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 [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

\[d(a, c) \leq d(a, b) + d(b, c).\]
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 the same weight for different items, which occur in the same order at each time step. ConvolutionAI Sequence Embedding
Collaborative filtering, He et al. [42], [43]. For instance, in the NAIS model for item-based recommendation, which unifies both individual- and union-level 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 $head\ entity + relation \approx tail\ entity$; 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 $S^u$ is known: $S^u = \{S^u_1, S^u_2, \ldots, S^u_{|S^u|}\} \in \mathcal{I}$. The action history of all users is $A = \{S^1, \ldots, S^{|U|}\}$. Given each user $u$ and her/his behavior sequence $S^u$, 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>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>(U, \mathcal{I})</td>
<td>user set, item set</td>
</tr>
<tr>
<td>(u, j, t)</td>
<td>user (u) in (U), item (j) in (\mathcal{I}), a specific time step</td>
</tr>
<tr>
<td>(S_u^t)</td>
<td>action sequence of user (u): ({S_u^t, \ldots, S_u^{</td>
</tr>
<tr>
<td>(S_u^t)</td>
<td>the item that user (u) interacted with at time step (t)</td>
</tr>
<tr>
<td>(f_j^u)</td>
<td>embedding vector associated with user (u)</td>
</tr>
<tr>
<td>(g_j^u)</td>
<td>embedding vector associated with item (j)</td>
</tr>
<tr>
<td>(a)</td>
<td>global translation vector</td>
</tr>
<tr>
<td>(T_u(t))</td>
<td>translation vector associated with user (u) at time step (t)</td>
</tr>
<tr>
<td>(b_j)</td>
<td>bias term associated with item (j)</td>
</tr>
<tr>
<td>(d(x, y))</td>
<td>distance between (x) and (y)</td>
</tr>
</tbody>
</table>

![Diagram](image)

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 \(\gamma_u^U \in \mathbb{R}^K\) and \(\gamma_j^I \in \mathbb{R}^K\) be the embedding vector of user \(u\) and item \(I\), 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 \(I\), the following relation should hold

\[
\gamma_u^U + T_u(t) \approx \gamma_j^I.
\]

In other words, \(\gamma_j^I\) 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 \(\{S_{u,t-L}, \ldots, S_{u,t-1}\}\), \(T_u(t)\) can be obtained from an \(L^{th}\) order Markov chains which is defined as follows:

\[
T_u(t) = a + f(\gamma_{t-L}^I, \ldots, \gamma_{t-1}^I),
\]

where \(f(\cdot)\) denotes the aggregation operation and \(a\) 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:

\[
f(\gamma_{t-L}^I, \ldots, \gamma_{t-1}^I) = \sum_{l=1}^{L} \alpha_l \gamma_{t-l}^I,
\]

where \(\alpha_l\) 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 \(\beta_u = \{\beta_{u,1,1}, \beta_{u,2,1}, \ldots, \beta_{u,1,L}\}\) for the target user \(u\). The revised schema is designed as follows:

\[
f(\gamma_{t-L}^I, \ldots, \gamma_{t-1}^I) = \sum_{l=1}^{L} (\alpha_l + \beta_{u,l}) \gamma_{t-l}^I.
\]

While the above solution sounds to be reasonable, it comes across a main obstacle. The size of \(\beta\) is very huge (i.e., \(|U| \times L\)), 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\(^{st}\) 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:

\[
h(u, t, l) = \phi (\gamma_{t-l}^I W_1 + P_2 W_2 + b),
\]

\[
\alpha(u, t, l) = \frac{\exp (\gamma_{t-l}^I h^T(u, t, l))}{\sum_{g=1}^{2K} \exp (\gamma_{t-g}^I h^T(u, t, l))},
\]

where \(W_1 \in \mathbb{R}^{K \times K}\), \(W_2 \in \mathbb{R}^{K \times K}\) and \(b \in \mathbb{R}^K\) are the trainable parameters. As we can see, the size of these parameters (i.e., \(2K^2 + (L + 1)K\), \(K \ll |U|\)) is much smaller than that of \(\beta\). Unlike standard attention mechanism that is unaware of the positions of recent items [38], [42], we inject a learnable position embedding matrix \(P \in \mathbb{R}^{L \times K}\) 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 \(\gamma_u^U\) of user \(u\) as the context vector and get the adaptive weight \(\alpha(u, t, l)\) as the normalized similarity between \(h(u, t, l)\) and \(\gamma_u^U\) with a softmax function. Figure 3 illustrates the architecture of ATM.
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:

$$f(\gamma_{S_{t-1}}^{u}, \cdots, \gamma_{S_{t-L}}^{u}) = \sum_{l=1}^{L} \alpha(u, t, l) \cdot \gamma_{S_{t-l}}^{u},$$  \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 $S_{t}^{u}$ being the next item (at time step $t$) with an $L^{th}$ order Markov chains is predicted by:

$$\text{Prob}(S_{t}^{u} \mid S_{t-1}^{u}, \cdots, S_{t-L}^{u}) \propto \mu_{S_{t}} - d(\gamma_{S_{t}}^{u}, T_{l}(t), \gamma_{S_{t-l}}^{u}),$$  \quad (8)

where $d(\cdot)$ denotes the $L_{2}$ distance. In Eq. (8), we add a single bias term $\mu$ to capture the overall item popularity.

### C. Optimization Criterion

Given a user u and the previous action sequence \{\$S_{1}^{u}, \ldots, S_{t-1}^{u}\}, $S_{t}^{u}$ > $u, t, j$ denotes that item $S_{t}^{u}$ is ranked higher than item $j$ for user u. Here it is a natural choice to optimize such ranking between $S_{t}^{u}$ 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 posteriori:

$$\arg \max_{\Theta} \prod_{u \in U} \prod_{t=L+1}^{L} \prod_{j \neq S_{t}^{u}} \text{Prob}(S_{t}^{u} > u, t, j \mid \Theta) \text{Prob}(\Theta),$$  \quad (9)

where $\Theta = \{a, b, \mu, \gamma_{u \in U}, \gamma_{j \in I}^{l}, W_{1}, W_{2}, P\}$ is the set of our model parameters. By employing a sigmoid function $\sigma(z) = \frac{1}{1 + e^{-z}}$, the ranking probability can be rewritten as the following:

$$\text{Prob}(S_{t}^{u} > u, t, j \mid \Theta) = \text{Prob}(\hat{p}_{u, t, S_{t}^{u}} - \hat{p}_{u, t, j} > 0 \mid \Theta)$$

$$= \sigma(\hat{p}_{u, t, S_{t}^{u}} - \hat{p}_{u, t, j}),$$  \quad (10)

where $\hat{p}_{u, t, S_{t}^{u}}$ 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 $\lambda$ is a regularization hyper-parameter.

$$\arg \max_{\Theta} \ln \prod_{u \in U} \prod_{t=L+1}^{L} \sum_{j \neq S_{t}^{u}} \text{Prob}(S_{t}^{u} > u, t, j \mid \Theta) \text{Prob}(\Theta)$$

$$= \arg \max_{\Theta} \sum_{u \in U} \sum_{l=L+1}^{L} \sum_{j \neq S_{t}^{u}} \ln \sigma(\hat{p}_{u, t, S_{t}^{u}} - \hat{p}_{u, t, j}) - \lambda \|\Theta\|^2.$$  \quad (11)

Due to the huge number of $(u, t, S_{t}^{u}, j)$ 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: $\Theta_{1} = \{a, \mu, \gamma_{u \in U}, \gamma_{j \in I}^{l}\}$ and $\Theta_{2} = \{b, W_{1}, W_{2}, P\}$. The first part of our model parameters is the same as TransRec and grows linearly with users and items. For parameters $\Theta_{2}$ (i.e., $2K^{2} + (L + 1)K$), they are shared among all users and items, 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.
Algorithm 1: The optimization for ATM.

**Input:** sequential data \( A \), learning rate \( \epsilon \), regularization hyper-parameter \( \lambda \);

**Output:** global transition bias \( \alpha \), item bias \( \mu \), embedding vectors \( \gamma_U, \gamma_I \), position embedding matrix \( P \), and parameters in attention network \( b, W_1, W_2 \);

1. Initialize \( \alpha, \mu, \gamma_U, \gamma_I, P \) with Gaussian distribution and \( b, W_1, W_2 \) with Xavier;
2. repeat
   3. draw \( (u, t, S^u_t) \) from \( A \);
   4. draw \( j \) from \( I \) with Xavier;
   5. for \( l \leftarrow 1 \) to \( L \). do
     6. Compute \( h(u, t, l) \) according to Eq. (5);
     7. for \( l \leftarrow 1 \) to \( L \). do
       8. Compute \( \alpha(u, t, l) \) according to Eq. (6);
       9. Compute \( T_u(t) \) according to Eq. (2) and (7);
       10. \( \gamma^U_u \leftarrow \gamma^U_u + T_u(t) \);
       11. Compute \( \hat{p}_{u,t,l} \) and \( \hat{p}_{u,t,j} \) according to Eq. (8);
       12. \( \hat{R}(u, t, S^u_t, j) = \hat{p}_{u,t,l} - \hat{p}_{u,t,j} \);
     13. for each parameter \( \theta \) in \( \Theta \) do
       14. \( \theta \leftarrow \theta + \frac{\partial \hat{R}(u, t, S^u_t, j)}{\partial \theta} - \lambda \theta \);
   15. until convergence;
16. return \( \Theta = \{ \alpha, b, \mu, \gamma^U_u, \gamma^I_I, W_1, W_2, P \} \).

**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 \( \hat{p}_{u,t,S^u_t} \) 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 \( \{S^u_{t-1}, \ldots, S^u_{t-L}\} \), the time complexity of obtaining an \( \alpha(u, t, l) \) is \( O(K^2L) \). Considering the \( \alpha(u, t, l) \) term is shared across the computation of \( \hat{p}_{u,t,S^u_t} \) and \( \hat{p}_{u,t,j} \), we only need to compute it once and cache it. As such, the computation of \( \hat{p}_{u,t,S^u_t} - \hat{p}_{u,t,j} \) takes additional time cost \( O(K^2L) \). 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 \( K \ll \min(|U|, |I|) \). 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.**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>User#</th>
<th>Item#</th>
<th>Action#</th>
<th>Sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cellphone</td>
<td>7,622</td>
<td>36,121</td>
<td>110,539</td>
<td>99.95%</td>
</tr>
<tr>
<td>Tool</td>
<td>7,945</td>
<td>42,614</td>
<td>131,008</td>
<td>99.96%</td>
</tr>
<tr>
<td>Clothing</td>
<td>25,160</td>
<td>156,091</td>
<td>379,740</td>
<td>99.99%</td>
</tr>
<tr>
<td>Epinions</td>
<td>1,444</td>
<td>17,356</td>
<td>24,331</td>
<td>99.90%</td>
</tr>
</tbody>
</table>

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

- **Amazon.** The first group of datasets is from Amazon.com and spain 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.** 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 to the precision at top-N.
to Precision@N, and is equivalent to Recall@N. Formally, the definitions of HR@N and NDCG@N are shown as follows:

$$\text{HR}@N = \frac{1}{|U|} \sum_{u=1}^{N} \sum_{a=1}^{\lceil \log_2(a+1) \rceil} \delta_u(a),$$

$$\text{NDCG}@N = \frac{1}{|U|} \sum_{u=1}^{N} \frac{2^{h_u(a)} - 1}{\log_2(a+1)},$$

where $|U|$ 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 [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.
- **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.

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 don’t 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, NARM, Caser, Fossil and MARank, 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 $\alpha$ in {0.1,0.2,0.4,0.6,0.8,1}. 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 made to better understand and explain 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].
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).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Metric</th>
<th>BPRMF</th>
<th>NeuMF</th>
<th>NARM</th>
<th>Caser</th>
<th>FPMC</th>
<th>PRME</th>
<th>Fossil</th>
<th>TransRec</th>
<th>MARank</th>
<th>ATM</th>
<th>Improve</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NDCG</td>
<td>1.7785</td>
<td>1.8932</td>
<td>1.9861</td>
<td>2.1739</td>
<td>2.3555</td>
<td>2.3626</td>
<td>2.4386</td>
<td>2.7320</td>
<td>2.9010*</td>
<td>3.2066</td>
<td>10.32%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5066</td>
<td>0.4227</td>
<td>12.04%</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>NDCG</td>
<td>2.3374</td>
<td>2.4816</td>
<td>2.5240</td>
<td>2.5570</td>
<td>2.6323</td>
<td>2.8985</td>
<td>2.9339</td>
<td>3.0666</td>
<td>3.3518*</td>
<td>3.9217</td>
<td>17.00%</td>
</tr>
<tr>
<td>Tool</td>
<td>K=16</td>
<td>HR</td>
<td>3.4487</td>
<td>4.2179</td>
<td>4.3651</td>
<td>4.4053</td>
<td>4.5060</td>
<td>4.68/6</td>
<td>4.8854</td>
<td>5.2234</td>
<td>5.4369*</td>
<td>5.8235</td>
</tr>
<tr>
<td></td>
<td>NDCG</td>
<td>0.9745</td>
<td>1.3155</td>
<td>1.4502</td>
<td>1.5325</td>
<td>1.5862</td>
<td>1.6946</td>
<td>1.6301</td>
<td>1.6607</td>
<td>1.7822*</td>
<td>1.9038</td>
<td>6.82%</td>
</tr>
<tr>
<td></td>
<td>K=64</td>
<td>HR</td>
<td>4.0654</td>
<td>4.4218</td>
<td>4.6863</td>
<td>4.7955</td>
<td>4.9087</td>
<td>5.0312</td>
<td>5.2360</td>
<td>5.5507</td>
<td>5.8349*</td>
<td>6.5375</td>
</tr>
<tr>
<td></td>
<td>NDCG</td>
<td>1.2874</td>
<td>1.5574</td>
<td>1.6217</td>
<td>1.6524</td>
<td>1.6681</td>
<td>1.7102</td>
<td>1.7058</td>
<td>1.8464</td>
<td>1.9586*</td>
<td>2.2019</td>
<td>15.58%</td>
</tr>
<tr>
<td>Clothing</td>
<td>K=16</td>
<td>HR</td>
<td>1.3672</td>
<td>1.4208</td>
<td>1.4536</td>
<td>1.4706</td>
<td>1.4785</td>
<td>1.4966</td>
<td>1.6057</td>
<td>1.8084</td>
<td>2.0896*</td>
<td>2.3236</td>
</tr>
<tr>
<td></td>
<td>NDCG</td>
<td>0.4111</td>
<td>0.4668</td>
<td>0.4772</td>
<td>0.4887</td>
<td>0.4999</td>
<td>0.5066</td>
<td>0.5399</td>
<td>0.5652</td>
<td>0.6155*</td>
<td>0.7028</td>
<td>14.55%</td>
</tr>
<tr>
<td></td>
<td>K=64</td>
<td>HR</td>
<td>1.8323</td>
<td>1.8959</td>
<td>1.9235</td>
<td>1.9436</td>
<td>1.9515</td>
<td>2.0191</td>
<td>2.2854</td>
<td>2.5358</td>
<td>2.7223*</td>
<td>2.9327</td>
</tr>
<tr>
<td></td>
<td>NDCG</td>
<td>0.6675</td>
<td>0.6919</td>
<td>0.7031</td>
<td>0.7088</td>
<td>0.7067</td>
<td>0.7359</td>
<td>0.8338</td>
<td>0.8870</td>
<td>0.9691*</td>
<td>1.1305</td>
<td>16.65%</td>
</tr>
<tr>
<td>Epinions</td>
<td>K=16</td>
<td>HR</td>
<td>1.4863</td>
<td>1.5235</td>
<td>1.5319</td>
<td>1.5928</td>
<td>1.6620</td>
<td>1.6834</td>
<td>1.7183</td>
<td>1.7313</td>
<td>1.7807*</td>
<td>1.8802</td>
</tr>
<tr>
<td></td>
<td>NDCG</td>
<td>0.3746</td>
<td>0.4227</td>
<td>0.4368</td>
<td>0.4406</td>
<td>0.4492</td>
<td>0.4978</td>
<td>0.5139</td>
<td>0.5216</td>
<td>0.5569*</td>
<td>0.6206</td>
<td>11.43%</td>
</tr>
<tr>
<td></td>
<td>K=64</td>
<td>HR</td>
<td>1.5928</td>
<td>1.6422</td>
<td>1.6997</td>
<td>1.7198</td>
<td>1.7313</td>
<td>1.7688</td>
<td>1.8698</td>
<td>2.0083</td>
<td>2.2096*</td>
<td>2.3367</td>
</tr>
<tr>
<td></td>
<td>NDCG</td>
<td>0.4132</td>
<td>0.4318</td>
<td>0.4725</td>
<td>0.4906</td>
<td>0.5047</td>
<td>0.5552</td>
<td>0.5977</td>
<td>0.7344</td>
<td>0.7587*</td>
<td>0.8209</td>
<td>6.79%</td>
</tr>
</tbody>
</table>

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 easy overfitting), whereas carefully designed but simpler model is more effective in high-sparcity 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.

![Fig. 4: Performance for different values of N (K=64).](image-url)
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 \( P \), and PAT denotes the designed position-aware attention mechanism.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cellphone</th>
<th>Tool</th>
<th>Clothing</th>
<th>Epinions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransRec</td>
<td>2.7320</td>
<td>1.6607</td>
<td>0.5652</td>
<td>0.5216</td>
</tr>
<tr>
<td>MAX</td>
<td>2.7621</td>
<td>1.6943</td>
<td>0.5879</td>
<td>0.5283</td>
</tr>
<tr>
<td>AVG</td>
<td>2.8036</td>
<td>1.7211</td>
<td>0.0002</td>
<td>0.5407</td>
</tr>
<tr>
<td>ATT</td>
<td>2.9751</td>
<td>1.8697</td>
<td>0.0864</td>
<td>0.5932</td>
</tr>
<tr>
<td>PAT</td>
<td>3.2006</td>
<td>1.9038</td>
<td>0.7028</td>
<td>0.6206</td>
</tr>
</tbody>
</table>

- 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 assigning 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.

**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.
TABLE VI: Attention weights breakdown of two sampled users.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>t-1</th>
<th>t-2</th>
<th>t-3</th>
<th>t-4</th>
<th>Target item ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cellphone</td>
<td>#128</td>
<td>0.36</td>
<td>#2057</td>
<td>0.05</td>
<td>#8311</td>
</tr>
<tr>
<td>Clothing</td>
<td>#745</td>
<td>0.31</td>
<td>#1260</td>
<td>0.09</td>
<td>#5891</td>
</tr>
</tbody>
</table>

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


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, ICSC, 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/