Vol.2021-DBS-174 No.8
2021/12/27

DemiRec: Dynamic Evolutionary Multi-Interest Network for Sequential Recommendation

Anyu Cai[1,a)]   Shin'ichi Konomi[1,b)]

Abstract: In recommender systems, capturing the rich sequential information in historical interaction sequences is essential for user representation learning. However, most of the existing methods only focused on modeling the user's historical information into one fixed-length vector, which brings two limitations. First, one fixed-length vector is insufficient to model a user's varying interests. This simplified user modeling leads to monotonous recommendations, making users bored and trapped in the information cocoon. Second, since the user's interests are naturally dynamic and evolving, merely considering past information can only capture the user's outdated interests and predict the interest changes based on a fixed period (i.e., they model the elapsed times since the historical interactions into a same interval). As a result, they cannot give different predictions when users access the system after different time intervals, resulting in suboptimal recommendation performance. In this paper, we seek to explicitly model the users' diverse interests and elapsed time within a sequential interests modeling framework to explore the causality between users' multi-interests and the influence of different elapsed times. We propose a Dynamic Evolutionary Multi-Interest network for sequential RECommendation (DemiRec), which models user's historical interactions into an interest sequence and predicts evolved interests based on sequential information and elapsed time. Extensive empirical studies on real-world datasets demonstrate that the proposed method significantly outperforms state-of-the-art multi-interest baselines.

Keywords: Sequential Recommendation, Dynamic Interests, Multi-interest

1. Introduction

With the development of the Internet, recommendation systems have played an increasingly important role in many fields as an information filtering tool. Recommendation algorithms discover users' interests in content or products based on user profiles and historical interactions. User interests are naturally dynamic. Therefore capturing the sequential information in the interactions has become a key to improving recommendation performance.

In the sequential recommendation, most of the existing works consider the user's historical interactions as an ordered sequence and focused on modeling sequential patterns to predict the user's next interest. Typical methods, such as Markov Chain, predicts the user's next interaction based on few previous interactions [11, 30]. Recurrent neural networks summarizes user's interaction sequence into a fixed-length hidden vector through a cyclic structure [13]. The attention mechanism can learn a set of weights to determine the importance of different parts of the sequence so that the model can better use the information in a long sequence [18]. Besides, temporal recommendation [21, 38] focuses on modeling specific time information in user sequences. The absolute timestamps of user's interactions help capture the periodicity and forgetfulness of users' behaviors.

Generally, previous works only focused on modeling the order and time information in users' interaction sequences. However, such an approach can only capture users' historical interest and simply predict users' interest based on a fixed period. In other words, these methods implicitly assume that the time intervals between the target interactions and the historical interactions (or the elapsed times) are the same. Based on this assumption, a user's predicted interest is used to make recommendations in any elapsed time situation. These methods cannot give different predictions when users access the system after different elapsed times, resulting in suboptimal recommendation performance.

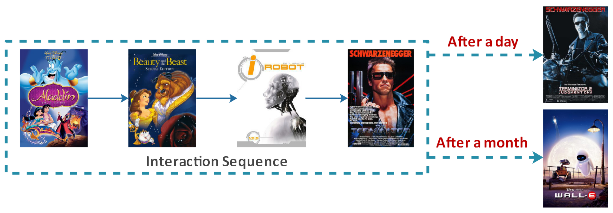


Fig. 1: An example of the effect of elapsed time on recommendation.

In addition, a user's interests are naturally diversified, and only modeling a user as a fixed-length vector is insufficient to represent user's complex interests [22]. Therefore, various methods including nonlinear factorization [35], Capsule Network [22] and the attention mechanism [5] are used to model the user with multiple-interest representations. However, they regard these interests as separate and static, ignoring the possible causality and interaction of the evolving interests.

In this paper, we argue that the causality and interactions of interests are essential for multi-interest recommendation methods. We argue that the elapsed time from historical interactions

[1] HDI Lab, Kyushu University, Fukuoka, Japan
[a)] anyucai@outlook.com
[b)] konomi@artsci.kyushu-u.ac.jp

is an extra important factor for predicting users' evolving interests, as shown in Figure 1. Thus, we propose a dynamic evolutionary multi-interest network for the sequential recommendation (DemiRec) for learning and predicting users' diverse interests with varying degrees of evolution. Inspired by the dynamic routing [22], and the Relative Position Representation of Self-Attention [32], our model first performs soft clustering on users' historical interactions to capture diverse and unordered interests. Then, the timestamps of the interactions in each cluster are used to restore the order and the latest interaction time of the user's interests. After restoring the user's interest sequence, we use the self-attention mechanism to learn the sequential relationship between interests by considering the absolute positions and causality. By considering the elapsed time since the latest interaction of interests, the model can give different predictions according to the possible degree of evolution with the user's interests. To summarize, the main contributions of this work are as follows:

- We propose to consider the causality and interaction relationships between the user's interests and different elapsed times since the user's historical interactions to better predict users' time-evolving interests.
- We designed a novel elapsed time aware multi-interest evolution model called DemiRec by combining the soft clustering of dynamic routing and the advantages of absolute position and elapsed time encodings for self-attention. DemiRec can learn the sequential relationship and interactions across the user's different interests to better predict the evolution of users' complex interests after a certain time interval.
- DemiRec significantly improves diversity with comparable accuracy performance, and outperforms the state-of-the-art multi-interest baseline in terms of both accuracy and diversity. In addition, we present the experiments to study the impact of elapsed time, and different components.

2. Related Work

2.1 Sequential Recommendation

Sequential recommendation methods consider sequential pattern information to obtaind higher accuracy and are widely used in the industry. Traditional methods such as FPMC [30] capture users' long-term interests by Matrix Factorization and adapt users' short-term interests by Markov Chains.

In recent years, deep learning technology has been widely used in sequential recommendation due to its powerful nonlinear representation and generalization ability. Recurrent Neural Network (RNN) has great performance to summarize long sequences into a fixed-length hidden vector. GRU4Rec [13] first introduces RNN based method to leverage complete interaction sequences. DREAM [40] based on RNN, learns the dynamic representation of items in the user's basket to reveal the user's dynamic interest. RRN [36] is the first recurrent recommender network that attempts to capture the dynamics of both user and item representation. HRNN [29] models user's interaction sequence with two levels hierarchical RNN to propagate user's historical interests to current interest. DIEN [41] combines attention mechanism with RNN to capture users' evolving interests. SR-GNN [37] model users' interaction sequences as a graph and use Graph Neural Network to capture the complex relationship between interactions.

2.2 Capsule Network

Capsule Network was first proposed by Hinton, et al . [14] and has become well-known since the dynamic routing method [31] was proposed and showed unique performance in Computer Vision. MIND [22] uses it to do soft clustering on users' historical interactions to capture diverse interests of users. ComiRec [5] studied to capture users' multiple interests by a dynamic routing model and a self-attention model respectively and consideres the order of the sequence. In addition, they proposed to balance the diversity and accuracy performance by a controllable factor. FAT [26] extracts future interactions from neighbors and uses dynamic routing to obtain the trend representation.

2.3 Attention Mechanisms

In recent years, the attention mechanism has gained a lot of consideration with its performance and interpretability in various fields [3, 39]. The attention mechanism determines which part of the input is more important and needs more concentration being paid by learning a set of weights. Earlier, NARM [24] and STAMP [25] incorporated vanilla attention mechanisms to capture user's current preference. Later, the Encoder-Decoder framework Transformer [34], became a breakthrough sequence method on machine translation tasks. It uses the scaled dot-product attention which is defined as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\mathbf{T}}}{\sqrt{d}}\right)\mathbf{V} \quad (1)$$

where $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ represent queries, keys and values respectively, and usually use the same object in self-attention. The self-attention modules of Transformer have also achieved state-of-the-art results on sequential recommendation, such as SASRec [18] and BST [6]. Since the self-attention layers don't treat the inputs as structured data, it is not aware of the order of the items. Vaswani, et al . [34] and Shaw, et al . [32] give solutions to add learnable position embeddings and relative position embeddings, respectively. TiSASRec [23] and MEANTIME [7] leverage the exact time information in the users' interaction sequences, by adding learnable absolute and relative time embeddings. Inspired by Bert [9] which leverages the Transformer, Bert4rec [33] uses bidirectional information of sequence context for training mask embedding of the target item.

3. Methodology

In this section, we first formulate the sequential recommendation problem and then introduce the details of each component of our multi-interest evolution model. The overall structure of our proposed framework is illustrated in Figure 2.

3.1 Problem Formulation

Given a set of users and a set of items which can be denoted as $u \in \mathcal{U}$ and $i \in \mathcal{I}$ respectively. In the setting of sequential recommendation, the implicit feedback of user u to items can be denoted as an item sequence as $\mathcal{S}^u = \left\{i^u_1, i^u_2, \ldots, i^u_{|\mathcal{S}^u|}\right\}$ where $i^u_t \in \mathcal{I}$ ordered by the corresponding timestamp sequence

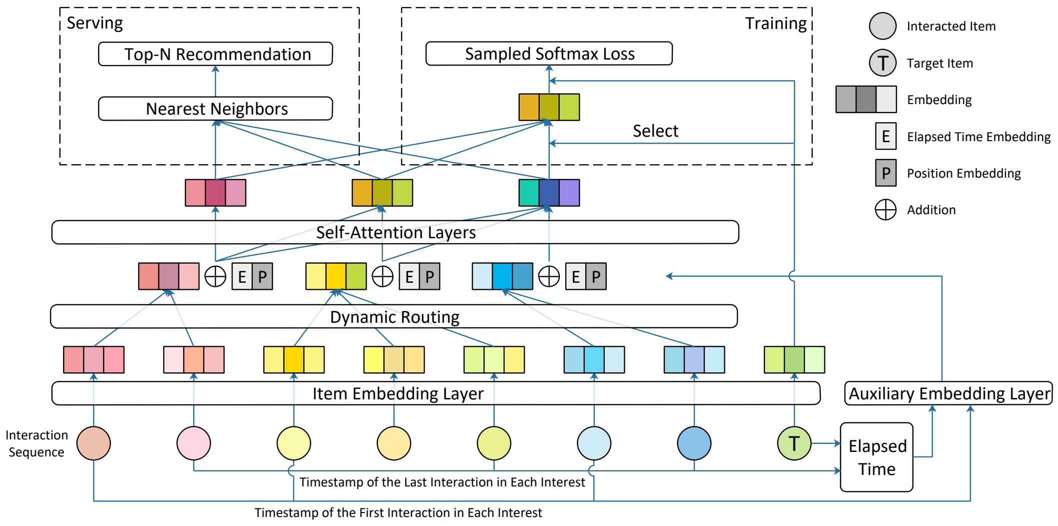


Fig. 2: An overview of DemiRec. DemiRec takes the user's interaction sequence as input. The embedding layer transforms item IDs into corresponding embeddings. Through the dynamic routing, unordered interests are extracted from item embeddings. Then, according to the timestamp of the first interaction and the last interaction in each interest cluster, which indicate the order and elapsed time respectively, position embedding and elapsed time embedding are added to each interest embedding for value and key in self-attention layers. The evolved interest embeddings generated through the self-attention layers are then used for model training and serving. For model training, the most similar interest embedding to the target embedding will be chosen to compute the sampled softmax loss. For serving, each interest embedding will independently retrieve top-N nearest items, generating the overall top-N recommendation.

Table 1: Notations

Notation	Description
u	a user
i	an item
t	a timestamp
d	the dimension of item embeddings
$\mathcal{U}$	the set of users
$\mathcal{I}$	the set of items
$\mathcal{T}$	the set of timestamps
$\mathcal{V}$	the set of interests
n	the maximum sequence length
k	the number of interest embeddings

$\mathcal{T}^u = \left\{t_1^u, t_2^u, \ldots, t_{|\mathcal{S}^u|}^u\right\}$. The goal of sequential recommendation task is to provide a recommendation list for each user u based on $\mathcal{S}^u$, and the real next interacted item $i_{|\mathcal{S}^u|+1}^u \in \mathcal{I}$ should be ranked as high as possible. Notations are summarized in Table 1.

3.2 Interest Sequence Extractor

3.2.1 Item Embedding Layer

First, we transform the user's historical interaction sequence into a fixed-length item sequence $\mathcal{I}^u = \left\{i_1^u, i_2^u, \ldots, i_n^u\right\}$ and timestamp sequence $\mathcal{T}^u = \left\{t_1^u, t_2^u, \ldots, t_n^u\right\}$ by clipping the earlier interactions or adding paddings. The interest sequence extractor layer takes item IDs sequence as the input. Then we adopt the widely-used embedding technology to embed these ID features into low-dimensional dense vectors. Therefore, the item embedding matrix is $\mathbf{M}^I \in \mathbb{R}^{|I|\times d}$. For item i in set $\mathcal{I}$, the embedding layer looks up and outputs the corresponding embedding e_i.

3.2.2 Dynamic Routing

We utilize the dynamic routing method based on MIND [22] to extract user's diverse interests. The item embeddings from user can be viewed as primary capsules. Through iterative dynamic routing, the outputs of interest capsules $v_j \in \mathbb{R}^d$ are determined by the weighted sum of primary capsules according to the coupling coefficients. The orientation and length of the output vector represent the user's specific interest and the probability that the interest exists, respectively.

3.2.3 Restoring Interests Sequence

After dynamic routing, the user's interactions are clustered into unordered interests according to the coupling coefficients. Then we restore the order and time information of the user's interests according to the item timestamps in the clusters. First, we find the subordination between items and interests according to the final coupling coefficients. If the coupling coefficient of item i to interest j is greater than the average coefficient of this interest, the item i is considered to belong to the interest j. Thus, the time information of interests can be obtained by the timestamp of the earliest and the latest interactions in each interest:

$$t_j = \min_{c_{ij}\geq\overline{c_j}} t_i \quad \text{and} \quad \hat{t}_j = \max_{c_{ij}\geq\overline{c_j}} t_i \tag{2}$$

where t_j are the start time of interest j, $\hat{t}_j$ are the latest interaction time of interest j, c_{ij} are the coupling coefficients. Then according to the start time of interests, we can restore the order of the user's interests $\mathcal{V}^u = \left\{v_1^u, v_2^u, \ldots, v_k^u\right\}$ and timestamps indicated the latest interaction time of corresponding interest $\mathcal{L}^u = \left\{\hat{t}_1^u, \hat{t}_2^u, \ldots, \hat{t}_k^u\right\}$.

3.3 Interests Sequence Modeling

3.3.1 Elapsed Time Modeling

We model the elapsed time since the latest interaction of the user's interests with the target interaction to find the relationship between the user's historical interests and the user's current interests. Generally, interest changes usually occurs after a long time interval, but the time interval of access is extremly diverse, leading to the data sparsity problem. Therefore, we use logarithm operation to map the elapsed times to a smaller range and make more data in a longer elapsed time encode. Specifically, giving the latest interaction time sequence $\mathcal{L}$ and a target interaction timestamp t_{n+1}, the scaled elapsed time of interest j is:

$$r_j = \log_{1+l}\left(t_{n+1} - \hat{t}_j\right) \quad (3)$$

where l is a hyperparameter of the base of logarithm operation. Tuning l can balance the precise encoding of each timestamp and alleviating data sparsity problem.

3.3.2 Auxiliary Embedding Layer

The self-attention model doesn't use the recurrent structure, it's not aware of the order of items in a sequence. Therefore following [32], we use two distinct learnable positional embedding matrices $\mathbf{M}_K^P \in \mathbb{R}^{k\times d}$, $\mathbf{M}_V^P \in \mathbb{R}^{k\times d}$ for keys and values in the self-attention mechanism, respectively. For interest v_i in the user interest sequence $\mathcal{V}$, the auxiliary embedding layer looks up and outputs the embedding p_i^k and p_i^v. Compare with the one learnable position embedding injected into the sequence, this method is more suitable and flexible for use in the self-attention mechanism without requiring additional linear transformations.

Similar to the positional embedding, elapsed time embedding matrices are $\mathbf{M}_K^R \in \mathbb{R}^{p\times d}$ and $\mathbf{M}_V^R \in \mathbb{R}^{p\times d}$ for keys and values in self-attention. For elapsed time encoding r_i in the timestamp sequence $\mathcal{L}$, the auxiliary embedding layer looks up and outputs the embedding r_i^k and r_i^v.

3.3.3 Elapsed Time-Aware Self-Attention

Based on Li, et al . [23], we apply the self-attention mechanism on interest sequence to predict users' interest changes. We add position embeddings and elapsed time embeddings to the keys and values in the self-attention calculation so that the order and elapsed time information can be considered:

$$\mathbf{K}_j = v_j W^K + r_j^k + p_j^k \quad \text{and} \quad \mathbf{V}_j = v_j W^V + r_j^v + p_j^v \quad (4)$$

Consider the causality in users' interest sequences, we forbade links between $\mathbf{Q}_i$ and $\mathbf{K}_j$ $(j > i)$, where $\mathbf{Q}_i = v_i W^Q, \mathbf{K}_j = v_j W^K + r_j^k + p_j^k$, so that only the first t interests will be considered when predicting the $(t+1)$st interest.

After the self-attention layer, a point-wise feed-forward network is applied identically to all z_i with sharing parameters to endow the model with nonlinearity and consider the interactions between different dimensions:

$$\mathrm{FFN}(z_i) = \mathrm{ReLU}\left(z_i \mathbf{W}^1 + \mathbf{b}^1\right)\mathbf{W}^2 + \mathbf{b}^2 \quad (5)$$

where $\mathbf{W}^1$, $\mathbf{W}^2$ are $d \times d$ matrices and $\mathbf{b}^1$, $\mathbf{b}^2$ are d dimensional vectors. The output interests from are then formed as a matrix $\mathbf{Z}_u = [z_1, z_2, \ldots, z_k] \in \mathbb{R}^{d\times k}$for downstream task.

3.4 Prediction

3.4.1 Model Training

After stacking self-attention layers and feed-forward layers, we get the prediction of the user's evolved interests according to the item clusters, sequence information and elapsed time. Then, we use an argmax operator to choose a corresponding interest embedding vector for the target item i:

$$\mathbf{z}_u = \mathbf{Z}_u\left[\mathrm{argmax}\left(\mathbf{Z}_u^{\mathrm{T}}\mathbf{e}_i\right)\right] \quad (6)$$

where e_i denotes the embedding of the target item i, and Z_u is the matrix formed by user interest embeddings. Then given a training sample (u, i) with the corresponding user interest embedding z_u and item embedding e_i, we can compute the probability of the user u interacting with the item i as:

$$P(i \mid u) = \frac{\exp\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{e}_i\right)}{\sum_{j\in\mathcal{I}} \exp\left(\mathbf{z}_u^{\mathrm{T}}\mathbf{e}_j\right)} \quad (7)$$

The sum operator of Equation 7 is computationally expensive. Therefore, we use a sampled softmax technique [16] to compute the approximate probability in training.

3.4.2 Serving

For online serving, we use the multiple interest embeddings processed by the multi-interest evolution model for each user. Through the approximate nearest neighbour approach, such as Faiss [17], each interest embedding can independently retrieve top-N items from the item pool with the highest similarities. Then, the items retrieved by multiple interests are ranked to determine the final overall recommend items for each user.

A common and straightforward way for ranking retrieved items is to use their inner production proximity with user interests. However, to achieve high accuracy, based on the sequence information, the multi-interest evolution model modified interest embeddings not only on direction but also on length, make interest representations got various strengths. This makes the simple use of proximity ignore some of the user's interests, make the recommendation results single, and damage the diversity. Although it is an effective method to maximize accuracy, diversity and novelty recommendation also contribute to improving the user experience. Therefore, follow the ComiRec [5], we use the following value function $Q(u, \mathcal{R})$ to balance the accuracy and diversity of the recommendation by a controllable factor $\lambda \geq 0$:

$$Q(u, \mathcal{R}) = \sum_{i\in\mathcal{R}} f(u, i) + \lambda \sum_{i\in\mathcal{R}} \sum_{j\in\mathcal{R}} g(i, j) \quad (8)$$

$$f(u, i) = \max_{1\leq k\leq K}\left(\mathbf{e}_i^{\mathrm{T}}\mathbf{z}_u^{(k)}\right) \quad (9)$$

$$g(i, j) = \delta(\mathrm{CATE}(i) \neq \mathrm{CATE}(j)) \quad (10)$$

where $\mathcal{R}$ is a set of items for final recommendation. Function $f(u, i)$ computes the maximum inner production proximity of the k-th interest embedding of the user u and item i, and function $g(i, j)$ is an indicator function. By adjusting λ from 0 to ∞, we can switch from only use the straightforward method to obtain the high accuracy to only consider the category diversity.

4. Experiments

In this section, we first introduce the benchmark datasets and our experimental settings. And then, we present the comparisons between DemiRec and other state-of-the-art methods to evaluate recommendation performance and in-depth analysis to verify the effectiveness of each component.

4.1 Datasets

We evaluate our methods on three large datasets from real world platforms. The statistics of the three datasets are shown in Table 2.

- **Amazon Books:** *Amazon* is a series of datasets introduced in [12, 27]. It comprises a large number of product reviews

Table 2: Statistics of datasets

Dataset	# users	# items	# interactions
Amazon Books	459,133	313,966	8,898,041
Steam	2,567,538	15,474	7,793,069
MovieLens-1M	6,040	3,416	999,611

of various category crawled from Amazon.com. We use the *Books* category of the *Amazon* dataset, which include 8.9 million reviews spanning May 1996 to July 2014.

- **Steam:** A dataset from a large online video game distribution platform introduced in [32], including 7.8 million reviews spanning October 2010 to January 2018. The dataset also includes rich information about user and games.
- **MovieLens-1M:** *MovieLens* [10] is a widely used benchmark for evaluating recommendation algorithms. We use the *MovieLens-1M*, which includes 1 million ratings on movies.

For all datasets, we determine the sequence order of implicit feedback by timestamps. Following the preprocessing procedure from [5], we partition all users in each dataset into training, validation and test set by the proportion of 8:1:1. To avoid data sparsity, we filter out the users and items with fewer than five interactions in our experiment. We train models using the entire interaction sequences of training users. In the evaluation, the first 80% interactions of the user from validation and test set are used to infer user embeddings from trained models, and the remaining 20% interactions are treated as target items to compute metrics.

4.2 Evaluation Metrics

We adopt two commonly used Top-N metrics, Recall and Normalized Discounted Cumulative Gain (NDCG) to compare the accuracy of models. Recall is obtained by dividing the number of corrected recommended items by the total number of all target items refers to the ratio of the real interacted items presenting in the top-N recommendation lists [19]. NDCG takes the exact ranking positions of correct recommended items in the list into consideration [15]. The recommend list length N is set to 20, 50 respectively as metrics for evaluation. For better interpretability, we adopt a per-user average for each metric [5, 19].

4.3 Competitors

We compare our proposed model, DemiRec, with the following methods. These methods include classic general recommendation (e.g. POP) without user modeling, Neural Network based methods (e.g. YouTube DNN, GRU4Rec, SASRec), and Capsule Network based methods (e.g. MIND, ComiRec). In our experimental setting, models should give the prediction for unseen users in the validation and test sets. Thus factorization-based methods are inappropriate for this setting.

- **POP:** A simple baseline that ranks all items according to their popularity in all users of training set, and the popularity is calculated by counting the number of interactions.
- **GRU4Rec [13]:** GRU4Rec use GRU unit in RNN to model to leverage complete interaction sequences.
- **SASRec [18]:** Using the self-attention mechanism to model user interaction sequences, SASRec can flexibly leverage the information of different parts in the sequence and achieve state-of-the-art results on sequential recommendation.
- **YouTube DNN [8]:** One of the most successful industrial recommender systems using a deep candidate generation network and a deep ranking network to complete the recommendation from a large-scale candidate set.
- **MIND [22]:** MIND is the industrial applicable recommendation model that first used Capsule Network to cluster users' interactions and extract diverse interests.
- **ComiRec [5]:** ComiRec studied to capture user's multiple interest by Capsule Network and Self-Attention, respectively and considered order information of user sequences.

For fair comparison, we consider embedding dimensions d from $\{16, 32, 48, 64\}$ and learning rate in $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and the default settings according to the respective papers for all methods except POP. For multi-interests methods, we set the interest number to 4. We tune hyper-parameters on the validation set using early stopping based on the Recall@50.

Due to the reproducibility problem caused by the lack of model details, we can not compare with FAT [26]. But through comparison with the baselines, we believe DemiRec has achieved competitive performance improvements.

4.4 Implementation Details

We implement DemiRec with tensorflow [1] 1.14 in Python 3.7 and fine-tune hyperparameters on the validation set. We set the average length of user sequences in each dataset as the max sequence length. We use two elapsed time-aware self-attention layers with the base of logarithm operation $l = 0.5$. We use Adam optimizer [20] with learning rate $lr = 0.0005$. The rest of our parameter settings can be found in [5]. To obtain stable results, we use the k-means++ [2] initialization method for dynamic routing in the experiments.

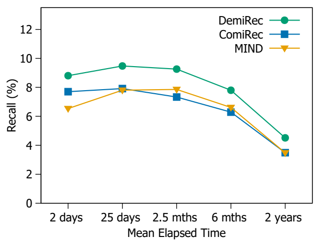
4.5 Recommendation Performance

Table 3 presents the recommendation performance of all methods on the three datasets. Among the baselines, SASRec outperforms other baselines on the most criteria. On the other hand, the best Capsule Network-based baseline can only achieve higher performance on capturing multiple long-term interests from Amazon Books users. By combining the advantages of capsule network and sequence modelling, our model DemiRec achieved significant improvement compare with Capsule Network-based baselines and outperform the best baseline over Amazon Books and Steam, which shows the effectiveness of leverage elapsed time and sequence information on interest level. For the dense MovieLens-1M, compared with SASRec, the clustering process makes DemiRec unable to fully use the information in the sequences. But a little performance sacrifice leads to a significant increase in diversity which is discussed in Section4.8.

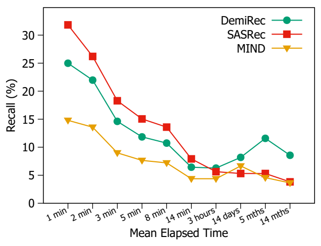
Figure 3 shows the recall of DemiRec, the best baseline and MIND (the base model which uses the raw interests) on Amazon Books and MovieLens datasets. The test samples are sorted and grouped according to the elapsed time. From the figures, we can find that the baselines degrade seriously for the samples with long elapsed time since they only model the historical information. On the contrary, DemiRec can maintain relatively high performance

Table 3: Recommendation performance on public datasets. The best performance result in each column is boldfaced, and the second-best performance result in each column is underlined. Improvements over the best baseline are shown in the last row. All the numbers in the table are percentage numbers with '%' omitted.

	Amazon Books				Steam				MovieLens-1M			
Model	Metrics@20		Metrics@50		Metrics@20		Metrics@50		Metrics@20		Metrics@50	
	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG	Recall	NDCG
POP	1.361	0.533	2.315	0.722	13.134	5.234	20.407	6.670	4.903	1.639	10.675	2.771
GRU4Rec	3.779	1.692	6.210	2.172	22.585	13.386	32.966	15.435	9.424	3.723	18.736	5.600
SASRec	5.274	2.900	8.210	2.322	23.860	14.106	34.322	16.171	**11.591**	**4.411**	**22.124**	**6.487**
YoutubeDNN	4.472	2.041	7.032	2.546	21.956	10.415	33.300	12.657	9.263	3.538	17.654	5.186
MIND	4.627	2.131	7.289	2.649	20.662	11.181	30.166	13.017	6.598	2.518	14.041	3.978
ComiRec	5.602	2.192	8.412	2.749	21.614	12.169	31.487	14.121	7.499	2.552	15.707	4.165
DemiRec	**6.232**	**2.957**	**9.392**	**3.581**	**25.581**	**14.454**	**36.738**	**16.660**	10.702	4.085	21.159	6.139



(a) Amazon Books



(b) MovieLens-1M

Fig. 3: Recall comparison between DemiRec, the best baseline and MIND on Amazon Books and MovieLens datasets grouped by elapsed time.

Table 4: Ablation analysis on three datasets.

	Books		Steam		MovieLens	
Metrics@50	Recall	NDCG	Recall	NDCG	Recall	NDCG
DemiRec/C	9.328	3.571	**37.091**	**17.457**	20.190	5.873
DemiRec/L	**9.836**	**3.868**	35.593	15.665	19.536	5.765
DemiRec/P	9.739	3.823	30.728	12.787	18.179	5.429
DemiRec/E	8.861	3.303	33.546	12.976	16.269	4.722
DemiRec	9.392	3.581	36.738	16.660	**21.159**	**6.139**

when users access the system after a long elapsed time.

4.6 Ablation Study

To study the effectiveness of considering sequence information on user's interests and the importance of elapsed time for squential recommendation, we analyze their impacts via an ablation study. Table 4 shows the performance comparison between our default method and following 4 variants on the three datasets:

- **DemiRec/C & DemiRec/L:** We unblind the links forbade in self-attention layers of DemiRec/C. This modification makes former interests influenced by future interests. To further dissect the effectiveness of considering influences between the user's interests, we turn off all links between different interests in the attention layers as DemiRec/L. The comparison results between these modifications and the default setting are different on each dataset. The result on Amazon Books shows that much of uses' interests are unaffected by other interests. In the Steam dataset, users' interests usually influence each other, whether old or new. The default setting gets the best performance on MovieLens. This result shows that some of the users' interests are affected by causal relationships, and it also benefits from dense information for each user. Overall, the links between users' interests should be established according to the actual application scenarios.
- **DemiRec/P:** We remove the position embeddings used in self-attention layers to trun off the sensitivity of the order of users' interest sequences. We denote this model as DemiRec/P. The comparison results with the default setting on the Steam and MovieLens datasets demonstrate the importance of using the order information of users' interests. DemiRec/P's result on Amazon Books outperforms the default setting, revealing that users' habits in books are different from those in games or movies, and users' interests tend to exist in parallel and long-term.
- **DemiRec/E:** In this modification, we remove the elapsed time embeddings to obtain interest predictions based on a default elapsed time learned through training. No matter how long it passed since the user's last access, DemiRec/E will give same recommendation. This simple prediction strategy makes DemiRec/E underperform the default setting in every dataset. This result verifies that the elapsed time is critical for predicting users' evolving interests.

Table 5: Effect of the interest number k in the dynamic routing.

	Books		Steam		MovieLens	
Metrics@50	Recall	NDCG	Recall	NDCG	Recall	NDCG
DemiRec(k=2)	**9.459**	**3.608**	**36.892**	15.962	19.399	5.618
DemiRec(k=4)	9.392	3.581	36.738	16.660	**21.159**	**6.139**
DemiRec(k=6)	9.341	3.604	36.374	**16.999**	20.651	6.083
DemiRec(k=8)	9.248	3.528	33.771	15.752	20.224	5.833

4.7 Analysis of Hyperparameters

4.7.1 Number of Interests:

Table 5 shows the performance of DemiRec with the number of interests k from 2 to 8. The optimal setting of the interest number depends on users' habits under different application scenarios. For the Amazon Books, DemiRec obtains the best performance when $k = 2$. For the Steam, DemiRec achieves similar performance when k is set to 2, 4 and 6. After verifying the recommendation results of each interest embedding, we find that users in Steam usually have 3 interests. When k is set to 6, DemiRec learned two identical clusters for each interest, resulting in a similar performance with $k = 2$ and $k = 4$. For the MovieLens, DemiRec obtains the best performance when $k = 4$.

4.7.2 Base of Logarithm Operation:

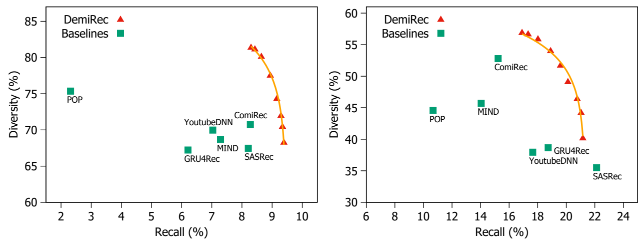
Table 6 shows the performance of DemiRec with the base of logarithm operation l in the elapsed time modeling from 2^{-2} to 2^2. The results demonstrate that the data sparsity of long elapsed time may damage the performance although a smaller value of l can give precise encoding for each timestamp. Conversely, a larger value of l will cause rough encoding for short elapsed time.

Table 6: Effect of the base of logarithm operation in the elapsed time modeling.

	Books		Steam		MovieLens	
Metrics@50	Recall	NDCG	Recall	NDCG	Recall	NDCG
DemiRec(l=2^{-2})	9.411	3.576	36.621	16.593	20.085	5.729
DemiRec(l=2^{-1})	9.392	3.581	**36.738**	**16.660**	**21.159**	**6.139**
DemiRec(l=2^{0})	**9.507**	**3.628**	36.497	16.607	20.073	5.637
DemiRec(l=2^{1})	9.407	3.555	36.517	16.434	19.909	5.651
DemiRec(l=2^{2})	9.358	3.537	36.438	16.150	20.081	5.912

Table 7: Accuracy and diversity performance for the controllable study.

	Books		MovieLens	
Metrics@50	Recall	Diversity	Recall	Diversity
DemiRec(λ = 0.0000)	**9.392**	68.299	**21.159**	40.207
DemiRec(λ = 0.0125)	9.344	70.485	21.039	44.232
DemiRec(λ = 0.0250)	9.293	71.994	20.767	46.441
DemiRec(λ = 0.0500)	9.157	74.334	20.118	49.128
DemiRec(λ = 0.1000)	8.941	77.557	19.601	51.750
DemiRec(λ = 0.2000)	8.644	**80.151**	18.905	**54.037**



(a) Amazon Books (b) MovieLens-1M

Fig. 4: Accuracy and diversity comparison between baselines and DemiRec with different the factor λ settings.

Thus, we need to tune l to balance both sides. The best l is 2^{-1} for Steam and MovieLens dataset, and 2^0 for Amazon Books, in our experimentations. This difference further verifies the experimental results in Section 4.6 that the users' interests in books tend to be long-term compared with those in games and movies.

4.8 Recommendation Diversity

The majority of recommendation algorithms have focused whole efforts on improving prediction accuracy. However, other important aspects of recommendation quality, such as diversity of recommendations, also play an important role to avoid monotony and improve user experience, which drawn more research attention in recent years [4,28]. Our method represents each user with multiple interests embeddings and expects more diverse results than methods using a single vector in user modeling. Thus, we analyze the diversity performance of our method with the following definition of individual diversity based on item categories:

$$\text{Diversity@N} = \frac{\sum_{j=1}^{N}\sum_{k=j+1}^{N}\delta\left(\text{CATE}\left(\hat{i}_{u,j}\right) \neq \text{CATE}\left(\hat{i}_{u,k}\right)\right)}{N \times (N-1)/2}, \tag{11}$$

where CATE(i) is the category of item i, $\hat{i}_{u,j}$ denotes the j-th recommended item for the user u, and $\delta(\cdot)$ is an indicator function.

4.8.1 Controllable Study:

To ensure the trade-off between accuracy and diversity in final top-N recommend items, we use the aggregation method proposed in ComiRec [5], which can control the balance of retrieving with the most intense interest or with diverse interests. Table 7 shows the model performance of DemiRec on the Amazon Books and MovieLens. By controlling the factor λ, we can balance the recommendation accuracy and diversity. From the table, we can see that within larger controllable factor λ, the recommendation diversity increases substantially by retrieving with more diverse user interests. Since we retrieve 50 items in this metric, high score items can be retained in the final recommendation list, resulting in a slight decrease in recall rate.

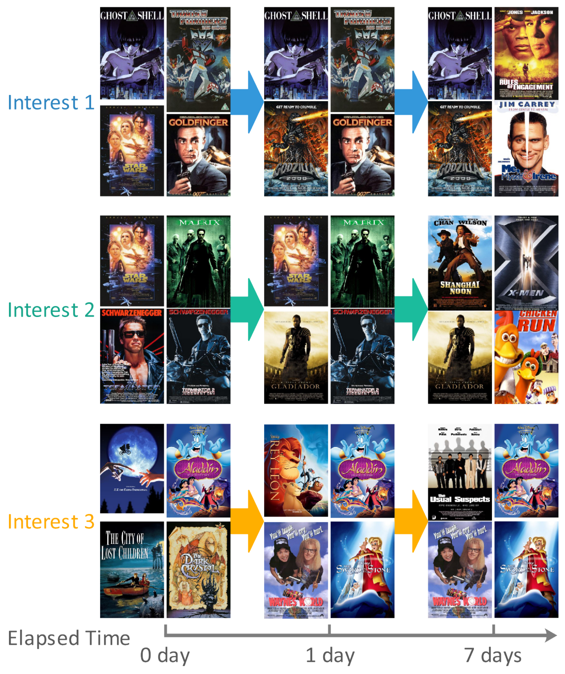


Fig. 5: Top 4 retrieved items of interests generated by DemiRec after different elapsed times.

4.8.2 Diversity Comparison:

Figure 4 compares the accuracy and diversity between baselines and DemiRec with different factor λ settings. The number of interests k is set to 4 for multi-interest models. The control factor λ is set to 0.05 for the aggregation method in ComiRec. From the Figure 4, we can find that our model outperform most baselines on both accuracy and diversity. For the comparison with SASRec on MovieLens-1M, a slight decrease in accuracy brings a significant improvement in diversity.

4.9 Case Study

Figure 5 shows the top 4 retrieved items of a randomly selected user after different elapsed times. We can find that our model learns three different fine-grained interests for the user. When the user accesses the system after different elapsed times, DemiRec gives different recommendations. As the elapsed time gets longer, interests are mixed together and novel items are recommended.

5. Conclusions

In this paper, we propose a dynamic evolutionary multi-interest network for the sequential recommendation to give evolved recommendation after different elapsed times. Our model uses an interest sequence extractor to generate users' interest sequences and models the sequential information and elapsed time on interest level to give dynamic evolved recommendations. Extensive empirical results demonstrate that our model significantly improves diversity with comparable accuracy performance, and outperforms the state-of-the-art multi-interest baseline in terms of both accuracy and diversity on three public benchmarks. We also explore various features of this model.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[2] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Keith Bradley and Barry Smyth. Improving recommendation diversity. In *Proceedings of the Twelfth Irish Conference on Artificial Intelligence and Cognitive Science, Maynooth, Ireland*, volume 85, pages 141–152. Citeseer, 2001.

[5] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2942–2951, 2020.

[6] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, pages 1–4, 2019.

[7] Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. Meantime: Mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 515–520, 2020.

[8] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *ACM transactions on interactive intelligent systems (Tiis)*, 5(4):1–19, 2015.

[11] Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200. IEEE, 2016.

[12] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[13] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[14] Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pages 44–51. Springer, 2011.

[15] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA, 2017.

[16] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

[17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 2019.

[18] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

[19] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254, 2001.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[21] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 447–456, 2009.

[22] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. Multi-interest network with dynamic routing for recommendation at tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2615–2623, 2019.

[23] Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 322–330, 2020.

[24] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[25] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1831–1839, 2018.

[26] Yujie Lu, Shengyu Zhang, Yingxuan Huang, Luyao Wang, Xinyao Yu, Zhou Zhao, and Fei Wu. Future-aware diverse trends framework for recommendation. *arXiv preprint arXiv:2011.00422*, 2020.

[27] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 43–52, 2015.

[28] Lijing Qin and Xiaoyan Zhu. Promoting diversity in recommendation by entropy regularizer. In *Twenty-Third International Joint Conference on Artificial Intelligence*. Citeseer, 2013.

[29] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 130–137, 2017.

[30] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820, 2010.

[31] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. *arXiv preprint arXiv:1710.09829*, 2017.

[32] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[33] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[35] Jason Weston, Ron J Weiss, and Hector Yee. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 65–68, 2013.

[36] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 495–503, 2017.

[37] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 346–353, 2019.

[38] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM international conference on data mining*, pages 211–222. SIAM, 2010.

[39] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.

[40] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 729–732, 2016.

[41] 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 *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 5941–5948, 2019.