This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

ATM: An Attentive Translation Model for Next-Item Recommendation

Bin Wu, Xiangnan He, Zhongchuan Sun, Liang Chen, and Yangdong Ye

***Abstract*—Predicting what items a user will consume in the next time (*i.e.,* next-item recommendation) is a crucial task for recommender systems. While the factorization method is a popular choice in recommendation, several recent efforts have shown that the inner product does not satisfy the triangle inequality, which may hurt the model's generalization ability. TransRec is a promising method to overcome this issue, which learns a distance metric to predict the strength of user-item interactions. Nevertheless, such method only uses the latest consumed item to model a user's short-term preference, which is insufficient for modeling fidelity. In this paper, we propose a simple yet effective method named ATM, short for *Attentive Translation Model*, to explicitly exploit high-order sequential information for next-item recommendation. Specifically, we construct a user-specific translation vector by accounting for multiple recent items, which encode more information about a user's short-term preference than the latest item. To aggregate multiple items into one representation, we devise a position-aware attention mechanism, learning different weights on items at different orders in a personalized way. Extensive experiments on four real-world datasets show that our method significantly outperforms several state-of-the-art methods.**

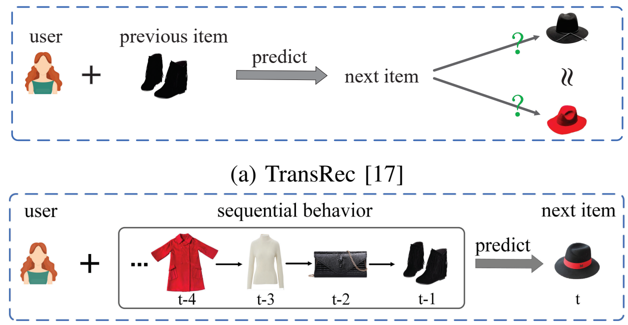
***Index Terms*—Attention, Translation, Next-Item Recommendation, Implicit Feedback.**

I. Introduction

RECOMMENDER systems aim to infer users' preferences on items and assist users in identifying desired information [1], [2]. In the past decade [3], many efforts have been made to develop general recommendation methods, such as neighbor-based methods [4], matrix factorization (MF) [5], [6], [7], and neural networks [8], [9], [10]. These methods focus on modeling user behaviors on items (*e.g.,* purchase/click records) and forgo other affiliated information like time, user profiles, and item attributes. While offering a generic solution for building recommendation service [11], CF methods usually provide suboptimal performance in personalized ranking and can be substantially improved by incorporating these information [12], [13], [14].

User behaviors are sequential by nature. As such, from a practical standpoint, predicting what items a user will consume in the next time (*i.e.,* short-term interest) could be more valuable than predicting user's general interest. This task is known as *next-item recommendation*, which tailors the model design and learning to predict the next item to consume. The *Factorized Personalized Markov Chain* (FPMC) method [15] is a pioneer and prevalent solution for next-item recommendation. It complements MF with the modeling of the interaction between the next item and the latest consumed item, which encodes users' short-term interests (*i.e.,* the first-order Markov assumption). However, a drawback of such factorization-based method lies in the use of the inner product to model the interaction between a user and an item, which does not satisfy the triangle inequality[1] [16] and may incur large ranking loss [2]. The triangle inequality states that for any three objects, the sum of any two pairwise distance should be greater than or equal to the remaining pairwise distance. For instance, items i and j are both similar to item k. The triangle inequality means that item i is also similar to item j. Therefore, these existing approaches based on inner product operator can only preserve the first-order proximity (*i.e.,* both i and j are similar to k), but fail to capture the second-order proximity (*i.e.,* i and j are also similar); this drawback leads to suboptimal performance, as described in [16].

A recent trend in recommendation research is to explore more expressive interaction function to model the user-item relation. For example, He *et al.* [2] formulates the neural collaborative filtering (NCF) framework, augmenting inner



(b) Our method (ATM: Attentive Translation Model)

Fig. 1: An example to illustrate the recommendation mechanism of TransRec and our method. (a) TransRec applies first-order Markov modeling which only considers the previous one item to predict the next. (b) Our method enhances TransRec with high-order Markov modeling, using multiple previous items to predict the next.

This work is supported by the National Key Research and Development Program of China under Grant No. 2018YFB1201403 and the National Natural Science Foundation of China under Grant No. 61772475. (*Corresponding author: Yangdong Ye*)

B. Wu, Z. Sun and Y. Ye are with the School of Information Engineering, Zhengzhou University, Zhengzhou, 450001, China (e-mail: wubin@gs.zzu.edu.cn, iezcsun@gs.zzu.edu.cn, yeyd@zzu.edu.cn).

X. He is with School of Information Science and Technology, University of Science and Technology of China, Hefei, 230027, China (e-mail: xiangnanhe@gmail.com).

L. Chen is with the School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, 510007, China (email: chenliang6@mail.sysu.edu.cn).

[1] (a, c)'s distance is bounded by the sum of distances between (a, b) and (b, c): $d(a, c) \leq d(a, b) + d(b, c)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

product with deep neural networks in interaction learning (*i.e.,* the NeuMF model); later the authors [18] extend the NCF framework by using outer product and convolution neural network to learn the interaction function (*i.e.,* the ConvNCF model); and He *et al.* [17] applies a distance metric that satisfies the triangle inequality to model user-item interaction (*i.e.,* the TransRec model). While NeuMF and ConvNCF are general recommendation methods that do not capture the sequential effect, TransRec is the most relevant work that is designed for next-item recommendation (*a.k.a.,* sequential recommendation). However, TransRec only uses the latest consumed item as short-term user interest (*i.e.,* the first-order Markov assumption as in FPMC), which we believe is insufficient and may lead to suboptimal performance.

Figure 1 shows an example to illustrate the limitation of first-order Markov modeling and the usefulness of high-order Markov modeling for next-item recommendation. The subfigure (a) shows the paradigm of TransRec which only uses the previous item to predict the next item, making it difficult to determine which hat better meets the user's current need. In contrast, the subfigure (b) shows our proposal which accounts for multiple previous items, making it possible to distinguish the user's current need better. Here the hat with black and red as the main color is a desired choice, considering the user's purchases of a red coat and a pair of black shoes in time $t-4$ and $t-1$.

In this paper, we focus on exploiting high-order sequential information for next-item recommendation. We propose a new method named ATM (short for *Attentive Translation Model*), which implements high-order Markov modeling under the TransRec framework. Specifically, when constructing the vector to "translate" a user to the next item, we consider multiple recent items rather than the most recent one. The key technical challenge here is how to aggregate the items from different orders to form a representative translation vector, which is critical to the model's effectiveness. Technically speaking, standard aggregation operations such as max pooling and average pooling [19] can be directly applied. However, they do not cater to the characteristics of personalized high-order Markov modeling thus are less suitable for next-item recommendation. To this end, we adopt a personalized attention strategy, which can learn different weights on items at different orders and for different users.

To summarize, our main contributions are threefold:

- We contribute a simple yet effective solution ATM to integrate personalized high-order Markov modeling into the translation-based recommendation framework.
- We develop a personalized attention mechanism, which is adaptive and position-aware, to capture the varying importance of information at different orders.
- We conduct extensive experiments on a variety of real-world datasets, demonstrating quantitatively that ATM significantly outperforms several state-of-the-art methods and qualitatively that it is capable of making meaningful recommendations.

The remainder of this paper is organized as follows. Section II briefly reviews preliminaries on general recommendation, sequential recommendation, attention mechanism and knowledge graph. Section III provides the formulation of the problem, elaborates our proposed method, and describes how to optimize ATM. In Section IV, extensive experiments are performed to evaluate the effectiveness of our proposed method for next-item recommendation. Lastly, we conclude this paper and give future work in Section V.

II. Related Work

In this section, we review general recommendation that are closely related to ours, followed by a summarization of studies on sequential recommendation and attention mechanism. Lastly we introduce knowledge graph technologies, especially translation-based recommendation.

A. General Recommendation

Traditional item recommendation usually relies on collaborative filtering [20], [21] to learn from implicit feedback such as purchases, clicks, and thumbs-up. Many of these approaches use matrix factorization techniques [22], [23], [24], [25], which seek to learn user and item embedding vectors and use the inner product to predict the strength of user-item interaction. Several existing works [26], [27] have shown that inner product does not satisfy the condition of the triangle inequality, which may limit generalization of the model [16]. Recently, a trend in recommendation research is to seek more expressive interaction function to model the user-item relation. For example, Hsieh *et al.* [16] formulated collaborative metric learning method, replacing the inner product by applying an Euclidean metric; and He *et al.* [18] proposed to use an outer product operation and Convolution Neural Network (CNN) to learn the interaction function. Despite great promise, these methods neglected the influence of sequential dynamics, which may be unsuitable for next-item recommendation [28], [29].

B. Sequential Recommendation

In recent years, the importance of sequential patterns in recommender systems has been gradually recognized by researchers [15], [30]. The early pioneer work by [15] proposed a Factorized Personalized Markov Chains (FPMC) method for next-item recommendation. The work combined the power of MF at modeling general tastes and the strength of the first-order Markov Chain (MC) at modeling sequential patterns. Afterwards, Personalized Ranking Metric Embedding (PRME) method [31] replaces the inner product operators in FPMC with Euclidean distance, where the condition of the triangle inequality plays a vital role in helping the method to generalize well [16]. As illustrated the limitation of first-order Markov modeling in Fig. 1, there are several methods adopting high-order Markov chains that consider multiple recent items. Specifically, He *et al.* [32] integrated similarity-based models with high-order Markov chains smoothly to predict personalized sequential behavior; this method learns adaptive weights for different orders and different users. Nevertheless, we argue that it is unreasonable to assign a same weight for different items, which occur in the same order at each time step. ConvolutionAl Sequence Embedding

Recommendation (Caser) [33], a Convolutional Neural Network (CNN) based method, regards the embedding matrix of high-order Markov chains as an "image" and adopts horizontal and vertical convolutional operations to capture sequential patterns. Recently, Yan *et al.* [34] extended the Caser method by leveraging pairwise encoding module and 2D convolutional filters to capture high-level sequential patterns. Other than the above MC/CNN-based methods, another branch of work adopts Recurrent Neural Network (RNN) to capture sequential patterns. For instance, Yu *et al.* [19] proposed a dynamic recurrent basket model based on RNN for next-basket recommendation; and Hidasi *et al.* [35], [36] proposed to use Gated Recurrent Unit (GRU) to model sequential behavior for session-based recommendation. More recently, Recurrent Convolutional Neural Network (RCNN) [37] leverages the convolutional operation of CNN to extract short-term sequential patterns, while adopts the recurrent operation of RNN to capture long-term dependencies. Since these RNN-based methods take the state from the last step and current action as their input, these dependencies make RNN-based methods less efficient (*i.e.,* higher model complexity). Moreover, the recent study [38] demonstrates that these sophisticated RNN/CNN-based methods underperform the simpler model FPMC by a large margin. It may be because these complex models require large amounts of data to capture long-term patterns, *i.e.,* easily overfitting in high-sparsity settings.

C. Attention Mechanism

Inspired by the psychological cognition scheme, attention mechanism has shown high performance in many tasks, such as image/video captioning [39] and machine translation [40]. Its key idea is to learn to assign adaptive weights for a set of features, *i.e.,* higher attentive weights demonstrate that the corresponding features are more informative. In recent years, due to its strong interpretability, the attention mechanism has been introduced into recommender systems [38], [41], [42], [43]. For instance, in the NAIS model for item-based collaborative filtering, He *et al.* [41] applied an attention network to distinguish which historical items in a user's profile are more important for a prediction. In the ACF model for multimedia recommendation, Chen *et al.* [42] employed a component-level attention module to select representative features for multimedia items, and an item-level attention module to choose informative items to infer the underlying users' preferences. In the NARM model for session-based recommendation, Li *et al.* [43] explored a hybrid encoder with an attention mechanism to capture user's main purpose in the current session. Recently, Yu *et al.* [38] introduced a Multi-order Attentive Ranking model (MARank) for next-item recommendation, which unifies both individual- and union-level sequential patterns for modeling user's short-term preference. MARank and all previous works usually adopt standard attention mechanism to capture user's varying attentions on different items. Nevertheless, standard attention mechanism is unaware of the positions (*i.e.,* orders) of the recent items. In fact, temporal order is very important for sequential recommendation. Comparatively, we devise a personalized attention mechanism for translation-based method, which is an adaptive and position-aware aggregation strategy.

D. Knowledge Graph

Recent years have witnessed rapid growth in knowledge graph (KG) construction and application [44]. A large number of KGs have been created, including Freebase, YAGO, NELL, DBpedia *et al.*, and successfully applied to many challenging tasks, from relation extraction [45] and question answering [46] to link prediction [47] and entity classification [48]. A typical KG is a multi-relational graph consisted of entities and relations. Each edge is represented as a triple of the form <*head entity, relation, tail entity*>, showing that two entities are connected by a specific relation, *e.g.,* <*Tom, graduated_from, NUS*>. Among the various KG techniques, TransE [49] is a prominent and representative model, which embeds entities and relationships of multi-relational data in a transition space that satisfies $\overrightarrow{\textit{head entity}} + \overrightarrow{\textit{relation}} \approx \overrightarrow{\textit{tail entity}}$; readers can refer to [44] for a detailed survey.

Notably, due to their scalability and superior performance over traditional factorization-based methods [50], translation-based methods have been adopted for recommender systems [17], [51], [52], [53], [54], [55]. For instance, Park *et al.* [51] integrated neighborhood information and translational metric learning to model the intensity and heterogeneity of user-item relationships. LRML [52], which is an extension of Collaborative Metric Learning (CML) [16], adopts an augmented memory module and learns to attend over memory blocks to construct relational embeddings. For fashion recommendation, Yang *et al.* [53] embedded items into a transition space, where category-specific complementary relations is modeled by a translation embedding to model the transition between items. The work that is most relevant to our work is [17], which introduces a translation-based method (TransRec) for next-item recommendation; this method embedded all items into a transition space, and then translated the previous item towards the next item by a translation vector. Our work is distinguished from the above methods in that we adopt high-order Markov chains to construct translation embedding, and extensive experiments have verified our assumption is realistic and reasonable.

III. Proposed Method

A. Problem Formulation

In this paper, we focus on solving the next-item recommendation task which is formulated as follows. Let $\mathcal{U}$ and $\mathcal{I}$ denote the set of users and items, respectively. For each user u, a sequence of actions $\mathcal{S}^u$ is known: $\mathcal{S}^u = \{\mathcal{S}_1^u, \mathcal{S}_2^u, \cdots, \mathcal{S}_{|\mathcal{S}^u|}^u\}$ with $\mathcal{S}_t^u \in \mathcal{I}$. The action history of all users is $\mathcal{A} = \{\mathcal{S}^1, \cdots, \mathcal{S}^{|\mathcal{U}|}\}$. Given each user u and her/his behavior sequence $\mathcal{S}^u$, the goal of next-item recommendation is to derive a total ranking $>_{u,t}$ over all the un-observed items at time t and to recommend the top-N items for the user u. The key notations and their explanations used in this paper are summarized in Table I.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

TABLE I: Notations

Notation	Explanation
$\mathcal{U}$,$\mathcal{I}$	user set, item set
u,j,t	user $u \in \mathcal{U}$, item $j \in \mathcal{I}$, a specific time step
$\mathcal{S}^u$	action sequence of user u: $\{\mathcal{S}^u_1, \cdots, \mathcal{S}^u_{\lvert\mathcal{S}^u\rvert}\}$
$\mathcal{S}^u_t$	the item that user u interacted with at time step t
γ^U_u	embedding vector associated with user u
γ^I_j	embedding vector associated with item j
$\boldsymbol{a}$	global translation vector
$T_u(t)$	translation vector associated with user u at time step t
b_j	bias term associated with item j
$d(x, y)$	distance between x and y

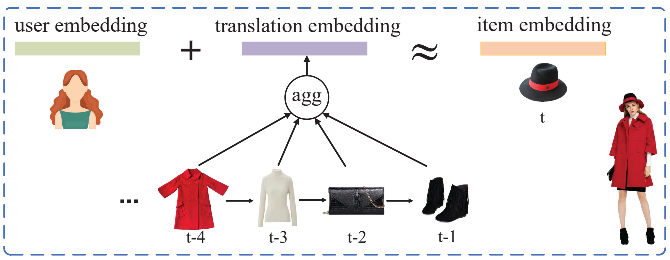


Fig. 2: Illustration of high-order Markov chains for a user-specific translation embedding. The agg symbol denotes aggregation operation. Best viewed in color.

B. Personalized Attentive Translation Model

In recommendation scenarios, users and items can be treated as 'entities' as well. Inspired by the translation embedding techniques [49], we represent each user/item as an embedding vector in a transition space, and treat every user-item interaction as one specific type of 'relation'. Let $\gamma^U_u \in \mathbb{R}^{\mathrm{K}}$ and $\gamma^I_{\mathcal{S}^u_t} \in \mathbb{R}^{\mathrm{K}}$ be the embedding vector of user u and item $\mathcal{S}^u_t$, respectively, and K be the *embedding size*, *i.e.,* the dimension of the embedding vector. To model the sequential behaviors, we utilize a user-specific translation vector $T_u(t)$ to model the user's short-term dynamics. In particular, if user u at time step t is translated towards the next item $\mathcal{S}^u_t$, the following relation should hold

$$\gamma^U_u + T_u(t) \approx \gamma^I_{\mathcal{S}^u_t}. \tag{1}$$

In other words, $\gamma^I_{\mathcal{S}^u_t}$ should be the nearest neighbor of $\gamma^U_u + T_u(t)$ in the transition space. Given the recent L items that user u has interacted with $\{\mathcal{S}^u_{t-1}, \cdots, \mathcal{S}^u_{t-\mathrm{L}}\}$, $T_u(t)$ can be obtained from an Lth order Markov chains which is defined as follows:

$$T_u(t) = \boldsymbol{a} + f(\gamma^I_{\mathcal{S}^u_{t-1}}, \cdots, \gamma^I_{\mathcal{S}^u_{t-\mathrm{L}}}), \tag{2}$$

where $f(\cdot)$ denotes the aggregation operation and $\boldsymbol{a}$ denotes the user-irrelevant translation vector to capture global transition bias. Figure 2 illustrates a general framework for aggregating high-order Markov chains.

One advantage of our method is that we can integrate different aggregation operators to construct the user-specific translation vector $T_u(t)$. To aggregate the high-order Markov chains into a single embedding vector, max pooling and average pooling are probably the most widely used [19]. Unfortunately, these two pooling operations lack the flexibility to model different orders of the Markov chain with different weights. Intuitively, recent actions should be more correlated with the next action. To achieve this goal, we can allow the recent L items unequally to the translation vector $T_u(t)$ by employing a weighted sum:

$$f(\gamma^I_{\mathcal{S}^u_{t-1}}, \cdots, \gamma^I_{\mathcal{S}^u_{t-\mathrm{L}}}) = \sum\nolimits_{l=1}^{\mathrm{L}} \alpha_l \cdot \gamma^I_{\mathcal{S}^u_{t-l}}, \tag{3}$$

where α_l is a trainable parameter that denotes the individualized weight of the order l in contributing to the translation vector $T_u(t)$. While this schema seems to be capable of differentiating the importance of different orders, it ignores the fact that different users may differ in sequential behaviors. Furthermore, each user employs the same weight vector to capture their short-term interests, which may limit the model's representation ability. To alleviate the limitation, an intuitive solution is to assign a personalized weight vector $\beta_u = \{\beta_{u,1}, \beta_{u,2}, \ldots, \beta_{u,\mathrm{L}}\}$ for the target user u. The revised schema is designed as follows:

$$f(\gamma^I_{\mathcal{S}^u_{t-1}}, \cdots, \gamma^I_{\mathcal{S}^u_{t-\mathrm{L}}}) = \sum\nolimits_{l=1}^{\mathrm{L}} (\alpha_l + \beta_{u,l}) \cdot \gamma^I_{\mathcal{S}^u_{t-l}}. \tag{4}$$

While the above solution sounds to be reasonable, it comes across a main obstacle. The size of β is very huge (*i.e.,* $|\mathcal{U}| \times \mathrm{L}$), which is lack of flexibility in practice. Particularly, we argue that it is unreasonable to assign a global weight for different items, which occur in the same order at each time step. Continuing an earlier example, suppose that a user is shopping at Amazon. At time step t, the user may put more attention on the red coat ($t-4$) and a pair of black shoes ($t-1$). At other time steps, the above aggregation operation will still assign higher weights on the 1^{th} order and 4^{th} order items. From the perspective of user representation learning, this strategy is not capable of capturing user's varying attentions on different items. Inspired by the recent success of attention mechanism in many tasks, such as machine translation [40], multimedia recommendation [42] and group recommendation [56], we design a personalized attention mechanism, which is an adaptive and position-aware aggregation operation, to capture the importance of each item in the short-term interests of a given user. Formally, the attention network is defined as:

$$\boldsymbol{h}(u,t,l) = \phi(\gamma^I_{\mathcal{S}^u_{t-l}} \boldsymbol{W}_1 + \boldsymbol{P}_l \boldsymbol{W}_2 + \boldsymbol{b}), \tag{5}$$

$$\alpha(u,t,l) = \frac{exp(\gamma^U_u \boldsymbol{h}^\top(u,t,l))}{\sum_{g=1}^{\mathrm{L}} exp(\gamma^U_u \boldsymbol{h}^\top(u,t,g))}, \tag{6}$$

where $\boldsymbol{W}_1 \in \mathbb{R}^{\mathrm{K}\times\mathrm{K}}$, $\boldsymbol{W}_2 \in \mathbb{R}^{\mathrm{K}\times\mathrm{K}}$ and $\boldsymbol{b} \in \mathbb{R}^{\mathrm{K}}$ are the trainable parameters. As we can see, the size of these parameters (*i.e.,* $2\mathrm{K}^2 + (\mathrm{L}+1)\mathrm{K}, \mathrm{K} \ll |\mathcal{U}|$) is much smaller than that of β. Unlike standard attention mechanism that is unaware of the positions of recent items [38], [42], we inject a learnable position embedding matrix $\boldsymbol{P} \in \mathbb{R}^{\mathrm{L}\times\mathrm{K}}$ to model the effect of different orders. $\phi(\cdot)$ is the activation function and we adopt *tanh* to enhance nonlinear capability. To reduce the size of the model parameters, we use the embedding γ^U_u of user u as the context vector and get the adaptive weight $\alpha(u,t,l)$ as the normalized similarity between $\boldsymbol{h}(u,t,l)$ and γ^U_u with a softmax function. Figure 3 illustrates the architecture of ATM,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

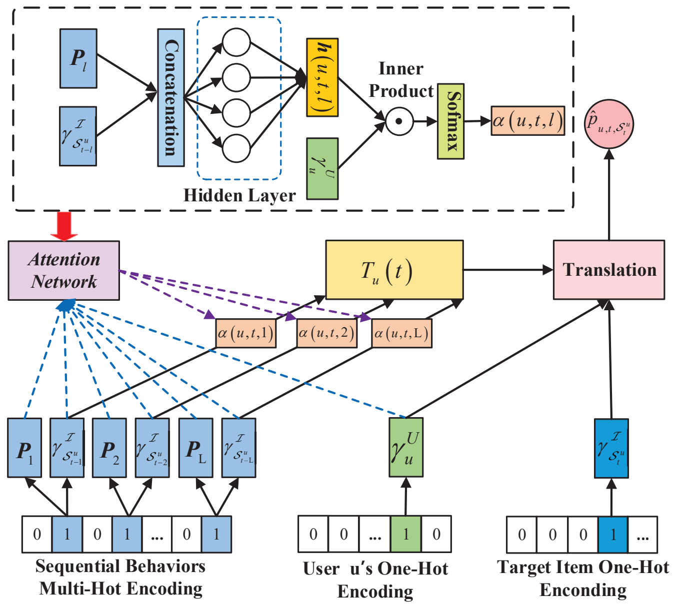


Fig. 3: An illustration of ATM model architecture. Best viewed in color.

which introduces the position-aware attention mechanism to differentiate the varying contributions of the user u's recently interacted L^{th} order items for the final prediction. As a result, we can compute user's dynamic preference as a sum of the item embeddings weighted by the attention scores as follows:

$$f(\gamma^I_{\mathcal{S}^u_{t-1}}, \cdots, \gamma^I_{\mathcal{S}^u_{t-\mathrm{L}}}) = \sum\nolimits_{l=1}^{\mathrm{L}} \alpha(u,t,l) \cdot \gamma^I_{\mathcal{S}^u_{t-l}}. \quad (7)$$

It is worth noting that ATM with the designed attention mechanism is capable of learning different weights on items at different orders and for different users. Finally, for user u, the probability of item $\mathcal{S}^u_t$ being the next item (at time step t) with an L^{th} order Markov chains is predicted by:

$$\begin{aligned} &Prob(\mathcal{S}^u_t \mid u, \mathcal{S}^u_{t-1}, \cdots, \mathcal{S}^u_{t-\mathrm{L}}) \\ &\propto \mu_{\mathcal{S}^u_t} - d(\gamma^U_u + T_u(t), \gamma^I_{\mathcal{S}^u_t}), \end{aligned} \quad (8)$$

where $d(\cdot)$ denotes the $\mathcal{L}_2$ distance. In Eq. (8), we add a single bias term μ to capture the overall item popularity.

C. Optimization Criterion

Given a user u and the previous action sequence $\{\mathcal{S}^u_1, ..., \mathcal{S}^u_{t-1}\}$, $\mathcal{S}^u_t >_{u,t} j$ denotes that item $\mathcal{S}^u_t$ is ranked higher than item j for user u. Here it is a natural choice to optimize such ranking between $\mathcal{S}^u_t$ and j by Sequential Bayesian Personalized Ranking (S-BPR) [15]. Assuming independence of all users, the model parameters of ATM can be inferred by optimizing the following maximum a posterior:

$$\arg\max_{\Theta} \prod_{u\in\mathcal{U}} \prod_{t=\mathrm{L}+1}^{|\mathcal{S}^u|} \prod_{j\neq\mathcal{S}^u_t} Prob(\mathcal{S}^u_t >_{u,t} j|\Theta)\ Prob(\Theta), \quad (9)$$

where $\Theta = \{\boldsymbol{a}, \boldsymbol{b}, \mu, \gamma^U_{u\in\mathcal{U}}, \gamma^I_{j\in\mathcal{I}}, \boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{P}\}$ is the set of our model parameters. By employing a sigmod function $\sigma(z) = \frac{1}{1+e^{-z}}$, the ranking probability can be rewritten as the following:

$$\begin{aligned} Prob(\mathcal{S}^u_t >_{u,t} j|\Theta) &= Prob(\widehat{p}_{u,t,\mathcal{S}^u_t} - \widehat{p}_{u,t,j} > 0|\Theta) \\ &= \sigma(\widehat{p}_{u,t,\mathcal{S}^u_t} - \widehat{p}_{u,t,j}), \end{aligned} \quad (10)$$

where $\widehat{p}_{u,t,\mathcal{S}^u_t}$ is a shorthand for the prediction in Eq. (8). Same as FPMC, the prior distributions over parameters are assumed to be Gaussian. Hence, we have the final objective function of the proposed model, where λ is a regularization hyper-parameter.

$$\begin{aligned} &\arg\max_{\Theta} \ln \prod_{u\in\mathcal{U}} \prod_{t=\mathrm{L}+1}^{|\mathcal{S}^u|} \prod_{j\neq\mathcal{S}^u_t} Prob(\mathcal{S}^u_t >_{u,t} j|\Theta)\ Prob(\Theta) \\ &= \arg\max_{\Theta} \sum_{u\in\mathcal{U}} \sum_{t=\mathrm{L}+1}^{|\mathcal{S}^u|} \sum_{j\neq\mathcal{S}^u_t} \ln\sigma(\widehat{p}_{u,t,\mathcal{S}^u_t} - \widehat{p}_{u,t,j}) - \lambda\|\Theta\|^2. \end{aligned} \quad (11)$$

Due to the huge number of $(u, t, \mathcal{S}^u_t, j)$ quadruples, directly optimizing the objective function in Eq. (11) is time consuming. Instead, following the approach of S-BPR, we independently draw the training quadruples by bootstrapping and apply Stochastic Gradient Descent (SGD) to update the model parameters. According to SGD, the complete algorithm is summarized in Algorithm 1.

Space Complexity. As shown in Eq. (9), the model parameters are composed of two parts: $\Theta_1 = \{\boldsymbol{a}, \mu, \gamma^U_{u\in\mathcal{U}}, \gamma^I_{j\in\mathcal{I}}\}$ and $\Theta_2 = \{\boldsymbol{b}, \boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{P}\}$. The first part of our model parameters is the same as TransRec and grows linearly with users and items. For parameters Θ_2 (*i.e.,* $2\mathrm{K}^2 + (\mathrm{L}+1)\mathrm{K}$), they are shared among all items and users, with the dimensionality of each variable is far less than the number of items and users. As K and L are usually very small (*e.g.,* K<100, L<5), this additional storage cost is practically negligible. Therefore, the space complexity of our proposed method is comparable with TransRec.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

Algorithm 1: The optimization for ATM.

Input: sequential data $\mathcal{A}$, learning rate ϵ, regularization hyper-parameter λ;
Output: global transition bias $\boldsymbol{a}$, item bias μ, embedding vectors γ^U,γ^I, position embedding matrix $\boldsymbol{P}$, and parameters in attention network $\boldsymbol{b}, \boldsymbol{W}_1, \boldsymbol{W}_2$;

```
Initialize a, μ, γ^U, γ^I, P with Gaussian distribution and b, W_1, W_2 with xavier;
repeat
    draw (u, t, S_t^u) from A;
    draw j from I\S^u;
    for l ← 1 to L do
        Compute h(u, t, l) according to Eq. (5);
    for l ← 1 to L do
        Compute α(u, t, l) according to Eq. (6);
    Compute T_u(t) according to Eq. (2) and (7);
    γ̃_u^U ← γ_u^U + T_u(t)
    Compute p̂_{u,t,S_t^u} and p̂_{u,t,j} according to Eq. (8);
    R̂(u, t, S_t^u, j) = p̂_{u,t,S_t^u} − p̂_{u,t,j};
    for each parameter θ in Θ do
        θ ← θ + ε · ( σ(R̂(u, t, S_t^u, j)) ∂R̂(u,t,S_t^u,j)/∂θ − λθ );
until convergence;
return Θ = {a, b, μ, γ^U_{u∈U}, γ^I_{j∈I}, W_1, W_2, P}.
```

Time Complexity. We analyze the time complexity of the predictive model of ATM, *i.e.,* Eq. (8). This reflects the time overhead of ATM in testing, and the training time of ATM should be proportional to the time overhead of testing. The time complexity of obtaining a prediction $\hat{p}_{u,t,\mathcal{S}_t^u}$ with TransRec (*cf.* Eq. (1)) is O(K), where K denotes the embedding size. Compared to TransRec, the additional time cost of obtaining a prediction score with ATM comes from position-aware attention network. Since the denominator of softmax function (*i.e.,* Eq. (6)) needs to traverse over all items in $\{\mathcal{S}_{t-1}^u, \cdots, \mathcal{S}_{t-\mathrm{L}}^u\}$, the time complexity of obtaining an $\alpha(u,t,l)$ is $O(\mathrm{K}^2\mathrm{L})$. Considering the $\alpha(u,t,l)$ term is shared across the computation of $\hat{p}_{u,t,\mathcal{S}_t^u}$ and $\hat{p}_{u,t,j}$, we only need to compute it once and cache it. As such, the computation of $\hat{p}_{u,t,\mathcal{S}_t^u} - \hat{p}_{u,t,j}$ takes additional time cost $O(\mathrm{K}^2\mathrm{L})$. In fact, as shown in our experiments, ATM could obtain the best performance when L = 3 or 4. Also, the embedding size is limited with $\mathrm{K} \ll \min(|\mathcal{U}|,|\mathcal{I}|)$. Thus, the additional time cost is acceptable and our method could be scaled up to large-scale datasets.

IV. Experiments

In this section, our experiments are intended to answer the following research questions:

RQ1 How does ATM perform when compared with several state-of-the-art competitors?

RQ2 How do different aggregation strategies affect the performance of our ATM method?

RQ3 How do different orders of Markov chains affect the performance of our ATM method?

RQ4 Is the designed attention mechanism capable of learning meaningful patterns?

A. Experimental Settings

TABLE II: Statistics of the evaluation datasets.

Dataset	User#	Item#	Action#	Sparsity
Cellphone	7,622	36,121	110,539	99.95%
Tool	7,945	42,614	131,008	99.96%
Clothing	25,160	156,091	379,749	99.99%
Epinions	1,444	17,556	24,531	99.90%

Datasets. To evaluate the performance of our method for next-item recommendation, we experimented on four publicly accessible datasets.

• **Amazon.**[2] The first group of datasets is from *Amazon.com* and span May 1996 to July 2014, containing a large number of consumptions of users in different categories. In particular, we take a series of broad categories including ‘Cell Phones and Accessories’, ‘Tools and Home Improvement’, and ‘Clothing, Shoes and Jewelry’, which are named as *Cellphone, Tool, and Clothing* for short.

• **Epinions.**[3] It is originally from *Epinions.com*, containing a large number of online consumer reviews from January 2001 to November 2013. This dataset was collected by [57] and is widely used for evaluating next item prediction methods.

In our experiments, we transform observed ratings into binary implicit feedback as ground truth, such that our goal is to rank items that a user would be likely to purchase/click. For each of the above datasets, we eliminate users that have less than 10 associated actions. As a result, Table II shows the characteristics of the final datasets.

Evaluation Protocol. Similar to previous works [18], [58], we adopt the leave-one-out evaluation to evaluate the performance of next-item recommendation. Specially, for each user, we sort his/her interactions in chronological order, holding out the second-to-last interaction as the validation set and the last interaction as the test set. The remainder is used for training. Unlike previous works [2], [56], [59] that randomly select 100 items that have not been consumed by the given user, we choose all items non-interacted by each user as the candidate items for next-item recommendation. As there are many un-observed items for each user, our evaluation protocol should handle much more challenging cases for next-item recommendation. We contend that our evaluation protocol is more suitable for providing a fair comparison for the next-item recommendation task, as it essentially avoids sampling bias during evaluation. In our experiments, we employ two ranking-oriented metrics, *Hit Ratio* (HR@N) and *Normalized Discounted Cumulative Gain* (NDCG@N), which have been widely used in the literature [59], [60], [61]. Intuitively, HR@N measures whether the testing item is in the top-N list, while NDCG@N accounts for the position of the hit by assigning higher scores to hits at top ranks. Notably, as we only have one test instance for each user, HR@N is proportional

[2]https://cseweb.ucsd.edu/~jmcauley/datasets.html
[3]http://jmcauley.ucsd.edu/data/epinions/

to Precision@N, and is equivalent to Recall@N. Formally, the definitions of HR@N and NDCG@N are shown as follows:

$$\mathrm{HR@N} = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\sum_{a=1}^{\mathrm{N}}\delta_u(a),$$
$$\mathrm{NDCG@N} = \frac{1}{|\mathcal{U}|}\sum_{u=1}^{|\mathcal{U}|}\frac{\sum_{a=1}^{\mathrm{N}}\frac{2^{\delta_u(a)}-1}{\log_2(a+1)}}{1/\log_2(a+1)}, \tag{12}$$

where $|\mathcal{U}|$ is the number of users. $\delta_u(a)$ is a binary indicator function that equals to 1 if the item at rank a is purchased in the test data, otherwise equals to 0.

Baselines. In our experiments, we compare ATM with several state-of-the-arts, including general recommendation methods, MC-based methods and RNN/CNN-based methods.

General Recommendation Methods:

- **BPRMF:** This baseline is a state-of-the-art pairwise method, which is proposed in [62]. It models the pairwise ranking for each pair of the unobserved and observed products, and employs SGD with bootstrap sampling for optimization.
- **NeuMF:** This is a state-of-the-art method with binary cross-entropy loss, which is proposed in [2]. This model seamlessly combines the linearity of MF and non-linearity of DNNs for modeling user-item interactions. It ignores the sequential patterns in the system.

MC-based Methods:

- **FPMC:** This baseline is described in [15], which is a state-of-the-art method for next-item recommendation. It unifies the strength of matrix factorization at modeling users' general preferences and the power of first-order Markov chain at capturing sequential patterns.
- **PRME:** This method is described in [31], which replaces the inner products in FPMC with Euclidean distances. It embeds users and items into two Euclidean spaces to model personalized Markov behavior.
- **Fossil:** It is proposed by He *et al.* [32], which combines factored similarity model (*i.e.,* FISM [63]) and high-order Markov chains by adopting a weight sum aggregation over multiple recent item embeddings.
- **TransRec:** This method is described in [17], which embeds each item in a shared Euclidean space and learns personalized translation vectors through this space for each user.
- **MARank [38]:** An improved version of FPMC, which unifies both individual- and union-level item interaction into preference inference model from multiple views, and shows significant performance gains on next-item recommendation.

RNN/CNN-based Methods:

- **NARM [43]:** An RNN-based state-of-the-art recommender, which adopts attention mechanism to the user's main purpose from the hidden states and combines it with sequential behaviors as a unified session representation to generate recommendations.
- **Caser [33]:** A CNN-based state-of-the-art recommender, which captures high-order Markov chains by employing horizontal and vertical convolutional operations on the embedding matrix of the L most recent items.

TABLE III: Properties of methods being compared.

Method	Sequentially -aware	Triangle -preserving	High-order Markov chains	Position -aware
BPRMF	✘	✘	✘	✘
NeuMF	✘	✘	✘	✘
NARM	✔	✘	✔	✔
Caser	✔	✘	✔	✘
FPMC	✔	✘	✘	✘
PRME	✔	✔	✘	✘
Fossil	✔	✘	✔	✘
TransRec	✔	✔	✘	✘
MARank	✔	✘	✔	✘
ATM	✔	✔	✔	✔

As other general recommender systems and sequential methods (*e.g.,* **eALS** [58], **ACF** [42], **GRU4Rec** [35], **GRU4Rec+** [36]) have been outperformed on similar datasets by our baselines, we omit comparison against them. We also dont's include temporal models, such as **TimeSVD++** [64] and **RRN** [65], which differ in setting from what we consider. In order to provide a clear understanding of the above methods, we provide a summary of their properties in Table III whether they are 'sequentially-aware', 'triangle-preserving', 'consider high-order Markov chains', and 'position-aware'.

Hyper-parameter Settings. For fair comparison, we implement BPRMF, FPMC, TransRec, PRME and our proposed model using TensorFlow. The learning rate is selected from {0.001,0.005,0.01,0.05,0.1}. For NeuMF[4], NARM[5], Caser[6], Fossil[7] and MARank[8], we adopt the authors' released source code and tune their hyper-parameters in the same way. For BPRMF, FPMC, TransRec, PRME and ATM, the regularization hyper-parameters are searched in {0.00001,0.0001,0.001,0.01,0.1,1,10}. For PRME, we search α in {0.1,0.2,0.4,0.6,0.8}. For ATM, we tune the hyper-parameter L in {1,2,3,4,5}. All experiments are conducted on a server equipped with Intel XeonCPU E5-2637@ 3.50GHz on 128GB, 3 NVIDIA GeForce Tian X Pascal (12GB for each).

B. Performance Analysis

Comparison with State-of-The-Arts (RQ1) We now evaluate the performance of ATM by comparing the results on four datasets with its competitors. Table IV shows recommendation accuracy in two metrics *w.r.t.* the embedding size. Moreover, we conduct the paired two-sample t-test experiments, demonstrating that the improvements are statistically stable and non-contingent (p-value $<$ 0.01). Due to space limitation, they are omitted. Several comparisons are maked to better understand and explian our findings as follows:

BPRMF vs. NeuMF. By combining the strength of linear MF and the power of non-linear multi-layer perceptron model, NeuMF substantially outperforms BPR on all datasets. This observation is consistent with that in [2].

[4]https://github.com/hexiangnan/neural_collaborative_filtering
[5]https://github.com/lijingsdu/sessionRec_NARM
[6]https://github.com/graytowne/caser_pytorch
[7]https://sites.google.com/view/ruining-he/.
[8]https://github.com/voladorlu/MARank

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

TABLE IV: Recommendation performance (%) of different methods on four datasets with N=50. Column 'Improve' indicates the percentage of improvements that ATM achieves relative to the * results. The best performing method in each case is boldfaced (higher is better).

Dataset	Metric	BPRMF	NeuMF	NARM	Caser	FPMC	PRME	Fossil	TransRec	MARank	ATM	Improve
Cellphone	HR	5.8384	6.3658	6.6573	7.2947	7.7145	7.8618	8.0687	8.6854	8.8431*	**9.6502**	8.07%
K=16	NDCG	1.7785	1.8932	1.9681	2.1739	2.3355	2.3626	2.4386	2.7320	2.9010*	**3.2006**	10.32%
	HR	7.1504	7.8601	7.9375	8.0345	8.2393	8.5935	8.8953	9.3545	9.6127*	**10.6893**	11.20%
K=64	NDCG	2.3374	2.4816	2.5240	2.5570	2.6323	2.8985	2.9339	3.0666	3.3518*	**3.9217**	17.00%
Tool	HR	3.4487	4.2179	4.3651	4.4053	4.5060	4.6786	4.8584	5.2234	5.4369*	**5.8235**	7.11%
K=16	NDCG	0.9745	1.3155	1.4502	1.5325	1.5862	1.6046	1.6301	1.6607	1.7822*	**1.9038**	6.82%
	HR	4.0654	4.4218	4.6863	4.7955	4.9087	5.0312	5.2360	5.5507	5.8349*	**6.5375**	12.04%
K=64	NDCG	1.2874	1.5574	1.6217	1.6524	1.6681	1.7102	1.7658	1.8464	1.9386*	**2.2019**	13.58%
Clothing	HR	1.3672	1.4208	1.4536	1.4706	1.4785	1.4966	1.6057	1.8084	2.0896*	**2.3236**	11.19%
K=16	NDCG	0.4111	0.4668	0.4772	0.4887	0.4999	0.5066	0.5399	0.5652	0.6135*	**0.7028**	14.55%
	HR	1.8323	1.8959	1.9235	1.9436	1.9515	2.0191	2.2854	2.5358	2.7233*	**2.9327**	7.68%
K=64	NDCG	0.6675	0.6919	0.7031	0.7088	0.7067	0.7359	0.8338	0.8870	0.9691*	**1.1305**	16.65%
Epinions	HR	1.4863	1.5235	1.5319	1.5928	1.6620	1.6834	1.7183	1.7313	1.7807*	**1.8802**	5.58%
K=16	NDCG	0.3746	0.4227	0.4368	0.4406	0.4492	0.4978	0.5139	0.5216	0.5569*	**0.6206**	11.43%
	HR	1.5928	1.6422	1.6997	1.7198	1.7313	1.7688	1.8698	2.0083	2.2096*	**2.3367**	5.75%
K=64	NDCG	0.4132	0.4318	0.4725	0.4906	0.5047	0.5832	0.6977	0.7344	0.7687*	**0.8209**	6.79%

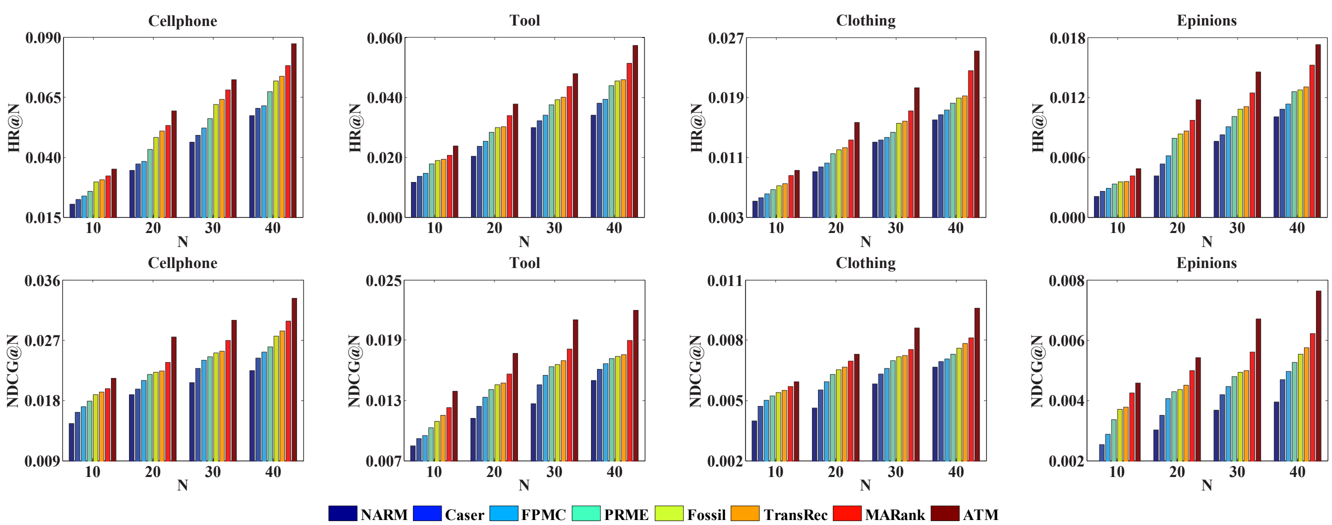


Fig. 4: Performance for different values of N (K=64).

NARM vs. BPRMF & NeuMF. Compared to BPRMF and NeuMF, NARM focus on capturing short-term dynamics among items. Remarkably, NARM achieves comparable prediction accuracy with BPRMF and NeuMF. This means that it is important to model sequential patterns in the next-item recommendation task.

FPMC vs. PRME. PRME shows consistent improvements over FPMC. This is because the inner product does not satisfy the crucial triangle inequality, which limits the expressiveness of FPMC.

TransRec vs. FPMC & PRME. Notably, TransRec has competitive performance with FPMC and PRME. This indicates that it is more reasonable to utilize translation mechanism to model user-item and item-item relationships.

FPMC vs. Fossil. Fossil achieves considerably better results than FPMC in all cases, which verifies the usefulness of high-order Markov chains in the next-item recommendation task.

Fossil vs. Caser. It is interesting to note that Fossil achieves better performance than Caser with a large margin. This owes to CNN-based model having more parameters to capture high-order transitions (*i.e.,* easy overfitting), whereas carefully designed but simpler model is more effective in high-sparsity datasets.

MARank vs. ATM. As expected, the later model significantly outperforms the former one. It indicates that 1) temporal order plays a significant role for making a meaningful recommendation.; 2) our personalized attention mechanism has the ability of learning adaptive weights on items at different orders and for different users; 3) constructing translation embedding with high-order Markov chains is quite reasonable.

Note that for all results presented so far, the size of the top-N list chosen is 50 (*i.e.,* N = 50). Next, we evaluate the overall performance achieved by various methods with varying the size of the top-N list. Figure 4 illustrates the HR@N and NDCG@N values of ATM versus some baselines for different values of N (*i.e.,* 10, 20, 30 and 40). The main findings are summarized as follows:

- All methods perform consistently with different datasets. Precisely, as N increases, HR@N and NDCG@N also increase.
- MARank can always outperform state-of-the-art CNN-based method — Caser. This indicates that it is beneficial to unify both individual-and union-level sequential patterns for modeling user's short-term preference.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

TABLE V: NDCG@50 of TransRec and ATM with different aggregation operations. MAX represents the max pooling strategy, AVG represents the average pooling strategy, ATT represents the attention mechanism without the positional embedding matrix $\boldsymbol{P}$, and PAT denotes the designed position-aware attention mechanism.

Dataset	Cellphone	Tool	Clothing	Epinions
TransRec	2.7320	1.6607	0.5652	0.5216
MAX	2.7621	1.6943	0.5879	0.5283
AVG	2.8036	1.7211	0.6002	0.5407
ATT	2.9751	1.8697	0.6864	0.5932
PAT	**3.2006**	**1.9038**	**0.7028**	**0.6206**

- ATM achieves the best performance in all datasets, further justifying that modeling high-order Markov chains as translation embedding is realistic and reasonable.

Study of Aggregation Operation (RQ2) To get a better understanding of our method, it is necessary to investigate the key component of ATM — aggregation operation. Table V shows the results of ATM with different aggregation operations. There are three key observations:

- **TransRec vs. MAX/AVG.** Our ATM method with the MAX strategy or the AVG strategy consistently outperforms TransRec in all cases. It demonstrates the benefit of using high-order Markov chains and the flexibility of our proposed ATM method.
- **MAX/AVG vs. ATT.** When the ATT mechanism is applied to our method ATM, the performance for next-item recommendation is significantly improved as compared with max pooling and average pooling. This because ATM with ATT mechanism is capable of assigns personalized weights on the L^{th} order Markov chains depending on user embeddings and item embeddings.
- **ATT vs. PAT.** This comparison shows the effect of using positional embedding matrix. Table V shows that adding positional embeddings causes ATM's performances increasing dramatically on four real-world datasets. The good quality of ATM with PAT mechanism demonstrates that our position-aware attention mechanism is capable of capturing user's varying attentions at different orders and on different items.

Effect of Different Orders of Markov Chains (RQ3) As there is little work on modeling high-order Markov chains with translation technique, it is necessary to discuss whether using a high-order translation is beneficial to the next-item recommendation task. Towards this end, we further studied ATM with different orders of Markov chains. L=4 indicates ATM with fourth-order Markov chains, and similar notions for others. Figure 5 shows the experimental results of both methods on NDCG@50. Results of another metric exhibit similar conclusions but is omitted for space reasons. From the figure, we have the following observations:

- It is easy to see that, the recommendation performance of TransRec is equal to ATM with a first-order Markov chain. This is because that setting L to 1 means that ATM boils down to TransRec.
- As we increase the order of the Markov chain (from 2 to 5), the performance tends to initially increase and then decrease. On all datasets, the optimal L value is 3 or 4. On Epinions, we find that when the L value is larger than 3, the performance of ATM starts to drop. It reveals that using a small order is sufficient for modeling short-term temporal dynamics, since the few most recent user-item interactions capture enough information to predict next action.

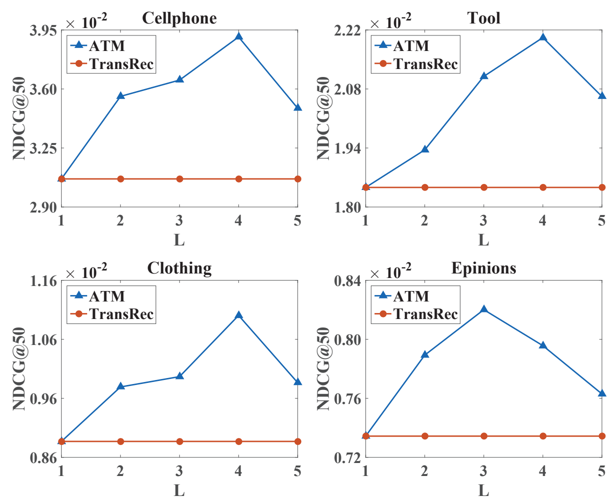


Fig. 5: Performance comparisons for different values of L on four real-world datasets (K=64).

Case Study (RQ4) Apart from the superior recommendation performance, another main advantage of ATM is its ability in interpreting user's varying attentions on different items. To illustrate this, we present some micro-level case studies in Table VI. Specifically, we randomly selected two users (*i.e.,* #806 and #3752) from cellphone and clothing datasets, and they recently bought four items which are shown in column 2 to 5. The target items (*i.e.,* #837 and #5786) are positive examples in the testing set. From the two cases, we find that the most important items appear in the end of activity sequences. This phenomenon confirms that the most recent action is more correlated with the next action. But this does not mean that higher-order items are meaningless. For instance, the two users also put high attentions on 4^{th} order items (*i.e.,* #128 and #745). To demonstrate the rationality, we further study the content of these items. We have following observations from real-world data: (1) In the fist case on cellphone, the target item (*i.e.,* #837) is iphone finger iRing, and the higher attended items (*i.e.,* #128 and #1080) are Black iphone and iphone case, while the lower attended items (*i.e.,* #2657 and #311) are USB wall charger and Earphone. Just as expected, when predicting user's purchase decision-making, the recent items, which is more relevant to the target item, were putted more attentions. (2) In the another case on clothing, the target item (*i.e.,* #5786) is a hat with black and red colors, and the higher attended items (*i.e.,* #745 and #9802) are a red coat and a pair of black shoes, while the lowest attended item (*i.e.,* #1260) is a white sweater. This well justifies our motivating example (*i.e.,* Fig. 1) in introduction, providing evidence that our method has the ability of uncovering meaningful patterns in consumption sequences.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

TABLE VI: Attention weights breakdown of two sampled users.

Dataset	t-4	t-3	t-2	t-1	Target item ID
Cellphone	#128 0.36	#2657 0.05	#311 0.13	#1080 0.46	#837
Clothing	#745 0.31	#1260 0.09	#5891 0.22	#9802 0.38	#5786

V. Conclusion

In this paper, we proposed a simple yet effective model, *i.e., ATM*, to exploit high-order sequential information for next-item recommendation. Specifically, we constructed a user-specific translation vector by aggregating multiple recent items, which encode more information about a user's short-term preference than the most recent item. In sharp contrast to two typical aggregation operations (*i.e.,* max pooling and average pooling), we designed a personalized attention mechanism for ATM, which has the capability of learning different weights on items at different orders and for different users. To evaluate our model, we conducted extensive experiments on multiple real-world datasets and found that ATM with the designed attention mechanism consistently surpasses with several state-of-the-art methods. In future, we plan to examine how to incorporate auxiliary information such as textual reviews [66], [67], [68] and social networks [13], [57] into ATM and analyze their effects for next-item recommendation.

References

[1] B. Smith and G. Linden, "Two decades of recommender systems at amazon.com," *IEEE Internet Computing*, vol. 21, no. 3, pp. 12–18, 2017.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 173–182.

[3] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[4] N. Koenigstein and Y. Koren, "Towards scalable and accurate item-oriented recommendations," in *Proceedings of the 7th ACM conference on Recommender Systems*, 2013, pp. 419–422.

[5] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q. Zhu, "A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 3, pp. 579–592, 2016.

[6] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, "An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1273–1284, 2014.

[7] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008, pp. 263–272.

[8] C. Ma, Y. Zhang, Q. Wang, and X. Liu, "Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 697–706.

[9] B. Yi, X. Shen, H. Liu, Z. Zhang, W. Zhang, S. Liu, and N. Xiong, "Deep matrix factorization with implicit feedback embedding for recommendation system," *IEEE Transactions on Industrial Informatics*, pp. 1–11, 2019.

[10] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys*, vol. 52, no. 1, pp. 5:1–5:38, 2019.

[11] I. Bayer, X. He, B. Kanagal, and S. Rendle, "A generic coordinate descent framework for learning from implicit feedback," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 1341–1350.

[12] X. Kong, F. Xia, J. Wang, A. Rahim, and S. K. Das, "Time-location-relationship combined service recommendation based on taxi trajectory data," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1202–1212, 2017.

[13] X. Wang, X. He, L. Nie, and T. Chua, "Item silk road: Recommending items from information domains to social users," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 185–194.

[14] R. L. Rosa, G. M. Schwartz, W. V. Ruggiero, and D. Z. Rodríguez, "A knowledge-based recommendation system that includes sentiment analysis and deep learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2124–2135, 2019.

[15] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, 2010, pp. 811–820.

[16] C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin, "Collaborative metric learning," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 193–201.

[17] R. He, W. Kang, and J. McAuley, "Translation-based recommendation," in *Proceedings of the 8th ACM Conference on Recommender Systems*, 2017, pp. 161–169.

[18] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T. Chua, "Outer product-based neural collaborative filtering," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2227–2233.

[19] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 729–732.

[20] Y. Zhang, K. Meng, W. Kong, and Z. Y. Dong, "Collaborative filtering-based electricity plan recommender system," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1393–1404, 2019.

[21] Y. Zhang, K. Meng, W. Kong, Z. Y. Dong, and F. Qian, "Bayesian hybrid collaborative filtering-based residential electricity plan recommender system," *IEEE Transactions on Industrial Informatics*, pp. 1–10, 2019.

[22] X. Luo, M. Zhou, S. Li, Y. Xia, Z. You, Q. Zhu, and H. Leung, "An efficient second-order approach to factorize sparse matrices in recommender systems," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 4, pp. 946–956, 2015.

[23] J. Yi, R. Jin, S. Jain, and A. K. Jain, "Inferring users' preferences from crowdsourced pairwise comparisons: A matrix completion approach," in *Proceedings of the 1st AAAI Conference on Human Computation and Crowdsourcing*, 2013.

[24] Y. Koren, "Factorization meets the neighborhood: A multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 426–434.

[25] J. Xu, Y. Yao, H. Tong, X. Tao, and J. Lu, "Hoorays: High-order optimization of rating distance for recommender systems," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 525–534.

[26] P. Ram and A. G. Gray, "Maximum inner-product search using cone trees," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 931–939.

[27] A. Shrivastava and P. Li, "Asymmetric lsh for sublinear time maximum inner product search," in *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014, pp. 2321–2329.

[28] S. Zhang, L. Yao, A. Sun, S. Wang, G. Long, and M. Dong, "Neurec: On nonlinear transformation for personalized ranking," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3669–3675.

[29] S. Zhang, L. Yao, L. V. Tran, and A. Zhang, "Quaternion collaborative filtering for recommendation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 1–8.

[30] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, and E. Chen, "Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1734–1743.

[31] S. Feng, X. Li, Y. Zeng, G. Cong, Y. M. Chee, and Q. Yuan, "Personalized ranking metric embedding for next new POI recommendation," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015, pp. 2069–2075.

[32] R. He and J. J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *Proceedings of the 16th IEEE International Conference on Data Mining*, 2016, pp. 191–200.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2019.2947174, IEEE Transactions on Industrial Informatics

[33] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, 2018, pp. 565–573.

[34] A. Yan, S. Cheng, W. Kang, M. Wan, and J. McAuley, "Cosrec: 2d convolutional neural networks for sequential recommendation," in *Proceedings of the 28th ACM International Conference on Conference on Information and Knowledge Management*, 2019, pp. 1–4.

[35] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proceedings of the 4th International Conference on Learning Representations*, 2016, pp. 1–10.

[36] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 2018, pp. 843–852.

[37] C. Xu, P. Zhao, Y. Liu, J. Xu, V. S. S.Sheng, Z. Cui, X. Zhou, and H. Xiong, "Recurrent convolutional neural network for sequential recommendation," in *Proceedings of the 28th International Conference on World Wide Web*, 2019, pp. 3398–3404.

[38] L. Yu, C. Zhang, S. Liang, and X. Zhang, "Multi-order attentive ranking model for sequential recommendation," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019, pp. 5709–5716.

[39] L. Chen, H. Zhang, J. Xiao, L. Nie, J. Shao, W. Liu, and T. Chua, "Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6298–6306.

[40] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proceedings of the 5th International Conference on Learning Representations*, 2015, pp. 1–15.

[41] X. He, Z. He, J. Song, Z. Liu, Y. Jiang, and T. Chua, "NAIS: neural attentive item similarity model for recommendation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 12, pp. 2354–2366, 2018.

[42] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T. Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017, pp. 335–344.

[43] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 1419–1428.

[44] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[45] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2011, pp. 541–550.

[46] A. Bordes, J. Weston, and N. Usunier, "Open question answering with weakly supervised embedding models," in *Proceedings of Joint European conference on machine learning and knowledge discovery in databases*, 2014, pp. 165–180.

[47] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015, pp. 2181–2187.

[48] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, 2011, pp. 809–816.

[49] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of the 26th International Conference on Neural Information Processing Systems*, 2013, pp. 2787–2795.

[50] R. Jenatton, N. L. Roux, A. Bordes, and G. Obozinski, "A latent factor model for highly multi-relational data," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, 2012, pp. 3176–3184.

[51] C. Park, D. Kim, X. Xie, and H. Yu, "Collaborative translational metric learning," in *Proceedings of the 18th IEEE International Conference International Conference on Data Mining*, 2018, pp. 367–376.

[52] Y. Tay, L. A. Tuan, and S. C. Hui, "Latent relational metric learning via memory-based attention for collaborative ranking," in *Proceedings of the 27th International Conference on World Wide Web*, 2018, pp. 729–739.

[53] X. Yang, Y. Ma, L. Liao, M. Wang, and T. Chua, "Transnfcm: Translation-based neural fashion compatibility modeling," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019, pp. 403–410.

[54] R. Pasricha and J. J. McAuley, "Translation-based factorization machines for sequential recommendation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018, pp. 63–71.

[55] T. Qian, B. Liu, Q. V. H. Nguyen, and H. Yin, "Spatiotemporal representation learning for translation-based POI recommendation," *ACM Transactions on Information Systems*, vol. 37, no. 2, pp. 18:1–18:24, 2019.

[56] D. Cao, X. He, L. Miao, Y. An, C. Yang, and R. Hong, "Attentive group recommendation," in *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018, pp. 645–654.

[57] T. Zhao, J. McAuley, and I. King, "Leveraging social connections to improve personalized ranking for collaborative filtering," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014, pp. 261–270.

[58] X. He, H. Zhang, M. Kan, and T. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 549–558.

[59] X. He, Z. He, X. Du, and T.-S. Chua, "Adversarial personalized ranking for recommendation," in *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018, pp. 355–364.

[60] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T. Chua, "Attentional factorization machines: Learning the weight of feature interactions via attention networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 2017, pp. 3119–3125.

[61] J. Ding, G. Yu, X. He, Y. Quan, Y. Li, T. Chua, D. Jin, and J. Yu, "Improving implicit recommender systems with view data," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 3343–3349.

[62] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009, pp. 452–461.

[63] S. Kabbur, X. Ning, and G. Karypis, "Fism: Factored item similarity models for top-n recommender systems," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013, pp. 659–667.

[64] Y. Koren, "Collaborative filtering with temporal dynamics," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 447–456.

[65] C. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, 2017, pp. 495–503.

[66] C. Ma, P. Kang, B. Wu, Q. Wang, and X. Liu, "Gated attentive-autoencoder for content-aware recommendation," in *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, 2019, pp. 519–527.

[67] H. Zhang, Y. Ji, J. Li, and Y. Ye, "A triple wing harmonium model for movie recommendation," *IEEE Transactions Industrial Informatics*, vol. 12, no. 1, pp. 231–239, 2016.

[68] H. Zhang, Y. Sun, M. Zhao, T. W. S. Chow, and Q. M. J. Wu, "Bridging user interest to item content for recommender systems: An optimization model," *IEEE Transactions on Cybernetics*, pp. 1–13, 2019.

Bin Wu is currently a PhD student with the School of Information Engineering at Zhengzhou University of China. He received his B.S. and M.S. degrees in Computer Science and Technology from Zhengzhou University. He worked six months as a visiting scholar in School of Computing, National University of Singapore. His research interests include machine learning, recommender systems, social network analysis, and multimedia.

Xiangnan He is currently a professor with the University of Science and Technology of China (USTC). He received his Ph.D. in Computer Science from National University of Singapore (NUS) in 2016, and did postdoctoral research in NUS until 2018. His research interests span information retrieval, data mining, and multi-media analytics. He has over 50 publications appeared in several top conferences such as SIGIR, WWW, and MM, and journals including TKDE, TOIS, and TMM. His work on recommender systems has received the Best Paper Award Honourable Mention in WWW 2018 and ACM SIGIR 2016. Moreover, he has served as the PC member for several top conferences including SIGIR, WWW, MM, KDD etc., and the regular reviewer for journals including TKDE, TOIS, TMM, TNNLS etc.

Zhongchuan Sun is currently working toward the PhD degree in the School of Information Engineering at Zhengzhou University. He received his B.S. and M.S. degrees in Computer Science and Technology from Zhengzhou University. His research interests include machine learning and recommender systems.

Liang Chen is currently an associate professor in the School of Data and Computer Science, Sun Yat-Sen University, China. He received the B.S. and PhD degrees in computer science from Zhejiang University, Hangzhou, China, in 2009 and 2015, respectively. He has many publications appeared in well-known conference proceedings and international journals, e.g, ICML, IJCAI, ICDE, WWW, ICSE, ICDM, CIKM, ICSOC, ICWS, TSC, TSMC, etc. He has served as the PC member or reviewer for several top conferences/journal including TNNLS, TKDE, ICSOC, etc. His research interests include recommender systems, graph computing, adversarial learning, and service mining.

Yangdong Ye is a Professor with School of Information Engineering at Zhengzhou University. He received his Ph.D. degree in China Academy of Railway Sciences. He has wide research interests, mainly including machine learning, pattern recognition, knowledge engineering and intelligent system. He has published several papers in peer-reviewed prestigious journals and conference proceedings, such as IEEE Transactions on Multimedia, Neural Networks, IEEE CVPR, IJCAI and ACM Multimedia. More details about his research and background can be found at http://www5.zzu.edu.cn/mlis/