Time-aware Multi-interest Capsule Network for Sequential Recommendation

Zhikai Wang* Yanyan Shen†

Abstract

In recent years, sequential recommendation has been widely researched, which aims to predict the next item of interest based on user's previously interacted item sequence. Many works use RNN to model the user interest evolution over time. However, they typically compute a single vector as the user representation, which is insufficient to capture the variation of user diverse interests. Some non-RNN models employ the dynamic routing mechanism to automatically vote out multiple capsules that represent user's diverse interests, but they are ignorant of the temporal information of user's historical behaviors, thus yielding suboptimal performance. In this paper, we aim to establish a time-aware dynamic routing algorithm to effectively extract temporal user multiple interests for sequential recommendation. We observe that the significance of an item to user interests may change monotonically over time, and user interests may fluctuate periodically. Following the intuitive temporal patterns of user interests, we propose a novel time-aware multi-interest capsule network named TAMIC that leverages two kinds of time-aware voting gates, i.e., monotonic gates and periodic gates, to control the influence of each interacted item on user's current interests during the routing procedure. We further employ an aggregation module to form a temporal multi-interest user representation which is used for next item prediction. Extensive experiments on real-world datasets verify the effectiveness of the time gates and the superior performance of our TAMIC approach on sequential recommendation, compared with the state-of-the-art methods.

1 Introduction

In decades, the rapid development of the e-commerce industry has spawned an increasing demand for personalized recommendation. In addition to the traditional setting that treats recommendation as a matrix completion problem, recent researches have devoted efforts to the sequential recommendation problem [13, 19].

A large number of works [15, 10] utilized recurrent neural networks (RNNs) to model user interests evolving in the passage of time. While achieving encouraging recommendation performance, existing RNN-based methods only capture one user interest by referring to the last item's hidden state [10] or performing aggregation over all the hidden states of the items in the sequence [24]. The single user interest representation is then used for the prediction of the next item. In practice, a user may have multiple interests encoded in the historical behaviors, but RNNs are incapable of modeling and distinguishing user's diverse interests, thus yielding suboptimal performance [4].

Recent works [4, 14] adopted the capsule networks [20] and employed the dynamic routing mechanism to automatically vote out several significant user interests based on the interacted items in user's historical behaviors. The obtained multiple user interests have proved beneficial to the final recommendation performance [14]. And these methods decouple the process of inferring user behavior representations and measuring user-item correlations, providing high efficiency for the matching stage with billion-scale items. However, existing capsule network based methods completely ignore the temporal information such as the timestamps of individual interactions and the time intervals between consecutive interactions. As a result, two user behavior sequences involving the same set (or multiset) of items but with different item orders or time intervals would derive the same user multi-interest representation, which is counter-intuitive and faces the risk of information loss. Hence, applying capsule networks for sequential recommendation needs further consideration.

Previous works [2, 6] tried to explicitly incorporate temporal information into the sequence modeling. Some of them [3] simply treated the timestamp of each interaction as an additional feature, while others [6] explored the influence of different time intervals on next item prediction. Nevertheless, the proposed time-aware deep models can be regarded as the enhancements of RNN-based or self-attention-based methods for sequential recommendation. They generally rely on the sequential structure of RNN or the positional encoding in self-attention to model temporal user interests. Unfortunately, existing techniques cannot directly apply to capsule networks due to the architecture difference.

In this paper, we aim to establish a time-aware multi-interest capsule network for sequential recommendation which enhances the dynamic routing mechanism

*Shanghai Jiao Tong University, Email: Cloudcatcher.888@sjtu.edu.cn

†Corresponding Author. Shanghai Jiao Tong University, Email:shenyy@sjtu.edu.cn

with temporal information to capture temporal multiple interests of users. Our work is motivated by the following key observations. First, the significance of an item to user interests may change monotonically over time. For example, a user who has a great passion on a new-series of toys might become less interested in them a few months later, indicating the user's interest on toys decays. Meanwhile, some try-on products that were initially purchased with a small quantity may suddenly become user's favorites and even dominate over other interests for a long time, exhibiting an enhancing trend on the interest. In either way, during the dynamic routing procedure, the timestamp of an interacted items imply its significance to user's current interests. Second, user interests may fluctuate seasonally or periodically at different frequencies and phases. For example, people would like buy more T-shirts in summer but more coats in winter. The basic necessities such as toothpaste will be purchased repeatedly over relatively fixed periods of time. In order to learn more accurate user interests from interacted items, it is plausible to allow an interest to be extracted from selective items appeared at a certain frequency and phase.

Following our observations, we design two kinds of time-aware voting gates, monotonic gates and periodic gates, to control the voting signals from items to high-level interests based on their timestamps. Specifically, the monotonic gates change monotonically according to the time interval between the target item and the controlled items, while the periodic gates fluctuate periodically over time with different periods and phases. We integrate the two kinds of gates into the dynamic routing procedure, and develop a Time-Aware Multi-Interest Capsule network (TAMIC for short) which is effective to capture both multiple user interests and temporal information for sequential recommendation. Extensive experimental results on real-world datasets verify the effectiveness of the two kinds of gates and the superior performance of our proposed TAMIC. To summarize, this paper makes the following major contributions.

- We propose to enhance the dynamic routing mechanism to incorporate temporal information for multi-interest-based sequential recommendation.
- We design two kinds of time-aware voting gates to capture the monotonic and periodic patterns in user interests, which can be seamlessly applied to capsule-network-based recommendation models.
- We conduct extensive experiments on three real datasets, the results on which demonstrate that our proposed TAMIC achieves 2.52%-5.18% higher NDCG than the state-of-the-art approaches [4, 23] and the two kinds of gates are effective to extract temporal multiple interests of users.

2 Related Work

Sequential recommendation attempts to learn user representations from historical item interaction sequences [13, 15, 19]. GRU4Rec [10] leverages Recurrent Neural Network (RNN) to capture sequential behavioral patterns of users. ACVAE [23] proposes using VAE to generate high-quality item representations and capture global and local item correlations in the sequence. However, the above methods all preserve one single latent vector for each user, which is insufficient to capture the variation of user's diverse interests [14, 18]. Besides, they ignore the time information involved in the user behavior sequences, thus yielding suboptimal performance. Recently, some researches [18, 25] have utilized capsule networks to capture user's diverse interests. The capsule network [20] was firstly used for image classification. It uses vectorized capsules to replace scalar neurons in neural networks and employs the dynamic routing mechanism based on a similarity metric to form capsules. Inspired by the high expressiveness of capsule networks, MIND [14] performs dynamic routing to extract user's high-level multi-interest capsules from the raw user behavior sequence for item recommendation. ComiRec [4] is another capsule-network-based sequence recommendation model, which optimizes the dynamic routing process and introduces a controllable method for interest selection. Unfortunately, the above methods fail preserve the sequential order among interactions and lack the ability of utilizing temporal information for sequential recommendation.

The temporal information including both the timestamps of individual interactions and the time intervals provides critical clues on the item to be interacted next. Various works have tried to exploit temporal information to boost the performance of sequential recommendation [1, 2, 6]. However, existing time-aware deep models typically treat timestamp as a context feature of interactions and cannot model monotonic or periodic patterns of temporal user interests explicitly.

3 Notations and Problem

Typically, a recommendation dataset consists of interaction records between a set $\mathcal{U}$ of users and a set $\mathcal{I}$ of items. Each interaction can be represented as a triplet (u, i, t), where $u \in \mathcal{U}$ is a user id, $i \in \mathcal{I}$ is an item id, and t is a UNIX timestamp. Let $\mathcal{A} = \{(u, i, t)\}$ denote all the interactions in the dataset. In the sequential recommendation setting, each instance consists of two parts. The output part is the triplet (u, i_a, t_a) describing that the user u interacts with the target item $i_a \in \mathcal{I}$ at the timestamp t_a. The input part is u's historical behaviors which is a sequence of u's previously interacted items ordered by timestamp, denoted as $\mathcal{S}_u = \{i \mid (u, i, t) \in \mathcal{A}, t < t_a\}$.

In practice, the sequential recommendation aims to find the top-N candidate items based on user's historical behaviors. More specifically, we want to learn a function f_{user} which can map a user's historical behaviors $\mathcal{S}_u$ into a multi-interest user representation $\boldsymbol{H}_u$:

$$\boldsymbol{H}_u = f_{user}(\mathcal{S}_u) = (\boldsymbol{h}_{u,1}, \cdots, \boldsymbol{h}_{u,K}) \in \mathbb{R}^{d \times K} \tag{3.1}$$

where $\boldsymbol{h}_{u,k}$ denotes the vector of the k^{th} interest of u ($k \in [1, K]$), K is the total number of user interests, and d is the dimension of each interest vector. The top-N candidate items are retrieved using a scoring function:

$$f_{score}(\boldsymbol{H}_u, \boldsymbol{e}_i) \tag{3.2}$$

where $\boldsymbol{e}_i \in \mathbb{R}^{d \times 1}$ denotes the embedding of item $i \in \mathcal{I}$. In what follows, we omit the subscription u when the context is clear.

4 Methodology

4.1 Embedding Module The input to this module is the user behavior sequence $\mathcal{S}_u$ and the target item i_a. This module implements an embedding layer that converts the one-hot vector $\boldsymbol{x}_i$ of item $i \in \mathcal{I}$ into a low-dimensional embedding $\boldsymbol{e}_i$, which is defined as follows:

$$\boldsymbol{e}_i = \boldsymbol{W}_{emb}\boldsymbol{x}_i \tag{4.3}$$

where $\boldsymbol{W}_{emb} \in \mathbb{R}^{d \times |\mathcal{I}|}$ is the embedding matrix, and d denotes the embedding dimension. Henceforth, we use $E_u = \{\boldsymbol{e}_i \mid (u, i, t) \in \mathcal{S}_u\}$ to denote the the set of items involved in the user behavior sequence $\mathcal{S}_u$, and use $\boldsymbol{e}_a$ to denote the target item embedding.

4.2 Time-aware Multi-interest Learning In MIND [14] and ComiRec [4], they used dynamic routing mechanism to extract user's time-irrelevant multi-interest representation. However, as we have mentioned before, the significance of each item to high-level interests will change over time. Hence, in this module we augment the standard dynamic routing algorithm with two kinds of time-aware gates to control the voting signals from item embeddings to user's interest representation vectors, allowing the user high-level interest capsules to concern the temporal information during the vote assignment.

4.2.1 Multi-Gated Dynamic Routing We first describe the dynamic routing procedure in the context of sequential recommendation. The original purpose of applying capsule network with dynamic routing is to exploit more detailed hidden information from the raw features [20]. MIND [14] has transferred the dynamic routing algorithm to the sequential recommendation task in order to learn a group of interests from a user's behavior sequence. Specifically, the item embeddings in E_u are transformed into low-level capsules, which will later be used to vote for K high-level capsules. Formally, each item embedding $\mathbf{e}_i \in E_u$ is transformed into K capsules' space using different affine transforming matrices $\{\boldsymbol{W}_k\}_{k=1}^K$ as follows:

$$\hat{\boldsymbol{e}}_{i|k} = \boldsymbol{W}_k\boldsymbol{e}_i, \qquad k \in [1, K] \tag{4.4}$$

where $\boldsymbol{W}_k \in \mathbb{R}^{d \times d}$. Then a dynamic routing procedure will be applied on the transformed low-level capsules for L iterations. In the l^{th} ($l \in [1, L]$) iteration, we aim to compute K high-level capsules $\{\mathbf{h}_k^{(l)}\}_{k=1}^K$. To do this, we first calculate the candidate vector $\boldsymbol{s}_k^{(l)}$ for the k^{th} high-level capsule, which is defined as follows:

$$\boldsymbol{s}_k^{(l)} = \sum_{i=1}^{|E_u|} c_{ik}^{(l)} \hat{\boldsymbol{e}}_{i|k} \tag{4.5}$$

$$c_{ik}^{(l)} = \frac{\exp(\hat{\boldsymbol{e}}_{i|k}\boldsymbol{h}_k^{(l-1)})}{\sum_{k=1}^K \exp(\hat{\boldsymbol{e}}_{i|k}\boldsymbol{h}_k^{(l-1)})} \tag{4.6}$$

The coupling coefficients between capsule i and all the capsules in the next layer should sum to 1. The k^{th} high-level capsules in the l^{th} iteration is computed as:

$$\boldsymbol{h}_k^{(l)} = squash(\boldsymbol{s}_k^{(l)}) = \frac{\|\boldsymbol{s}_k^{(l)}\|^2}{1 + \|\boldsymbol{s}_k^{(l)}\|^2} \frac{\boldsymbol{s}_k^{(l)}}{\|\boldsymbol{s}_k^{(l)}\|} \tag{4.7}$$

where the squash function leaves the direction of the vector $\boldsymbol{s}_k$ unchanged but decreases its magnitude. Intuitively, Eq. (4.5)-(4.7) softly cluster the low-level capsules into K different high-level capsules, which can be regarded as K different interests extracted from the user behavior sequence.

However, the standard dynamic routing process is ignorant of the sequential order and the difference of the time-intervals in $\mathcal{S}_u$. Specifically, two item sequences with different interaction time-intervals or interaction orders will be clustered into the same group of interest capsules. Further, the monotonicity and periodicity patterns are also ignored. Hence, we refine the dynamic routing procedure by introducing a gate $g_{ik}^{(l)}$ in Eq. (4.5) to control the strength of votes in a time-aware manner. The refined voting equation is:

$$\boldsymbol{s}_k^{(l)} = \sum_{i=1}^{|E_u|} g_{ik}^{(l)} c_{ik}^{(l)} \hat{\boldsymbol{e}}_{i|k} \tag{4.8}$$

The gate $g_{ik}^{(l)}$ is composed of two time-aware gates via pooling:

$$g_{ik}^{(l)} = pooling(\bar{g}_{ik}^{m,(l)}, \bar{g}_{ik}^{p,(l)}) \tag{4.9}$$

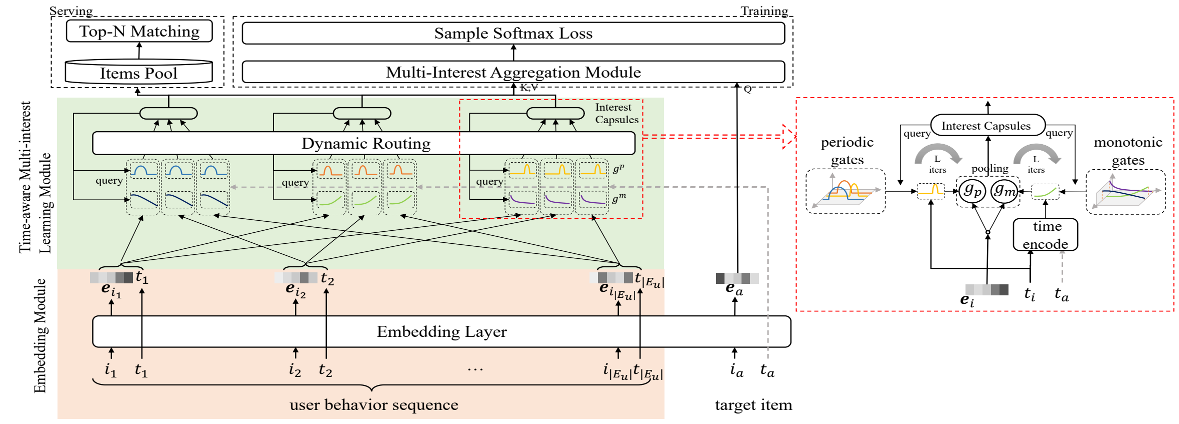


Figure 1: Overview of the TAMIC model.

where the $pooling(\cdot)$ can be max-pooling or average-pooling operation. The $\bar{g}_{ik}^{m,(l)}$ and $\bar{g}_{ik}^{p,(l)}$ denote the monotonic gate and periodic gate respectively. At a high level, we have:

$$\bar{g}_{ik}^{m,(l)} = f_{monotonic\_gate}(t_i, t_a, \boldsymbol{h}_k^{(l-1)}) \tag{4.10}$$

$$\bar{g}_{ik}^{p,(l)} = f_{periodic\_gate}(t_i, \boldsymbol{h}_k^{(l-1)}) \tag{4.11}$$

The refining enables us to model the monotonic change of significance of items to high-level interests, and the periodic fluctuation of high-level interests during dynamic routing. If the interest fluctuates monotonically like toys, the monotonic gate will open or close gradually while the periodic gate will close all the time, and vice versa. The next two sections will discuss in detail how these two gates are designed and how the monotonicity and periodicity patterns are concerned during dynamic routing procedure. The pseudocode of the time-aware dynamic routing is provided in Algorithm 1.

4.2.2 Monotonic Gate As we discussed before, the significance of some items to high-level interests will change monotonically over time. As the interest-specific item embeddings $\{\hat{\mathbf{e}}_{i|k}\}$ form high-level user interests, we can utilize the UNIX timestamp of each interacted item to control its influence on a particular interest. Some methods [1] used dampen functions to model the decay of a signal over time. However, these methods have three drawbacks: (1) the dampen function mainly captures the decaying signal but ignores the enhancing trend; (2) the decay coefficient in the existing methods requires dedicated adjustment; (3) all the time intervals share one dampen function which may not be suitable for the multi-interest scenario. Other works [6] chose to learn unique embeddings for different timestamps, taking the risk of parameter explosion.

In order to rectify these shortcomings, we firstly leverage the time encoding [3] to encode the timestamp information into a low-dimensional temporal embedding $\boldsymbol{t} \in \mathbb{R}^d$. Given a UNIX timestamp t, the i^{th} value in the temporal embedding $\mathbf{t}$ is computed as:

$$\boldsymbol{t}_i = \begin{cases} \sin(t/10000^{i/d}), & \text{if } i \text{ is odd} \\ \cos(t/10000^{i/d}), & \text{otherwise} \end{cases} \tag{4.12}$$

The time encoding is unique for different timestamps and preserves the sequential order, which means the euclidean distance between the embeddings of two distant dates can be larger. It is worth mentioning that not all the user behavior sequences start from the same UNIX timestamp 1, and hence we normalize the timestamps in each behavior sequence by separating the time span of the sequence into several baskets. We then assign each timestamp into the corresponding basket and use the regular basket numbers to replace the timestamps accordingly. The implementation details can be found in Section 4.5.

Given the time embeddings, a simple method to implement the monotonic gate is to use a neural network that takes the time embedding of an interaction as input and produces a value indicating the influence of a particular item on the interests. However, there can be different monotonic patterns and each user interest may involve a mixture of multiple monotonic patterns. Hence, instead of using one neural network, we propose to use a group of $G1$ networks, each of which controls the openness of one child monotonic gate. We further leverage the attention mechanism to obtain the final monotonic gate by aggregating the openness statuses of all the child gates.

The openness of each child monotonic gate is determined by the time information of an interacted item t_i, that of the target item t_a, and the time interval $t_a - t_i$. Formally, the j^{th} child monotonic gate of the i^{th} item in the user behavior sequence is formulated as:

$$g_i^{m,j} = f^j([\boldsymbol{t}_\delta \oplus \boldsymbol{t}_i \oplus \boldsymbol{t}_a]),\ j \in [1, G1] \tag{4.13}$$

Algorithm 1: Time-aware Dynamic Routing

```
Input  : behavior embeddings and timestamps
         $(\boldsymbol{e}_i, t_i), i \in \mathcal{S}_u$;
         target timestamp $t_a$;
         iteration times $L$;
         number of interest capsules $K$;
         monotonic gate embeddings
         $\{\boldsymbol{a}^{m,j}, j \in [1, G1]\}$;
         periodic gate embeddings
         $\{\boldsymbol{a}^{p,j}, j \in [1, G2]\}$;
Output: interest capsules in the final iteration
         $\{\boldsymbol{h}_k^{(l)}, k = 1, \cdots, K\}$
1  For each interest capsule $k$: initialize $\boldsymbol{h}_k = \boldsymbol{0}$;
2  for each item $i$ do
3      for $j \in [1, G1]$, calculate $g_i^{m,j}$ by Eq. (4.13);
4      for $j \in [1, G2]$, calculate $g_i^{p,j}$ by Eq. (4.15);
5  end
6  for $l \leftarrow [1, L]$ do
7      for each item $i$ do
8          calculate $\bar{g}_{ik}^{m,(l)}$,$\bar{g}_{ik}^{p,(l)}$ by
            Eq. (4.14),Eq. (4.17);
9      end
10     For each interest capsule $k$: calculate $\boldsymbol{h}_k^{(l)}$ by
        Eq. (4.8);
11 end
```

where $\oplus$ refers to concatenation operation. $\boldsymbol{t}_\delta$ denotes the time encoding of the time interval $t_a - t_i$. f denotes the neural network which is implemented by multilayer perceptron(MLP) in our experiments. We use the sigmoid function as the last layer of the MLP to bound the gate output within 0 and 1, i.e., $g_i^{m,j} \in (0, 1)$. Note that the neural networks in the child gates are initialized differently to learn different monotonic patterns, and they are shared by all the iterations and interests. We then use an attention mechanism to calculate the weighted average of all the child gates as the final monotonic gate $\bar{g}_{ik}^{m,(l)}$ from the i^{th} item to the k^{th} interest in the l^{th} iteration. To be interest-specific, $\bar{g}_{ik}^{m,(l)}$ is computed based on the interest representation $\boldsymbol{h}_k^{(l-1)}$, as defined:

$$\bar{g}_{ik}^{m,(l)} = \sum_{j=1}^{G1} \frac{\boldsymbol{a}^{m,j\top} \boldsymbol{h}_k^{(l-1)}}{\sum_{b=1}^{G1} \boldsymbol{a}^{m,b\top} \boldsymbol{h}_k^{(l-1)}} g_i^{m,j} \tag{4.14}$$

where $\boldsymbol{a}^{m,j} \in \mathbb{R}^{d\times 1}$ is the embedding for the j^{th} child monotonic gate, which serves as the key to calculate the matching score with $\boldsymbol{h}_k^{(l-1)}$. Although the child gates are shared among all the interests, Eq. (4.14) will assign different weights to them, leading to different monotonic gates for different interests.

4.2.3 Periodic Gate

The design of the periodic gate is motivated by the phased-LSTM [17] which introduces a time gate into the vanilla LSTM. The time gate will open and close periodically with different periods and phases in each channel so as to capture the periodically changed signals such as the sine functions. In the dynamic routing procedure, the periodic gate aims to capture the periodicity patterns for user interests. Similar to the monotonic gate, we define $G2$ child periodic gates with different initial phase s and period τ to cope with various periodicity patterns. To better simulate user's typical interaction habits like daily, weekly, seasonal or annual purchasing, the period of each child gate are manually chosen from a multiple of days, weeks, months or years. In opposite, the phases are learnable parameters trained together with the model. The subscription k and the superscription l will be omitted for convenience. Formally, the j^{th} child periodic gate of i^{th} item in user behavior sequence is computed as:

$$g_i^{p,j} = \begin{cases} \sin(\pi\phi_i^j / r^j), & 0 < \phi_i^j < r^j \\ \alpha\phi_i^j, & r^j < \phi_i^j < 1 \end{cases} \tag{4.15}$$

$$\phi_i^j = \frac{(t_i - s^j) \mod \tau^j}{\tau^j} \tag{4.16}$$

where the $r^1, \cdots, r^{G2}$ are parameters to be learned, denoting the interest existing ratios in the whole period. α is manually set as the inactive value. The $\tau^1, \cdots, \tau^{G2}$ are the changing periods of user interests.

Similar to the monotonic gate, we also use the attention mechanism to calculate the weighted average of all the child periodic gates as the final periodic gate. Formally, the periodic gate $\bar{g}_{ik}^{p,(l)}$ from the i^{th} item to the k^{th} interest in the l^{th} iteration is computed as:

$$\bar{g}_{ik}^{p,(l)} = \sum_{j=1}^{G2} \frac{\boldsymbol{a}^{p,j\top} \boldsymbol{h}_k^{(l-1)}}{\sum_{b=1}^{G2} \boldsymbol{a}^{p,b\top} \boldsymbol{h}_k^{(l-1)}} g_i^{p,j} \tag{4.17}$$

where $\boldsymbol{a}^{p,j} \in \mathbb{R}^{d\times 1}$ is the embedding for the j^{th} child periodic gate, and $\boldsymbol{h}_k^{(l-1)}$ is the k^{th} interest obtained from the previous iteration.

4.3 Multi-Interest Aggregation

After performing L iterations in the time-aware multi-interest learning module, we obtain K high-level capsules $\boldsymbol{H}_u = \{\boldsymbol{h}_1^{(L)}, \cdots, \boldsymbol{h}_K^{(L)}\}$ representing user's different interests. In this module, we use the target item embedding $\boldsymbol{e}_a$ as the query to evaluate the importance of each user interest. We then derive a target-item-sensitive user representation $\boldsymbol{v}_u \in \mathbb{R}^{d\times 1}$ by aggregating user interests in an attentive manner. Formally, we have:

$$\boldsymbol{v}_u = \sum_{k=1}^{K} \frac{\boldsymbol{e}_a{}^\top \boldsymbol{h}_k^{(L)}}{\sum_{k'=1}^{K} \boldsymbol{e}_a{}^\top \boldsymbol{h}_{k'}^{(L)}} \boldsymbol{h}_k^{(L)} \tag{4.18}$$

4.4 Training and Serving

Finally, we perform dot product between $\boldsymbol{v}_u$ and the target item embedding $\boldsymbol{e}_a$ to produce the preference score, i.e., $\boldsymbol{v}_u{}^\top \boldsymbol{e}_a$.

To optimize the parameters of our TAMIC model, we maximize the following objective function:

$$\mathcal{L} = \sum_{\mathcal{S}_u \in \mathcal{T}} \log \frac{\exp(\boldsymbol{v}_u{}^\top \boldsymbol{e}_a)}{\sum_{i \in \mathcal{I}'} \exp(\boldsymbol{v}_u{}^\top \boldsymbol{e}_i)} \tag{4.19}$$

where $\mathcal{T}$ denotes the set of training instances, and $\mathcal{I}' \subset \mathcal{I} \backslash \{i_a\}$ is a small sampled subset of items excluding the target item i_a.

During the serving stage, the multi-interest aggregation module will be discarded because the target item is unknown. Each interest vector of a user can independently retrieve top-N items from the large-scale item pool by the nearest neighbor library such as Faiss [12]. Then we compute the score of the items based on their inner production proximity with user interests to obtain the final top-N items, where the score function is:

$$f_{score}(\boldsymbol{H}_u, \boldsymbol{e}_i) = \max_{1 \le k \le K} \boldsymbol{e}_i{}^\top \boldsymbol{h}_k^{(L)} \tag{4.20}$$

where $\boldsymbol{e}_i \in \mathbb{R}^{d \times 1}$ is the embedding of item i in $\mathcal{I}$.

4.5 Implementation Details We implemented our proposed TAMIC model based on Tensorflow 1.15 framework and trained on a 64-bit Linus server equipped with 32 Intel Xeon@2.10GHz CPUs, 128GB memory and four Titan RTX 2080ti GPUs. The embedding matrix of items $\boldsymbol{W}_{emb}$ and the gates' embeddings $\{\boldsymbol{a}^{p,j}\}$,$\{\boldsymbol{a}^{m,j}\}$ are initialized randomly, where we choose 64 as the embedding dim. By default, we set both numbers of child monotonic gates and child periodic gates to 8, and set the number of high-level interest capsules K to 4. The inactive value α is set to 0.1. The batch size is set to 128. Compared with MIND [14], the extra parameters of TAMIC are in the MLPs of the monotonic gates in Eq. (4.13) and the key embeddings of the two kinds of gates in Eq. (4.14) and Eq. (4.17), which are negligible compared to the parameters of item embeddings. Thus the total computation complexity of TAMIC keeps same to that of MIND [14] which is $O(K|E_u|d)$ and K, $|E_u|$, d refers to the interest capsules number, the user's historical interaction sequence length and the embedding dimension, respectively.[1]

5 Experiments

5.1 Experimental Settings

5.1.1 Datasets We use three real-world sequential recommendation datasets from Amazon[2] and Taobao[3]. The statistics of the datasets are listed in Table 1.

[1]Code is in https://github.com/Cloudcatcher888/TAMIC
[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://tianchi.aliyun.com/dataset

Table 1: The statistics of the datasets.

Dataset	#users	#items	#interactions
Amazon Clothing	285,464	376,859	5,748,920
Amazon Books	459,133	313,966	8,898,041
Taobao	976779	1708530	85384110

- **Amazon**: this dataset consists of reviews of different kinds of products from Amazon [16]. We consider two categories of products and obtain **Amazon Clothing** and **Amazon Books** subsets.

- **Taobao**: this dataset is collected from the e-commerce platform Taobao [9]. We use the item id and the UNIX timestamp for our experiments.

5.1.2 Baselines and Evaluation Protocols We compare our proposed TAMIC with various existing sequential and non-sequential recommendation methods. As there will be unseen users in the validation and test sets, MF-based methods are inapplicable to our setting. We consider four categories (I: non-RNN, II: RNN/CNN, III: time-aware, IV: capsule-network-based) of comparison methods as follows.

- **Most Popular (I)** This recommends the top-N globally most popular items to all the users.
- **YouTube DNN (I)** [21] is a basic deep learning model, which uses multi-layer perceptrons for preference prediction.
- **GRU4Rec (II)** [10] is a typical GRU-based method for sequential recommendation.
- **ACVAE (II)** [23] is a state-of-the-art model which employs VAE to learn high-quality item representations and apply a recurrent and convolutional structure to capture global and local item correlations in the sequence.
- **TIEN (III)** [6] is an enhanced version of DIEN which consider the timestamp information to enrich the item implicit semantic information in RNN.
- **TiSASRec (III)** [2] is time interval aware transformer recommendation framework which also considers the timestamp information.
- **MIND (IV)** [14] is a multi-interest recommendation framework, which is the first work to adopt the dynamic routing algorithm for modeling multiple interests from user behavior sequences.
- **ComiRec (IV)** [4] is the state-of-the-art recommendation method which employs self-attention to learn user multi-interest representations.

We used the published source codes of all the comparison methods except for the Most Popular and tuned the hyperparameters to their best performance. For fair comparison, we follow the same data preprocessing and data splitting rule in ComiRec [4] for our method and baselines, which splits all users into training/validation/test sets by the proportion of 8:1:1. We train models using the entire user behavior sequences of training users based on Eq. (4.18). TAMIC and models in Category IV are tested in the serving stage using Eq. (4.20). For validation and testing, we uses the first 80% of user behaviors to infer user embeddings and compute metrics by predicting the remaining 20% user behaviors. We use the hit ratio (HR), recall and NDCG on Top20/50 as the metrics, which are commonly used in sequential recommendation works [4, 19].

5.2 Performance Comparison Table 2 provides the evaluation results of different methods on three datasets. The influence of random initialization to the results is less than 0.0003. The results on different metrics are consistent. From Table 2, we can see that TAMIC improves the recommendation performance significantly compared with all the baselines and it reports the best performance in all the cases. On average, TAMIC achieves 6.4%, 5.9% and 4.2% relative improvements than the best baseline model on the three datasets, respectively. The models in Category I do not consider sequential user behaviors, and hence perform worse than the methods in Category II and III. The RNN-based and CNN-based models in Category II capture the sequential information of user behaviors. The time aware methods in Category III can capture both the time order information and the time interval information of user behaviors, which can achieve better performance than GRU4REC and DIEN. However, models in Category II and III generally perform worse than multi-interest networks in Category IV. This indicates the importance of using multiple vectors to learn users' different interests. Both MIND and ComiRec ignore the important temporal information during dynamic routing, resulting in inferior performance compared with TAMIC. Thanks to the decoupled approximate nearest neighbor search, the inference time of selecting thousands of candidate items from the billion-scale item pool during serving stage can be less than 15 milliseconds.

5.3 Ablation Study We perform an ablation study on *Amazon Clothing* to evaluate the effects of different components of TAMIC. Similar conclusions can be drawn using the other datasets. We mainly consider the following settings.

- **BaseModel:** The base model of TAMIC that only uses the dynamic routing algorithm like MIND [14].
- **ComiRec+Temporal Feature:** This model uses the ComiRec [4] as the base model and **directly** concatenates the time encoding [3] of timestamp with the item embedding as the new item embedding to consider the temporal information simply.
- **BaseModel+Monotonic:** A refined model that only uses the monotonic gate during dynamic routing.
- **BaseModel+Periodic:** A refined model that only uses the periodic gate during dynamic routing.
- **TAMIC (Average-Pooling):** A variant of TAMIC that performs average pooling to combine the outputs from the monotonic gate and the periodic gate in Eq. (4.9).
- **TAMIC (Max-Pooling):** The complete version of the proposed model that performs max pooling to combine the outputs from the monotonic gate and the periodic gate.

The results are presented in Table 3. BaseModel+Periodic improves the top-20 and top-50 NDCGs by 11.13% and 8.35%, respectively. BaseModel+Monotonic improves the top-20 and top-50 NDCGs by 7.16% and 6.61%, respectively. The reason might be that the user interests to clothes involve strong seasonal patterns. Moreover, performing max pooling over two kinds of gates is generally better than average pooling. This is because max pooling enables the reservation of any kind of temporal patterns in the final user multi-interest representation, thus benefiting the next-item prediction. TAMIC outperforms the ComiRec directly using the concatenating time encoding by 5.12% and 4.51% in top-20 and top-50 NDCGs, which shows that the monotonic and periodic gates can capture temporal information more effectively on this dataset.

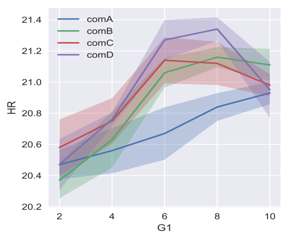
5.4 Parameter Sensitivity We investigate the sensitivity of the number of child monotonic gates G_1, and meanwhile we evaluate the influence of different period combinations of the child periodic gates on the model performance. The value of $G1$ is chosen from {2,4,6,8,10}. The periods of the child periodic gates are chosen from the following combinations: **comA**: [1d, 30d, 365d], **comB**: [0.5d, 1d, 7d, 30d, 365d], **comC**: [1d, 7d, 30d, 60d, 90d, 365d], **comD**: [1d, 7d, 30d, 90d, 180d, 365d], where $1d = 86400$ *seconds*. From **comA** to **comD**, we include more child periodic gates (i.e., $G2 \in \{3, 5, 6\}$) and gradually pay more attention to long-term periodic patterns like monthly or annual periods. Figure 2 shows the top-50 HR results of our model with different values of $G1$ and the four period combinations on **Amazon Clothing** and **Taobao** datasets. The trends on Amazon Books are similar to those on Amazon Clothing. On both datasets, the performance of TAMIC becomes better when $G1$ increases from 2 to 6. As $G1$ exceeds 6, the performance of TAMIC is

Table 2: The performance comparison results. All the results are percentage numbers with '%' omitted.The underlined values mark the best performance of the baselines in each category.

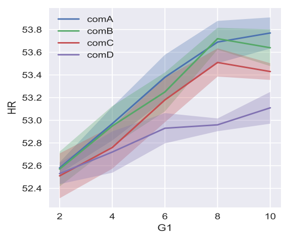
Category			I		II		III		IV		
Method			Most Popular	DNN	GRU4REC	ACVAE	TIEN	TiSASRec	MIND	ComiRec	TAMIC
Amazon Clothing	Top 20	HR	5.38	15.95	14.01	16.42	16.56	17.22	16.05	17.31	**18.23**
		NDCG	4.71	14.97	13.15	14.82	15.03	15.74	14.95	15.96	**17.04**
		recall	3.71	11.03	9.95	10.73	10.98	11.55	10.84	11.91	**12.96**
	Top 50	HR	5.88	17.42	16.51	18.52	19.65	20.21	19.15	20.22	**21.34**
		NDCG	5.12	16.36	15.23	16.75	17.77	18.33	17.84	18.68	**19.90**
		recall	4.98	12.34	11.53	12.21	13.18	13.78	12.92	13.95	**15.02**
Amazon books	Top 20	HR	3.02	10.29	8.95	10.17	11.04	12.05	10.62	12.01	**12.89**
		NDCG	2.26	7.67	6.80	8.45	8.95	8.94	7.93	9.19	**9.64**
		recall	1.37	4.57	4.06	4.81	4.43	4.48	4.86	5.31	**5.67**
	Top 50	HR	5.27	15.89	13.66	15.14	16.31	16.73	16.14	17.58	**18.22**
		NDCG	3.94	12.08	10.37	11.24	12.01	12.49	12.23	13.52	**14.26**
		recall	2.40	7.31	6.50	6.98	7.31	7.36	7.64	8.11	**8.67**
Taobao	Top 20	HR	5.42	28.79	35.75	35.83	36.12	37.15	38.12	41.75	**43.22**
		NDCG	2.07	14.51	22.01	23.28	22.31	21.78	20.39	24.01	**25.02**
		recall	0.40	4.21	5.88	6.18	6.42	6.38	6.28	6.89	**7.42**
	Top 50	HR	9.31	39.11	43.07	45.13	44.64	45.43	45.85	52.42	**53.82**
		NDCG	3.60	20.25	25.31	26.89	28.21	30.11	25.07	31.37	**32.79**
		recall	0.74	6.17	7.49	7.88	7.89	8.84	8.16	9.82	**10.89**

Table 3: Ablation study on Amazon Clothing.

Method	Top 20		Top 50	
	NDCG	Recall	NDCG	Recall
Base Model	14.95	10.84	17.84	12.92
ComiRec + Temporal Feature	16.21	12.31	19.04	13.98
Base Model + Periodic Gate	16.65	12.55	19.33	14.57
Base Model + Monotonic Gate	16.02	11.94	19.02	14.21
TAMIC (Average-Pooling)	**17.06**	12.87	19.78	14.92
TAMIC (Max-Pooling)	17.04	**12.96**	**19.90**	**15.02**

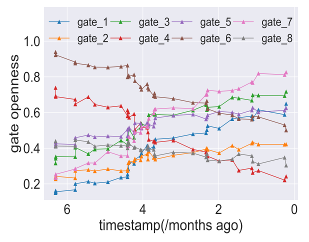


(a) Amazon Clothing

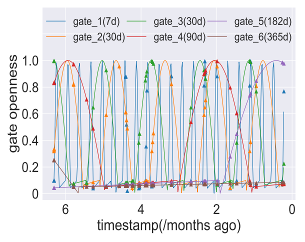


(b) Taobao

Figure 2: The top-50 HR results of TAMIC with different numbers of child monotonic gates and different combinations of child periodic gates.



(a) 8 child monotonic gates



(b) 8 child periodic gates

Figure 3: The openness of child monotonic/periodic gates observed at different timestamps of a sampled user behavior sequence by a test user in Taobao.

more sensitive to the period combinations. This implies the number of different monotonic patterns is not large in practice. The performance of TAMIC on Amazon Clothing is generally increasing when the period combination changes from **comA** to **comD**. In contrast, the performance TAMIC on Taobao exhibits an opposite trend. This follows the intuition that user interests in clothing involve more and longer periodic patterns compared the interests in general goods supplied by Taobao.

5.5 Case Study Lastly, we perform a case study to illustrate the effectiveness of monotonic and periodic gates. We sampled one user behavior sequence involving one toy and one toothpaste, which has a length of N (i.e., $N = 36$). Figure 3 visualizes the 8 child monotonic gates and 6 child periodic gates respectively. We can see that the learned child monotonic gates mimic monotonic functions with different monotonicity such as enhancing (e.g., gate 4,6) and decaying (e.g., gate 1,3,7) trends. Meanwhile, the child periodic gates with the same period can learn different phases and open time ratios (e.g., gate 2,3). These results confirm our proposed gates are effective in capturing different temporal patterns. We then focus on the toy and the toothpaste in the sequence which vote for high-level interest capsules independently. We study the attention scores of the child gates on the most voted interest (i.e., $\max_k\{c_{ik}\}$ in Eq. (4.6)) for each item. The highest attention scores for the toy and toothpaste among the 8 child monotonic gates are observed at gate 7 and gate 4, respectively. This implies the significance of the toy to its related interest decays over time, and the toothpaste acts reversely. We also observe the two highest attention scores for the toothpaste among the 6 child periodic gates are obtained at gate 4. This means the user interest related to toothpastes has a period of about 2-3 months. The attention scores for the toy over all the child periodic gates are similar, which is consistent with the decay pattern of the user interest on toys.

6 Conclusion

In this paper, we propose a time-aware multi-interest capsule network named TAMIC for sequential recommendation, which captures diverse user interests with the consideration of temporal information in user behavior sequences. We introduce two kinds of voting gates, namely monotonic gates and periodic gates, to adaptively control the voting signals from items to user high-level interest capsules based on the timestamps, thus enabling the learning of time-aware user multi-interest representations. The experimental results on real-world datasets verify the superiority of TAMIC compared with various state-of-the-art sequential recommendation approaches. In future work, we will try to apply monotonic gates and periodic gates to non-capsule models like transformer-based sequential recommendation methods.

Acknowledgement

This work is supported by the National Key Research and Development Program of China (No. 2018YFC0831604). Yanyan Shen is also supported by Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), the Tencent Wechat Rhino-Bird Focused Research Program, and SJTU Global Strategic Partnership Fund (2021 SJTU-HKUST).

References

[1] Akihiro Yamaguchi, and Ken Ueno. Learning Time-series Shapelets via Supervised Feature Selection. In *SDM*, pages 9–17, 2021.

[2] Jiacheng Li, Yujie Wang, and Julian McAuley. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*, pages 322–330, 2020.

[3] Xuchao Zhang, Wei Cheng, Bo Zong, Yuncong Chen, Jianwu Xu, Ding Li, and Haifeng Chen. Temporal Context-Aware Representation Learning for Question Routing. In *SDM*, pages 753–761, 2020.

[4] Yukuo Cen, Jianwei Zhang, Xu Zou, Hongxia Yang, and Jie Tang. Controllable Multi-Interest Framework for Recommendation. In *KDD*, 2020.

[5] Jiaxi Tang, and Ke Wang. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*, pages 565–573, 2018.

[6] Xiang Li, Chao Wang, and Tao Zhuang. Deep Time-Aware Item Evolution Network for Click-Through Rate Prediction. In *CIKM*, pages 785–794, 2020.

[7] Zhibo Xiao, Luwei Yang, and Hao Wang. Deep Multi-Interest Network for Click-through Rate Prediction. In *SDM*, pages 265–273, 2020.

[8] Jimmy Lei Ba Diederik. Adam: A Method for Stochastic Optimization. In *ICLR*, pages 785–794, 2015.

[9] Han Zhu, Xiang Li, Pengye Zhang, Han Li, and Kun Gai. Learning Tree-based Deep Model for Recommender Systems. In *KDD*, pages 1079–1088, 2018.

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, Domonkos Tikk. Session-based Recommendations with Recurrent Neural Networks. In *arXiv*, 2015.

[11] Kalervo Järvelin and Jaana Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, 51(2):243–250, 2017.

[12] Jeff Johnson, and Hervé Jégou. Billion-scale similarity search with GPUs. In *arXiv*, 2017.

[13] Wang Kang, Julian McAuley. Self-Attentive Sequential Recommendation. In *ICDM*, pages 197–206, 2018.

[14] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *CIKM*, 2019.

[15] Ee-Peng Lim, Marianne Winslett, and Shane Culpepper. Neural Attentive Session-based Recommendation. In *CIKM*, pages 1419–1428, 2017.

[16] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-Based Recommendations on Styles and Substitutes. pages 43–52, 2015.

[17] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences. In *NIPS*, 2016.

[18] Chenliang Li, Cong Quan, and Libing Wu. A Capsule Network for Recommendation and Explaining What You Like and Dislike. In *SIGIR*, 2019.

[19] Steffen Rendle, Lars Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*, pages 811–820, 2010.

[20] Sara Sabour, Nicholas Frosst, Geoffrey E Hinton. Dynamic Routing Between Capsules. In *arXiv*, 2017.

[21] Paul Covton, and Emre Sargin. Deep Neural Networks for YouTube Recommendations. pages 191–198, 2016.

[22] Fei Sun, Jun Liu, Jian Wu, Changhua Pei. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations. In *CIKM*, 2019.

[23] Zhe Xie, Chengxuan Liu, and Yue Ding. Adversarial and Contrastive Variational Autoencoder for Sequential Recommendation. In *WWW*, 2021.

[24] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep Interest Evolution Network for Click-Through Rate Prediction. In *AAAI*, pages 5941–5948, 2018.

[25] Tingting Liang, Philip S Yu. Joint Training Capsule Network for Cold Start Recommendation. In *IJCAI*, pages 1942–1947, 2020.