2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00208

Sequential Recommendation on Dynamic Heterogeneous Information Network

Tao Xie [1], Yangjun Xu [1], Liang Chen [1*], Yang Liu [1], Zibin Zheng [1]
[1] *School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China*
xiet23@mail2.sysu.edu.cn, xyjlearner@gmail.com, chenliang6@mail.sysu.edu.cn,
liuy296@mail2.sysu.edu.cn, zhzibin@mail.sysu.edu.cn

***Abstract*—The sequential recommendation has been widely used to predict users' preferences in the near future by utilizing their dynamic interactions with items. However, existing methods only consider single-typed interactions (e.g., purchase), ignoring the rich heterogeneous information such as multi-typed interactions (e.g., click, purchase) and item attributes (e.g, category), which leads to a suboptimal model. We can integrate this rich information by introducing Dynamic Heterogeneous Information Networks (DHINs). Our solution contains three special designs: 1) *Static Initialization*; 2) *Heterogeneous User Memory Network*; 3) *Two-level attention mechanism*. Extensive experiments conducted on two real-world datasets show that our model outperforms other state-of-the-art solutions. Furthermore, we provide some insights into parameter settings and model interpretability.**

***Index Terms*—Heterogeneous Information Network; Sequential Recommendation; User Memory Network; Attention Mechanism**

I. Introduction

In the era of information overload, the recommender systems play an important role in many real-world applications [1]–[3]. Among the various recommendation methods, the sequential recommendation [4]–[11] stands out for its effectiveness in capturing dynamic information and modeling the tendency of user preferences based on the users' sequential historical behaviors. Factorized Personalized Markov Chains (FPMC) [4] factorizes the transition matrix over underlying MC to model personalized sequential behaviors, by combining the power of Markov Chain (MC) and Matrix Factorization (MF) [12] approaches. Fossil [5] integrates similarity-based methods with MC to make personalized sequential predictions on sparse and long-tailed datasets. The deep learning model DREAM [6] leverages RNN to capture global sequential patterns and learn representations of dynamic user interests, by compressing all of a user's previous records into a fixed hidden representation. There are many other works [7], [8] focusing on involving RNN variants for the session-based recommendation. While effective, the computation of RNN-based methods cannot be fully parallelized within a sequence, then Caser [9] and NextItNet [10] abandon the RNN structures and propose CNN-based solutions. Thus, many advanced sequential recommendation methods have attracted more and more attention recently. Despite prevalence and effectiveness, the existing sequential-based methods only consider single-typed interactions information, such as [9]–[11]. We argue that they are insufficient in modeling user interests since they ignore the rich heterogeneous information and the potential item relationships.

Take a specific sequential recommendation scenario as an example, as shown in Figure 1. A user who often clicks, stars and buys NBA products, happened to buy a smartphone. Then the user has clicked several phone cases but has not yet decided which to buy. There are three strategies to model this sequence (take the last four behaviors as an example): 1) while only considering the purchase behaviors, the behaviors sequence is: NBA, NBA, Phone, NBA, which tends to give NBA products or smartphones as recommendation results; 2) while treating all types of behaviors as the same type, the behaviors sequence is: NBA, Phonecase, Phonecase, Phonecase, which tends to give some phone cases and NBA products as recommendation results; 3) while distinguishing different types of behaviors, the behaviors sequence is the same as the second one, but with their corresponding types: purchase, click, click, click. This strategy can recommend not only NBA products but also NBA-themed phone cases.

Therefore, an accurate sequential-based recommender system should discriminately consider the multi-typed dynamic interactions (e.g, click, star, purchase) and leverages the potential item relationships. It is of critical importance to develop a solution that can fully exploit this rich heterogeneous information in an effective and end-to-end manner. The Dynamic Heterogeneous Information Network (DHIN) incorporates dynamic information with static heterogeneous information. We propose a novel framework called Dynamic Heterogeneous Information Memory Network (DHIMN) based on DHINs. Our framework is equipped with three key techniques to model dynamic heterogeneous information and static attributes, including: 1) *Static Initialization*, which learns initial embeddings from static attributes; 2) *Heterogeneous User Memory Network*, which extends the traditional user memory network to handle the multi-typed and dynamic interactions ; 3) *Two-level Attention Mechanism*, which models the user preferences for different behavior types as well as the potential item relationships. In summary, the major contributions of our work can be concluded as follows:

- We highlight the importance of considering the static heterogeneous and multi-typed dynamic information into the recommendation.
- We propose a novel sequential recommendation model

*Contact Author

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00208

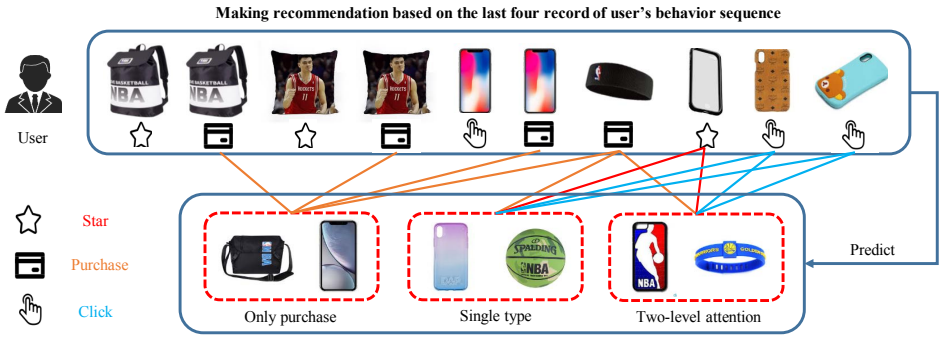

Fig. 1. Three different strategies to model multi-typed behaviors in above sequence: 1) only considering the purchase behaviors; 2) treating all types of behaviors as the same type; 3) based on our two-level attention.

DHIMN, which models the multi-typed behaviors and potential item relationships in an end-to-end manner.

- We conduct extensive experiments to demonstrate the effectiveness of DHIMN and provide some insights into parameter settings and model interpretability.

II. Methods

In this section, we will first define the research problem, and introduce the general framework of the proposed model DHIMN, then discuss the three composed modules in detail.

A. *Problem Definition*

Given a dynamic heterogeneous information network $G = (V, E)$, our task is to predict the next possible buying item among the candidate items for each user u, based on his/her historical behavior sequence. Suppose we have m users $U = \{u_1, u_2, ..., u_m\} \in V$, n items $I = \{i_1, i_2, ..., i_n\} \in V$ and l types of interactions $T = \{t_1, t_2, ..., t_l\} \in E$.

Our major challenges are:

1) How to leverage the dynamic information to perform sequential recommendations, while existing HIN-based works only considering the static information;

2) How to better modeling the user preferences for different behavior types as well as the potential relationships among items from the historical behavior sequence.

B. Recommendation Framework

Our recommendation framework consists of three main components: static initialization, heterogeneous user memory network, and two-level attention mechanism.

*1) **Static Initialization**:* The static attributes help better understand user preferences. In Figure 1, the NBA-themed phone case meets the user's taste better and tends to rank higher in the recommended list, since it belongs to both the categories of NBA-products and phone cases. To leverage this static information in DHINs, we provide an optional step for embedding initialization that builds upon a GCN-based message-passing layer [13], [14], which obtains the initial embeddings by propagating and refining embeddings from the static attributes.

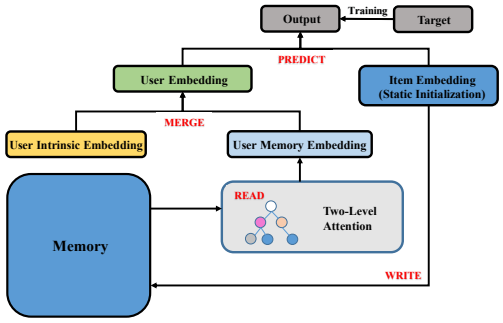

Fig. 2. The general recommendation framework of DHIMN.

*2) **Heterogeneous User Memory Network**:* Traditional recommendation methods based on the user memory network, like RUM [11], cannot handle multi-typed interactions, because they only consider single-typed interactions (i.e., each user has only one memory matrix). Therefore, we expand the dimensions of the memory matrix for each user. Each dimension of the memory matrix has a corresponding interaction type, as shown in Figure 3.

To calculate the predicted scores, we pass the inner product of current user embedding p_u and current item embedding q_i to the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.

$$\hat{y}_{ui} = PREDICT(p_u, q_i) = \sigma(p_u^T \cdot q_i) \tag{1}$$

The sigmoid function transforms the output into the interval of [0,1], so we choose binary cross-entropy as our loss function. Let I_u^+ and I_u^- be the positive and negative samples corresponding to user u, respectively. And the regularization term $\lambda\|\theta\|_F^2$ is added to prevent the model from overfitting, where $\|\cdot\|_F$ refers to the F norm. The loss function to be minimized can be defined as follows:

$$\begin{aligned} Loss &= -log\prod_{u,i}(\hat{y}_{ui})^{y_{ui}}(1-\hat{y}_{ui})^{1-y_{ui}} + \lambda\|\theta\|_F^2 \\ &= -(\sum_u \sum_{i\in I_u^+} log\hat{y}_{ui} + \sum_u \sum_{i\in I_u^-} log\hat{y}_{ui}) + \lambda\|\theta\|_F^2 \end{aligned} \tag{2}$$

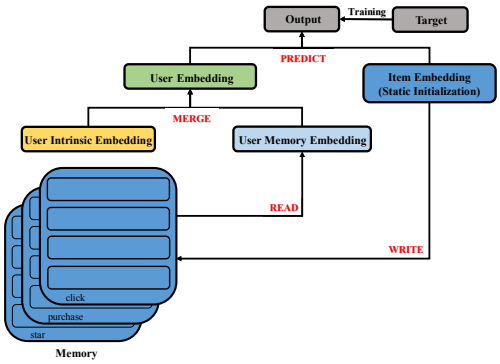

Fig. 3. Illustration of the heterogeneous user memory network.

Specifically, let p_u^* denote the user intrinsic embedding and q_i denote the item embedding. They are both randomly initialized and updated by the backpropagation. Finally, we obtain the memory enhanced user embedding p_u by merging p_u^* with memory embedding p_u^m with weighted addition. This helps capture both intrinsic and dynamic features. The parameter α_1 is used to control the contribution of p_u^m to p_u:

$$p_u = \mathit{MERGE}(p_u^*, p_u^m) = p_u^* + \alpha_1 p_u^m \tag{3}$$

The p_u^m can be derived from each user's personal memory matrix $M_u \in R^{l \times d \times s}$, where d is the embedding size, and s is the number of memory slots. Let $M_{u,t} \in R^{d \times s}$ denote the user u's memory of a certain type t, and the corresponding memory slot $m_{u,t}^k \in R^d$ be the column of $M_{u,t}$. Each memory slot stores the dynamic features learned from the historical items. At first, we have tried to use a FIFO (First In First Out) mechanism to update the memory by replacing the oldest slot with the current item embedding q_i in each round of training. However, it led to an unfavorable performance. The main reason might be that the limited number of memory slots makes it unable to capture the long-term information of the historical sequence. Therefore, we tend to use a similar global feature-based method inspired by feature-level RUM [11]. We use a global features $F \in R^{l \times s \times d}$ to capture the long-term information of entire sequence, and $f_k \in R^{l \times d}$ refers to the k-th global feature vector.

Reading operation. An attention mechanism is adopted to give the relevance score between q_i and f_k, which can be referred as attention weights on different memory features:

$$w_{ik} = q_i^T \cdot f_k \tag{4}$$

$$z_{ik} = \frac{\exp(w_{ik})}{\sum_j \exp(w_{ij})} \tag{5}$$

$$p_{1,u}^m = \sum_{k=1}^{s} z_{ik} \cdot m_u^k \tag{6}$$

Writing operation. We update $M_{u,t}$ for the user u according to the current inputted record $< u, i, t >$. Since there may be some noise records in the entire sequence, the memory should have the ability to forget. Therefore, we adopt the erase-add mechanism for memory updating, which first erases the memory before adding new contents:

$$\mathit{erase}_i = \sigma(q_i) \tag{7}$$

$$m_{u,t}^k \leftarrow m_{u,t}^k \odot (1 - z_{ik}^t \cdot \mathit{erase}_i) \tag{8}$$

$$\mathit{add}_i = tanh(q_i) \tag{9}$$

$$m_{u,t}^k \leftarrow m_{u,t}^k + z_{ik}^t \cdot \mathit{add}_i \tag{10}$$

Where $\mathbf{1}$ is a column vector of all 1's, and z_{ik}^t is the soft attention weight calculated during the reading process. According to Eq(8), the element of a memory slot will be reset to zero only if the corresponding z_{ik}^t and erase_i are both 1, or remain unchanged when either of them is 0. The model extracts key features of q_i while erasing, and updates them into the memory according to the Eq(10). In this way, the erase-add mechanism automatically determines which signals should be weakened and which to be strengthened.

*3) **Two-level Attention Mechanism**:* Our proposed method not only considers the multiple-typed interactions, but also leverages a two-level attention mechanism to model user preferences for different behavior types as well as the potential item relationships.

As the previous discussion, the memory feature embedding $p_{1,u}^m$ is derived from the relevance score between q_i and f_k. Inspired by the works on attention [15], [16], we define another way to obtain memory embedding (here denotes as $p_{2,u}^m$):

$$p_{2,u}^m = \sum_{t \in T} \beta_1(u,t) \cdot s_{u,i}^t \tag{11}$$

The first-level attention learns the user's preferences for different behavior types. It distinguishes the contribution of different behavior types when calculating $p_{2,u}^m$. The $\beta_1(u,t)$ is the attention weight of user u on a certain type t:

$$\beta_1(u,t) = \frac{\exp(a(p_u)^*, x_t)}{\sum_{t' \in T} \exp(a(p_u^*, x_{t'}))} \tag{12}$$

$$a(p_u^*, x_t) = h_1^T(\mathit{ReLU}(W_1(p_u^* \odot x_t) + b_1) \tag{13}$$

The scoring function Eq(13) aims to obtain the attention score $a(p_u^*, x_t)$, which gives the relevance score between p_u^* and x_t, where x_t is the embedding of a certain type t. Finally, the attention weights of user u on the interaction type t can be derived from a softmax operation Eq(12) on attention score.

The second-level attention learns the potential relationships between current item and historical items (considered as memory slots). It gives the relevance score between q_i and $m_{u,t}$. The $s_{u,i}^t$ in the Eq(11) describes the user's profile based on his/her dynamic memory, which can be derived from the below equations:

$$s_{u,i}^t = \sum_{k=1}^{s} \beta_{2,t}(i,k) \cdot m_{u,t}^k \tag{14}$$

$$\beta_{2,t}(i,k) = \frac{\exp(b_t(q_i, m_{u,t}^k))}{\sum_{k'=1}^{s} \exp(b_t(q_i, m_{u,t}^{k'}))} \tag{15}$$

$$b_t(q_i, m_{u,t}^k) = h_{2,t}^T(\textit{ReLU}(W_{2,t}(q_i \odot m_{u,t}^k) + b_{2,t})) \tag{16}$$

Here $\beta_{2,t}(i,k)$ is the attention weights of current item embedding q_i on the k-th memory slot m_u^k, and the calculation is similar with $\beta_1(u,t)$. The $m_{u,t}^k$ represents a certain dynamic features of the user's memory according to previous discussion. Finally, with all $s_{u,i}^t$ and $\beta_1(u,t)$ calculated, we can obtain the memory embedding $p_{2,u}^m$, according to Eq(11).

$$p_u^m = \textit{MERGE'}(p_{1,u}^m, p_{2,u}^m) = p_{1,u}^m + \alpha_2 p_{2,u}^m \tag{17}$$

Then we use another merge funtion Eq(17) to get the final memory embedding p_u^m, where α_2 controls the contribution of $p_{2,u}^m$ to p_u^m. Finally, the p_u^m is returned to the previous module for further optimization, according Eq(3).

C. Complexity Analysis

The complexity of our model is mainly composed of two parts: memory reading and memory writing. In each round of train, let $T(read)$ and $T(write)$ denote the number of operations when memory reading and writing. And $T(other)$ for other parts including *MERGE*, *MERGE'* and *PREDICT*. Since the $T(other)$ is much smaller than $T(read)$ and $T(write)$, we will not further discuss the $T(other)$. There are m users in our scenario. Suppose each user has $\bar{n}$ interactions, where $\bar{n}$ is an average number of interactions, including both positive and negative interactions. Then the total number of operations in each round of training is:

$$T = m \times \bar{n}((T(read) + T(write) + T(other)) \tag{18}$$

Reading Operation. It mainly contains two parts: 1) the reading operation of *Heterogeneous User Memory Network*. The number of operations of Eq(4), Eq(5) and Eq(6) is $ls(2d-1)$, $4ls$, $lsd + s - 1$, respectively. 2) *Two-level Attention*. The number of operations of Eq(11) is $l(2d-1)$. Note that each scoring function contains two fully-connected layers, while the first one has a bias as well as activation function and the second one does not. Assuming that the first fully-connected layer has a output neurons, and we do not consider the operation of the activation function. The number of operations of the first-level and second-level attention is $ld + 2lda + 2la + 3l$, $l(sd + 2sda + 2sa + 3s)$. Approximately, the total operations of memory reading are:

$$\begin{aligned} T(read) \approx\ & 3ls(d+1) + s - 1 + l(2d-1) \\ & + (s+1)(ld + 2lda + 2la + 3l) \end{aligned} \tag{19}$$

Writing Operation. The writing process is relatively simple. Suppose each sigmoid function has 4 operations, and each tanh function has 5 operations. Then the number of operations of the erase part is $4d + 3sd$, while the add part is $5d + 2sd$. Besides, updating of $m_{u,t}$ needs another sd operations, so the total number of operations of the memory writing is:

TABLE I
DATASET STATISTICS ACCORDING TO EACH TYPES

Dataset	Type	Users	Items	Interactions
Jingdong	Click	2351	136440	386371
	Purchase	2351	21392	27829
Taobao	Click	1887	76418	253690
	Favorite	1887	6892	7719
	Cart	1887	7740	10053
	Purchase	1887	5160	6251

$$T(write) = 9d + 6sd \tag{20}$$

Although $T(read)$ seems complicated, the overall number of operations is not that high as it seems. It is because the values of l, d, s, and a are actually small, which are much smaller than the number of users and items.

III. EXPERIMENTS AND RESULTS

In this section, we conduct experiments on two real-world datasets to evaluate our method and give some discussions.

A. Dataset

We conducted experiments on two real-world datasets. One is the Jingdong dataset[1], and the other is the Taobao dataset[2]. The raw datasets contain lots of irrelevant records and noises. We conducted several steps to preprocess these two datasets: 1) Group all interactions by each user, and sort them by timestamp; 2) Only keep those users who have all types of interactions; 3) Remove the users with too few or too many interactions; 4) Since the number of items is far larger than the user's, we finally select about 2000 users and the corresponding items; 5) Split each user's last purchase record into the test set and the rest into the training set; The statistics of filtered datasets are shown in Table I.

B. Experiment Settings

*1) **Evaluation Protocol**:* We adopt the leave-one-out protocol for all models, which has been widely used in evaluation [16], [17]. For each user, we randomly select 999 negative samples and treat the last purchase record as a positive sample. We adopt Hit Ratio (HR) and Normalized Discounted Cumulative Gain (NDCG) as evaluation metrics.

*2) **Baselines**:* In order to make a comprehensive comparison from multiple angles, several baselines are selected from traditional recommendation, sequential recommendation and heterogeneous recommendation. They are BPR [18], NeuMF [17], Caser [9], NextItNet [10], RUM [11], KGAT [19].

*3) **Parameter Settings**:* In general, we randomly initialize all the model parameters and use an SGD optimizer to update them. Note that the item embeddings can be initialized with static information embeddings. For the hyperparameters, the slots number s is set to 20, the embedding size d is set to 50, the output neurons number a is set to 32, and the weighting parameter α_1 and α_2 are both set to 0.2. Finally, our model recommends top-5 or top-10 candidate items for each user.

[1]https://github.com/guyulongcs/IJCAI2019_HGAT
[2]https://tianchi.aliyun.com/competition/entrance/231522/information

TABLE II
TOP-K RECOMMENDATION PERFORMANCE COMPARISON OF ALL METHODS. THE IMPROV. DENOTES THE RELATIVE IMPROVEMENT OF OUR PROPOSED MODEL (DHIMN) OVER THE STRONGEST BASELINE.

Dataset	Metric	Single-typed Models					Multi-typed Models				
		Traditional		Sequential			Heterogeneous	Proposed (DHIMN)			
		BPR	NeuMF	Caser	NextItNet	RUM	KGAT	Without two-level	With two-level	With two-level (static init)	Improv.
Jingdong	HR@5	0.152	0.139	0.069	0.076	0.124	*0.561*	0.743	**0.781**	0.772	39.2%
	NDCG@5	0.140	0.127	0.054	0.053	0.091	*0.438*	0.695	**0.704**	0.687	60.7%
	HR@10	0.159	0.150	0.087	0.089	0.145	*0.648*	0.773	**0.810**	0.810	25.0%
	NDCG@10	0.142	0.126	0.056	0.054	0.093	*0.459*	0.705	**0.714**	0.699	55.6%
Taobao	HR@5	0.122	0.127	0.068	0.065	0.139	*0.783*	0.940	0.949	**0.956**	22.1%
	NDCG@5	0.110	0.111	0.048	0.043	0.114	*0.645*	**0.897**	0.893	0.895	39.0%
	HR@10	0.132	0.147	0.077	0.071	0.148	*0.836*	0.952	0.966	**0.968**	15.8%
	NDCG@10	0.114	0.119	0.050	0.049	0.115	*0.644*	**0.901**	0.899	0.899	39.9%

TABLE III
SAMPLE SIZE ON TRAINING SET AND TEST SET OF SINGLE-TYPED MODELS AND MULTI-TYPED MODELS.

Models	Dateset	Positive Samples	Negative Samples
Single-typed	Train	Purchase	Never Purchase
	Test	Purchase	Never Purchase
Multi-typed	Train	Interact	Never Interact
	Test	Purchase	Never Purchase

TABLE IV
COMPARISON BETWEEN OUR PROPROSED MODELS AND SINGLE-TYPE MODELS (RUM) WHILE USING MULTI-TYPE RELATIONS INFORMATION

Dataset	Metric	Multi-type Information					
		Sequential			Proposed (DHIMN)		
		Caser	NextItNet	RUM	With two-level	With two-level (static init)	Improv.
Jingdong	HR@5	0.295	0.107	*0.734*	**0.781**	0.772	6.4%
	NDCG@5	0.272	0.094	*0.687*	**0.704**	0.687	2.5%
	HR@10	0.313	0.125	*0.761*	**0.810**	0.810	6.4%
	NDCG@10	0.278	0.100	*0.696*	**0.714**	0.699	2.6%
Taobao	HR@5	0.795	0.165	*0.937*	0.949	**0.956**	2.0%
	NDCG@5	0.772	0.146	**0.898**	0.893	0.895	-
	HR@10	0.801	0.182	*0.951*	0.966	**0.968**	1.8%
	NDCG@10	0.773	0.152	**0.906**	0.899	0.899	-

C. Performance

We conducted experiments on both single-typed and multi-typed interactions to show their different impacts on performance. Note that we evaluate both single-typed models[3] and multi-typed models[4] on purchase behaviors, while treating other types of interactions as auxiliary information for the multi-typed models. Thus, we design different sampling strategies for single-typed and multi-typed models, as shown in Table III. For a fair comparison, we used the same embedding size and applied the same evaluation protocol to all baselines.

According to Table II, the performance of our model far exceeds that of the traditional recommendation models, and achieves a significant improvement compared with KGAT which also uses multi-typed information. By considering the multi-typed information, the multi-typed models perform better than the single-typed models. When comparing KGAT with our model, it also can be inferred that the two-level attention mechanism handles the multi-typed information better and further improves the performance. Besides, when using the static initialization for item embeddings, the performance of our model gains even higher on the Taobao dataset. However, the effectiveness of the static initialization depends much on the quality of the static attributes, which varies on different datasets. That's why the performance drops a little on the Jingdong dataset compared with the random initialization one.

For further comparison, we extended the sequential recommendation baselines from using single-typed information to multi-typed information by treating all types as the same type, as shown in Table IV. It can be inferred that the multi-typed interactions do carry more useful information since the performance of these baselines all improves. Moreover, the performance of the Taobao dataset (4 types) is better than that of the Jingdong dataset (2 types), which also confirms the above inference. However, these baselines cannot distinguish different interaction types, which leads to a suboptimal model. On the contrary, our model can make better use of multi-typed information and achieves better performance, by considering the user preferences for different behavior types and the potential relationships among items.

[3]The models that use single-typed interactions
[4]The models that use multi-typed interactions

D. Parameter Discussion

*1) **Effect of embedding size**:* The embedding size has the most obvious impact on model performance. As shown in Figure 4, the performance first grows rapidly and then gradually converges, as the embedding size increases. A large embedding size increases the model complexity, while the small one leads to insufficient results. Therefore, the choice of embedding size requires a trade-off. The default embedding size of our model is set to 50.

*2) **Effect of weighting parameter** α_1:* The weighting parameter α_1 controls the contribution of memory embedding to user embedding. To find out whether the memory network is helpful, we adjust α_1 from 0 to 1, as shown in Figure 5. The model gives unfavorable results while not using the memory network (i.e., $\alpha_1 = 0$). By involving the memory network, the performance improves and gets the best results when α_1 is around 0.2. However, when α_1 continues to rise, the performance drops. These results show that the memory network indeed helps, but putting too much focus on memory features weakens the intrinsic features, which is also important.

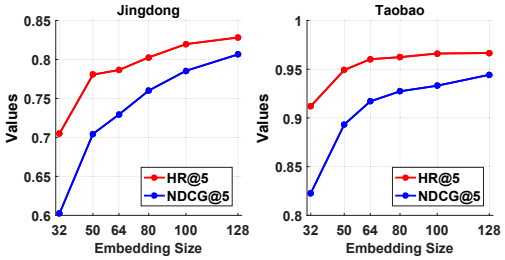

Fig. 4. Performance of our model under different embedding sizes.

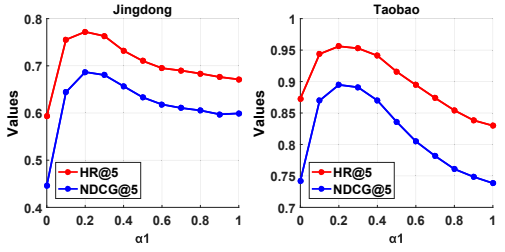

Fig. 5. Performance of our model under different choices of weighting parameter α_1.

E. Model Interpretability

In addition to learn weights automatically, the attention mechanism also improves the model interpretability. Here we will discuss the first-level attention, which learns the user preferences for different behavior types. The attention weights are shown in Figure 6. We can see that users focus more on the click behaviors on both datasets. One possible reason is that click records make up the majority of the datasets. Another reason might be that click behaviors are more effective, which can be explained from two aspects: 1) Users usually compare similar items before buying, which generates many useful click behaviors for predicting which to buy next. 2) The buying behaviors are less effective because users usually won't buy the same or similar items again in the near future.

IV. Conclusion

In this paper, we propose a sequential recommendation model DHIMN, which leverages the multi-typed and dynamic information in DHIN, and uses the two-level attention mechanism to model user preferences for different behavior types as well as the potential item relationships. To corporate the static and dynamic information in DHIN, we provide an optional step to learn the initial item embeddings from static attributes (categories). DHIMN has significant improvement on two real-world datasets, compared with several state-of-art baselines.

Acknowledgment

The paper was supported by the National Natural Science Foundation of China (61702568, U1711267), the Key-Area Research and Development Program of Guangdong Province (2020B010165003), the Guangdong Basic and Applied Basic Research Foundation (2020A1515010831), the Program for Guangdong Introducing Innovative and Entrepreneurial Teams (2017ZT07X355), and the Key Research and Development Program of Guangdong Province of China (2018B030325001).

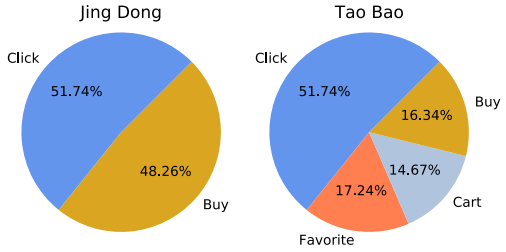

Fig. 6. Attention weights on different behavior types.

References

[1] L. Chen, Y. Xie, Z. Zheng, H. Zheng, and J. Xie, "Friend recommendation based on multi-social graph convolutional network," *IEEE Access*, vol. 8, pp. 43 618–43 629, 2020.
[2] F. Xie, L. Chen, Y. Ye, Y. Liu, Z. Zheng, and X. Lin, "A weighted meta-graph based approach for mobile application recommendation on heterogeneous information networks," in *ICSOC*. Springer, 2018, pp. 404–420.
[3] L. Chen, Y. Liu, X. He, L. Gao, and Z. Zheng, "Matching user with item set: collaborative bundle recommendation with deep attention network," in *IJCAI*. AAAI Press, 2019, pp. 2095–2101.
[4] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*, 2010, pp. 811–820.
[5] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*. IEEE, 2016, pp. 191–200.
[6] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *SIGIR*, 2016, pp. 729–732.
[7] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *ICLR*, 2016.
[8] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *Workshop on DLRS*, 2016, pp. 17–22.
[9] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*, 2018, pp. 565–573.
[10] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," in *WSDM*, 2019, pp. 582–590.
[11] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *WSDM*, 2018, pp. 108–116.
[12] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.
[13] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
[14] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *SIGIR*, 2019, pp. 165–174.
[15] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*. IEEE, 2018, pp. 197–206.
[16] X. Xin, X. He, Y. Zhang, Y. Zhang, and J. Jose, "Relational collaborative filtering: Modeling multiple item relations for recommendation," in *SIGIR*, 2019, pp. 125–134.
[17] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.
[18] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*. AUAI Press, 2009, pp. 452–461.
[19] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *SIGKDD*, 2019, pp. 950–958.