how the time interval between previous and next items affects the closeness between projected item embeddings. From Fig. 10 we can see that the optimal value of λ can be found easily, since the performance improves as λ increases until λ reaches an optimal value after which the performance drops.

In summary, the hyperparameters of CTransRec are relatively easy to tune and its performance is consistently better than TransRec under different settings and update methods.

4.4 Next-category Prediction (Q3)

By replacing item embeddings $\mathbf{p}$ and weighted category projection vector $\mathbf{s}_i$ in Eq. 4 with category embeddings $\mathbf{c}$ and category-specific projection $\mathbf{o}$ respectively, we can set the probability that user u transits from the category c_i of previous item i to category c_j of next item j as:

$$\begin{gathered}\mathbf{M}_{u,i} = \mathbf{g}_u \mathbf{o}_i^T,\ \mathbf{M}_{u,j} = \mathbf{g}_u \mathbf{o}_j^T \\ Prob(c_j|u,c_i) \propto \beta_{c_j} - e^{\lambda(t_j - t_i)} ||\mathbf{M}_{u,i}\mathbf{c}_i + \mathbf{q}_u - \mathbf{M}_{u,j}\mathbf{c}_j||_2,\end{gathered} \tag{8}$$

where β_{c_j} is the bias for category c_j. CTransRec ranks each possible next category (c_j) according to the value of predictor $Prob(c_j|u,c_i)$; finally, the categories with highest ranks are suggested to the user.

Assume that one test triple is $\langle i, u, ? \rangle$, where the ground truth is item j. i and j belong to categories $\{c_1, c_2\}$ and $\{c_3, c_4\}$, respectively. Then the test triple is further divided into four test triples: $\langle c_1, u, c_3 \rangle$, $\langle c_1, u, c_4 \rangle$, $\langle c_2, u, c_3 \rangle$ and $\langle c_2, u, c_4 \rangle$. To compute AUC and HIT@50, R_{u,c_j} which represents the rank of category c_j among all categories for u is used instead of $R_{u,j}$ in Eq. 7:

$$AUC = \frac{1}{|\mathcal{T}|} \sum_{<c_i, u, c_j, f_{j,i}> \in \mathcal{T}} \frac{1}{|\mathcal{C} \setminus \mathcal{S}^u|} \sum_{c_{j'} \in \mathcal{C} \setminus \mathcal{S}^u} \mathbf{1}(R_{u,c_j} < R_{u,c_{j'}}),$$

$$Hit@50 = \frac{1}{|\mathcal{T}|} \sum_{<c_i, u, c_j, f_{j,i}> \in \mathcal{T}} \mathbf{1}(R_{u,c_j} \leq 50),$$

TABLE 3
Results of next-category prediction ($k = 10, \eta = 0.5$)

Data	Metric	FPMC-LBPR	CTransRec$_{ct}$	Improvement
Yelp	AUC	0.8101	0.8693	7.31%
	Hit@50	12.34%	16.46%	33.39%
GoogleLocal	AUC	0.8193	0.8631	5.35%
	Hit@50	10.14%	15.28%	50.69%
Dianping	AUC	0.7921	0.8542	7.84%
	Hit@50	9.39%	14.86%	58.25%

where $\mathcal{C} \setminus \mathcal{S}^u$ are categories that u has not selected before.

We compare the performance of CTransRec$_{ct}$ with FPMC-LBPR [22] for next-category prediction using SGA and $k = 10, \eta = 0.5$. As we can see in Tab. 3, CTransRec shows considerable improvement for the next-category prediction task compared to FPMC-LBPR.

4.5 Experiments on Unmanned Stores (Q4)

We conducted an experimental study using data from eight Alpha unmanned stores during a nine-month period in 2017 and 2018. The experimental unmanned stores are located at different sides of a central business district and each store is close to the entrance of the nearby metro. The test models are TransRec and CTransRec. Considering the large volume of real data when the system is deployed after the experimental stage, translation based methods are promising candidates as they achieve a good balance between accuracy and running time on sparse data.

The training data was collected during the first eight months and there are 4,123 customers who bought at least one item, 6,311 items (including virtual goods such as top-up points for mobile services, games, transportation, etc.) in 213 categories which were bought at least once and 921,211 purchasing actions in total. In the last month, we used *A/B testing* [34] to compare the recommendation quality of TransRec and CTransRec$_{ct}$ on the recommendation task after 6,000 *last* actions (i.e., the last action before the customer moves to checkout): the system randomly showed the top-1 recommendation result from either TransRec or CTransRec$_{ct}$ on the screen at checkout, i.e., 3,000 out of 6,000 recommendations were made based on TransRec, and the other 3,000 recommendations were from CTransRec$_{ct}$. We simply measured the percentage that the customer took the recommendation as the *success rate*.

TransRec achieved a success rate of 4.12%, while CTransRec$_{ct}$ had a success rate of 5.41%. This result shows that the proposed CTransRec model has a better performance than previous translation based recommendation models in a real scenario. Note that a different experimental task like, what we did in Secs. 4.2, 4.3, 4.4 and what was done in previous work [2], predict the last action, not the item to be recommended at checkout. For such a problem, the success rate can be higher, because in our case the customer might have already picked all items in her shopping list when moving to checkout and buying more items at this stage could be difficult. However, the goal for the test is to measure the ability of the models to increase Alpha's sales and we set the prediction target to the recommended item at checkout that customer may be willing to buy. In addition, we did not provide discount for the recommended item, which is a typical promotion strategy used in many convenience stores, in order to show that the sale increase comes only from the successful prediction.

TABLE 4
Predictors of sequential recommenders

Method	Predictor	Definition
FPMC	$Prob(j\|u,i)$	$\mathbf{m}_u^T \mathbf{n}_j + \mathbf{b}_j^T \mathbf{v}_i$
FPMC-LR		$\mathbf{m}_u^T \mathbf{n}_j + \mathbf{b}_j^T \mathbf{v}_i, j \in N_i$
FPMC-LBP		$\mathbf{m}_u^T \mathbf{n}_j + \mathbf{b}_j^T \mathbf{v}_i + \rho d_{j,i}^{-1}$
PRME		$-\big(\alpha \cdot \|\|\mathbf{m}_u - \mathbf{n}_j\|\|_2^2 + (1-\alpha) \cdot \|\|\mathbf{y}_i - \mathbf{y}_j\|\|_2^2\big)$
HRM		$soft\Big(\mathbf{n}_j^T \cdot agg(\mathbf{m}_u, N_t^u), N_t^u\Big)$
Fossil		$\sum_{j' \in \mathcal{I}^u} \mathbf{m}_{j'}^{s\,T} \mathbf{n}^s{}_j + (\delta + \delta_u)\mathbf{b}_j^T \mathbf{v}_i$
TransRec		$\beta_j - \|\|\mathbf{p}_i + \mathbf{q}_u - \mathbf{p}_j\|\|_2$
CTransRec		$\beta_j - e^{\lambda(t_j - t_i)} \|\|\mathbf{M}_{u,i}\mathbf{p}_i + \mathbf{q}_u - \mathbf{M}_{u,j}\mathbf{p}_j\|\|_2$
FPMC-LBPR	$Prob(c_j\|u,c_i)$	$\mathbf{m}_u^T \mathbf{n}_j^c + \mathbf{b}_j^{cT} \mathbf{v}^c{}_i$

5 Related Work

Since the subject of this paper is sequential recommendation, we divide previous works into two categories: general recommenders and sequential recommenders.

5.1 General Recommenders

Traditional recommender systems do not consider sequential behaviors and they are typically based on collaborative filtering, especially matrix factorization (MF) [34]. MF models user preferences and item properties by factorizing the user-item interaction matrix into two low-dimensional latent matrices. Due to its effectiveness on large-scale data [35], MF has been successfully deployed in practice. The cold-start problem (i.e., data sparsity), where historical data is not available for new users or items, is one of the most challenging issues in recommender systems [36, 37, 38]. One solution to alleviate this problem is to incorporate additional context features (e.g., social network [39, 40], user grouping data [41], relationships in a graph [42], locations of users and items [43] and review text [44, 45]) into MF. However, it is hard to use general recommenders for sequential recommendation tasks directly, since user sequential behaviors should also be modeled [5].

5.2 Sequential Recommenders

Sequential recommender systems typically rely on either Markov Chains (MC) [1] or neural networks (e.g., RNNs and CNNs) [6, 46] to capture sequential patterns and predict the target user's next action based on his/her previous action(s).

MC models sequential behavior by learning a transition graph over items. Tab. 4 summarizes how these models define the predictor (i.e., the probability $Prob(j|u,i)$ that user u transits from the previous item i to the next item j or the probability $Prob(c_j|u,c_i)$ that user u transits from previous item category c_i to next category c_j). Rendle et al. [1] proposed FPMC which subsumes both MF and MC and its predictor consists of the inner product of user and item factors $\mathbf{m}_u$, $\mathbf{n}_j$ from MF and the inner product of the factors of previous item $\mathbf{v}_i$ and next item $\mathbf{b}_j$ from MC. Along this line, several approaches have been proposed for sequential recommendation. FPMC-LR [3] extends FPMC, by using a cube consisting of users, locations and neighborhood locations (i.e., N_i is the neighborhood locations of location

i in Tab. 4) for next-POI recommendation. Having the same motivation, FPMC-LBP [21] added spatial constraints to FPMC for next-POI recommendation. Constraint term $\rho d_{j,i}^{-1}$ in FPMC-LBP indicates the spatial preference of user u to visit a POI j at distance $d_{j,i}$ far away from the previous POI i; ρ is a learned parameter. Then, an intermediate latent pattern layer is used together with the predictor to capture the pattern-level preference. FPMC-LBP also considers the influence of time. Compared to CTransRec which adopts numerical variables and can model the influence of various time intervals, FPMC-LBP utilizes categorical variables for time (e.g., days of the week) and can only model coarse-grained sequential recommendation. FPMC-LBPR [22] extends FPMC with Listwise Bayesian Personalized Ranking for the next-category recommendation. The item factors $\mathbf{n}$, $\mathbf{b}$ and $\mathbf{v}$ in FPMC are replaced by category-level factors $\mathbf{n}^c$, $\mathbf{b}^c$ and $\mathbf{v}^c$, respectively. After the next category is predicted, next target item is chosen among the items belonging to this category. PRME [4] replaces the inner product in FPMC by Euclidean distance and the two components for user-item interaction and item-item interaction are weighted through linear interpolation. Each item is encoded by a low-dimensional vector $\mathbf{y}$ in the metric space. HRM [5] extends FPMC by adopting two-step aggregation operations agg (one step can be either average pooling or maximum pooling) over item factors in recent actions N_t^u and user factors $\mathbf{m}_u$. The inner product of $\mathbf{n}_j$ and the aggregated item representation is passed into a softmax function $soft$ as the predictor. Fossil [17] is a similarity-based sequential recommendation model which uses item-item similarity (factorizing into two item latent matrices $\mathbf{M}^s$ and $\mathbf{N}^s$) instead of user-item interaction in the cube. Like PRME, the predictor of Fossil is also weighted using $\delta + \delta_u$, where δ is a global parameter and δ_u is associated with u. Although it does not exactly belong to the FPMC family, STELLAR [47] also models cubical interactions $\langle$user, location, time$\rangle$ via tensor factorization.

With the recent advances in deep learning, there are many models considering using neural networks in sequential recommendation. DREAM [48] applies RNNs in sequential recommendation and GRU4Rec [6] adopts GRU to model click sequences for non-personalized session-based recommendation. Later, Donkers et al. [49] modified GRU and Quadrana et al. [15] adopted hierarchical RNNs so that personalized sequential recommendation can be modeled. Zhu et al. [50] introduced Attention-GRU into a sequential recommender for ranking brands. To capture both long-term and short-term user preferences, there are several works considering using LSTM. Wu et al. [7] utilized LSTM to predict future behavioral trajectories. Phased LSTM [27] considers incorporating time information in addition to sequential order into LSTM. Zhu et al. [28] argued that Phased LSTM only captures the timestamps of the individual actions, rather than the time intervals between consecutive actions; hence, they proposed Time-LSTM, which also considers the time intervals. Other approaches consider more additional contextual information (e.g., knowledge bases [51] and video [52]) when designing RNN-based techniques. Tuan and Phuong [46] proposed to use 3D CNNs and utilize content features such as item descriptions and item categories. Caser [8] embeds a sequence of recent items as an *image* and then adopts CNNs to learn the features.

As shown in our experiments and in the literature [10, 11], traditional MC based approaches do not offer high-quality recommendations, while neural network based models do not provide fast recommendations. TransRec [2, 13] is a translation based recommender for sequential recommendation. TransRec borrows the ideas behind the TransE KG completion method [18] and models each user as the 'translation' between the previous item bought by her and next item to be chosen by her. Then, a similar objective function as the one used in TransE is injected into FPMC as the predictor. TransRec yields much better results compared to previous approaches and scales to large datasets [13], although it cannot handle complex users like our CTransRec. Similar to TransRec, which adopts the idea of word embedding, Geo-Teaser [53] is a POI recommender inspired by the success of word embedding. Geo-Teaser can capture contextual information in check-in sequences.

6 Conclusion

In this paper, we propose CTransRec, which utilizes auxiliary information (item category and timestamp) in recommender systems to improve the performance of sequential recommendation. CTransRec aims at handling complex users in its translation based model. The improvement of CTransRec comes from not only the additional information used in training but also the fact that CTransRec models complex translations well via category-specific projection and temporal dynamic relaxation. The superiority of CTransRec over existing methods is twofold:

1) CTransRec gives better recommendations for complex users, compared to existing translation based recommendation models.
2) The runtime cost of CTransRec is roughly two times higher compared to the existing translation based method and CTransRec scales well on large data in practice.

In the applications of sequential recommendation, there typically exists textural information (e.g., reviews [45]), spatial information (e.g., trajectory [54]) and social information (e.g., social tags [55]). In the future, we plan to unify the predictor in a more general form where any side information can be incorporated into CTransRec for handling complex users, in order to further improve the quality of sequential recommendation.

References

[1] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*, 2010, pp. 811–820.

[2] R. He, W. Kang, and J. McAuley, "Translation-based recommendation," in *RecSys*, 2017, pp. 161–169.

[3] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *IJCAI*, 2013, pp. 2605–2611.

[4] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new POI recommendation," in *IJCAI*, 2015, pp. 2069–2075.

[5] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning hierarchical representation model for nextbasket recommendation," in *SIGIR*, 2015, pp. 403–412.

[6] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *arXiv Preprint*, vol. abs/1511.06939, 2016. [Online]. Available: https://arxiv.org/abs/1511.06939

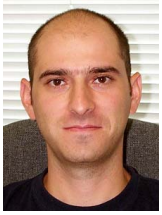
[7] C. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *WSDM*, 2017, pp. 495–503.

[8] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*, 2018, pp. 565–573.

[9] W. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018.
[10] M. Ludewig and D. Jannach, "Evaluation of session-based recommendation algorithms," *User Modeling and User-Adapted Interaction*, vol. 28, no. 4-5, pp. 331–390, 2018.
[11] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 66:1–66:36, 2018.
[12] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, 2017.
[13] R. He, W. Kang, and J. McAuley, "Translation-based recommendation: A scalable method for modeling sequential behavior," in *IJCAI*, 2018, pp. 5264–5268.
[14] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL (1)*, 2015, pp. 687–696.
[15] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *RecSys*, 2017, pp. 130–137.
[16] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "STAMP: short-term attention/memory priority model for session-based recommendation," in *KDD*, 2018, pp. 1831–1839.
[17] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*, 2016, pp. 191–200.
[18] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013, pp. 2787–2795.
[19] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.
[20] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy," in *RecSys*, 2011, pp. 165–172.
[21] J. He, X. Li, L. Liao, D. Song, and W. K. Cheung, "Inferring a personalized next point-of-interest recommendation model with latent behavior patterns," in *AAAI*, 2016, pp. 137–143.
[22] J. He, X. Li, and L. Liao, "Category-aware next point-of-interest recommendation via listwise bayesian personalized ranking," in *IJCAI*, 2017, pp. 1837–1843.
[23] Y. Wang, R. Gemulla, and H. Li, "On multi-relational link prediction with bilinear models," in *AAAI*, 2018, pp. 4227–4234.
[24] Y. Ding and X. Li, "Time weight collaborative filtering," in *CIKM*, 2005, pp. 485–492.
[25] Y. Koren, "Collaborative filtering with temporal dynamics," in *KDD*, 2009, pp. 447–456.
[26] N. N. Liu, M. Zhao, E. W. Xiang, and Q. Yang, "Online evolutionary collaborative filtering," in *RecSys*, 2010, pp. 95–102.
[27] D. Neil, M. Pfeiffer, and S. Liu, "Phased LSTM: accelerating recurrent network training for long or event-based sequences," in *NIPS*, 2016, pp. 3882–3890.
[28] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-lstm," in *IJCAI*, 2017, pp. 3602–3608.
[29] M. Fan, Q. Zhou, E. Chang, and T. F. Zheng, "Transition-based knowledge graph embedding with relational mapping properties," in *PACLIC*, 2014, pp. 328–337.
[30] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
[31] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv Preprint*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701
[32] B. Shi and T. Weninger, "Open-world knowledge graph completion," in *AAAI*, 2018, pp. 1957–1964.
[33] B. Recht, C. Ré, S. J. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *NIPS*, 2011, pp. 693–701.
[34] F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*. Springer, 2015.
[35] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis, "FEXIPRO: fast and exact inner product retrieval in recommender systems," in *SIGMOD Conference*, 2017, pp. 835–850.
[36] B. Wang, M. Ester, J. Bu, Y. Zhu, Z. Guan, and D. Cai, "Which to view: Personalized prioritization for broadcast emails," in *WWW*, 2016, pp. 1181–1190.
[37] B. Wang, M. Ester, Y. Liao, J. Bu, Y. Zhu, Z. Guan, and D. Cai, "The million domain challenge: Broadcast email prioritization by cross-domain recommendation," in *KDD*, 2016, pp. 1895–1904.
[38] Y. Zhu, Z. Guan, S. Tan, H. Liu, D. Cai, and X. He, "Heterogeneous hypergraph embedding for document recommendation," *Neurocomputing*, vol. 216, pp. 150–162, 2016.
[39] H. Li, D. Wu, W. Tang, and N. Mamoulis, "Overlapping community regularization for rating prediction in social recommender systems," in *RecSys*, 2015, pp. 27–34.
[40] H. Li, D. Wu, and N. Mamoulis, "A revisit to social network-based recommender systems," in *SIGIR*, 2014, pp. 1239–1242.
[41] D. Ding, H. Li, Z. Huang, and N. Mamoulis, "Efficient fault-tolerant group recommendation using alpha-beta-core," in *CIKM*, 2017, pp. 2047–2050.
[42] Y. Qian, H. Li, N. Mamoulis, Y. Liu, and D. W. Cheung, "Reverse k-ranks queries on large graphs," in *EDBT*, 2017, pp. 37–48.
[43] Z. Lu, H. Li, N. Mamoulis, and D. W. Cheung, "HBGG: a hierarchical bayesian geographical model for group recommendation," in *SDM*, 2017, pp. 372–380.
[44] C. Wang, M. Niepert, and H. Li, "LRMM: learning to recommend with missing modalities," in *EMNLP*, 2018, pp. 3360–3370.
[45] A. García-Durán, R. Gonzalez, D. Oñoro-Rubio, M. Niepert, and H. Li, "Transrev: Modeling reviews as translations from users to items," *arXiv Preprint*, vol. abs/1801.10095, 2018. [Online]. Available: http://arxiv.org/abs/1801.10095
[46] T. X. Tuan and T. M. Phuong, "3d convolutional networks for session-based recommendation with content features," in *RecSys*, 2017, pp. 138–146.
[47] S. Zhao, T. Zhao, H. Yang, M. R. Lyu, and I. King, "STELLAR: spatial-temporal latent ranking for successive point-of-interest recommendation," in *AAAI*, 2016, pp. 315–322.
[48] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *SIGIR*, 2016, pp. 729–732.
[49] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *RecSys*, 2017, pp. 152–160.
[50] Y. Zhu, J. Zhu, J. Hou, Y. Li, B. Wang, Z. Guan, and D. Cai, "A brand-level ranking system with the customized attention-gru model," in *IJCAI*, 2018, pp. 3947–3953.
[51] J. Huang, W. X. Zhao, H. Dou, J. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *SIGIR*, 2018, pp. 505–514.
[52] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *RecSys*, 2016, pp. 241–248.
[53] S. Zhao, T. Zhao, I. King, and M. R. Lyu, "Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation," in *WWW*, 2017, pp. 153–162.
[54] J. Huang, X. Huangfu, H. Sun, H. Li, P. Zhao, H. Cheng, and Q. Song, "Backward path growth for efficient mobile sequential recommendation," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 1, pp. 46–60, 2015.
[55] N. Hariri, B. Mobasher, and R. D. Burke, "Context-aware music recommendation based on latenttopic sequential patterns," in *RecSys*, 2012, pp. 131–138.

Hui Li is currently an assistant professor in the School of Information Science and Engineering, Xiamen University. His research interests include data mining and data management with applications in recommender systems and knowledge graph. He received his B.Eng. degree in software engineering from Central South University (2012), and his MPhil and PhD degrees in computer science from the University of Hong Kong (2015, 2018).

Ye Liu obtained his Ph.D degree from National University of Singapore (2018). Before that he received the M.Sc. degree from Peking University. He is now a Research Fellow in National University of Singapore. His research interests lie mainly in the areas of urban computing, ubiquitous computing, data mining, machine learning, artificial intelligence and their applications in human activity analysis, social network analysis and knowledge graph.



Nikos Mamoulis received his diploma in computer engineering and informatics in 1995 from the University of Patras, Greece, and his PhD in computer science in 2000 from HKUST. He is currently a faculty member at the University of Ioannina. Before, he was professor at the Department of Computer Science, University of Hong Kong. His research focuses on the management and mining of complex data types.

David S. Rosenblum is Professor of Computer Science and Dean of the School of Computing at the National University of Singapore (NUS). His research interests span many problems in software engineering, distributed systems and ubiquitous computing. He is a Fellow of the ACM and IEEE and a Senior Member of the Singapore Computer Society.