Similitude Attentive Relation Network for Click-Through Rate Prediction

Hangyu Deng*, Yulong Wang*, Jia Luo and Jinglu Hu
Graduate School of Information, Product and System, Waseda University,
2-7 Hibikino, Wakamatsu, Kitakyushu-shi, Fukuoka, Japan, 808-0135
deng.hangyu@fuji.waseda.jp, wangyulong@suou.waseda.jp, w.iac19160@kurenai.waseda.jp, jinglu@waseda.jp

***Abstract*—In online advertising systems, having a good knowledge of user behavior is crucial for click-through rate (CTR) prediction. In recent years, many researchers turn to seek a better way of user representation by modeling the behavior sequences with recurrent neural network (RNN). However, recurrent layers implicitly adopt the assumption that elements with different orders are fundamentally different, which is inefficient in many practical scenarios with much uncertainty and complicated hidden states. In this paper, we follow the paradigm of Relation Network (RN), and propose a new model called Similitude Attentive Relation Network (SARN). The user behavior is modeled as a graph, where nodes correspond to the visited items and edges correspond to the relations. To capture the latent user interest better, the model concentrates on the relations between items, rather than the translation on the time series. More specifically, the model tries to learn the similarity between items in a semantic space through a learnable dot-product operation and blend both of the item representations and relational information together as the final relations. We define our user representation on an attentive pooling of the relations directly. To verify the effectiveness of our method, extensive experiments on two public datasets and one real-world online advertising dataset are conducted. Experimental results show that our methods achieve usually better performance than others. Besides, we explore the properties of our model by controlled experiments and show the learned relational knowledge by visualizing the inner states of SARN.**

***Index Terms*—recommender systems, online advertising, click-through rate prediction, relation network**

I. Introduction

Click-through rate (CTR) prediction is one of the most important tasks in online advertising. Click-through rate is the ratio of clicks on the displayed advertisements to the total number of impressions. Many online advertising systems take cost-per-click model as the payment model, which means publishers make money only when an advertisement is clicked. The revenue can be approximately estimated as the sum of CTR multiplied by the benefit of each click across all the advertisement impressions. Therefore, CTR prediction plays a crucial role in online advertising.

In most online advertising systems, the advertisements are tailored to the users [1]–[3]. Thus CTR prediction is a task that learns the patterns of the user-item interactions. According to practical lessons and recent research, one of the most important features in CTR prediction is the historical information over the timeline, which contains important clues about the user's preference and behavioral patterns [4]–[7]. To some degree, a user can be represented as a sequence of items that he/she has interacted with. Therefore, many researchers apply RNN to CTR prediction in recent years [5], [8], [9]. Recurrent layers are born with a strong relational inductive bias of time translation invariance, which means a different order of the same input data makes a different result [10]. However, in many real-world cases, the user behaviors can be generated from very complicated hidden states and uncertainty. In other words, the user behavior sequence can be considered as a mixture of several sub-sequences, the behavioral patterns may be not so significant in the whole sequence. This may result in inefficient generalization for recurrent layers. In fig. 1, we take user *A26KXNN6H1IL0I* in Amazon Electronics dataset as an example. Since the sequence is mixed of camera related items, laptop related items and MP3 related items, we believe this user is interested in all these categories in the same period, and to a large extent, this sequence is generated by chance. Many different permutations of these items that keep consistent order within each category should also express approximately the same user representation. A way to deal with this kind of challenge is to introduce relational structures. Essentially, relational structures consider data as a graph, so this kind of module is naturally insensitive to the original order of data, and it may show better generalization ability in our example. However, many current deep learning models still lack explicit modules that reason the relations on the items. Recently, relation network (RN) has drawn attention from researchers. It is designed to capture the relational features between entities (*e.g.,* items), and has been proven to be effective in variable fields, such as computer vision [11]–[14]

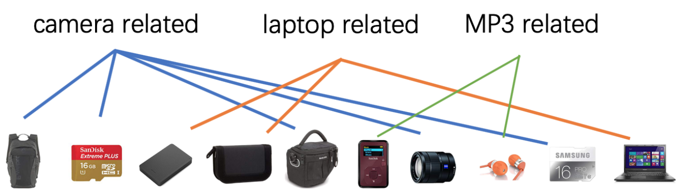


Fig. 1: An example to illustrate the complexity and uncertainty behind a user behavior sequence. This user is interested in three kinds of items. However, these items are not grouped by their categories exactly.

*The two authors contributed equally to this paper.

and deep reinforcement learning [15]. Better yet, this kind of approaches does not rely on any prior knowledge of the structured data.

Now that making use of relational structures has brought significant success in many fields, we believe reasoning the relations inside user behavior sequences is a considerable way for improving CTR prediction. In this paper, we utilize a fully connected graph to model the user behaviors. For simplicity, only *click* behaviors are taken into consideration, so that user behavior can be roughly defined as a list of visited items by a user with the past time. First, the model obtains item representations by employing an embedding layer like many others do. These item representations correspond to the nodes in the graph. Then our model executes representation learning and relational reasoning on all the item pairs, which correspond to the edges. To be specific, we design a more effective and flexible relational module for reasoning the similarities between items in semantic space. The output of the relational module contains pairwise relational information between any two items visited by the user. Our model structure is invariant to the order, and we turn to introduce relative positional information of each item to help our model converge. Thanks to the structure of our model, we can softly encode these information into the model by utilizing additional tasks and adding their loss terms as regularization.

In one word, our model executes representation learning and relational reasoning on a user behavior graph in an explicit manner, which helps improving the performance and interpretability of our model. The main contributions of our work are as follows:

- We point out the importance of handling the complexity and uncertainty behind the user behavior sequence and further investigate the way to perform relational reasoning to achieve better and more explainable CTR predictions.
- We propose SARN, which models the relations inside each user behavior sequence and distills similitude relations and in an explicit manner.
- We conduct comprehensive experiments on three datasets, and our method shows better performance and interpretability over many existing methods for CTR prediction.

II. Related Work

A. Feature Interaction Based Methods

In general, feature engineering has been crucial for many prediction tasks. As for CTR prediction, feature interaction based methods usually take user, target item and context features as input and try to predict the possibility that the user will click the target item. They usually map sparse features into low dimensional dense vectors and perform feature interaction operations, such as inner products, to extract sophisticated low- and high-order mixed features [16]–[18]. These methods indeed made a great progress in CTR prediction. However, recent research shows that it's hard for them to extract user's various interests or attribute-based collaborative signals, because their inductive biases essentially assume user-item pairs to be independent samples [19].

B. Behavior Sequence Based Methods

Intuitively, there is a huge semantic distance between users and items, which causes difficulty for deep learning models in learning proper representations for both of them. Behavior sequence based methods, rather than concentrating on the features, turn to model the user's historical behaviors. In this case, user representations contain not only user profiles, but also a list of items. By introducing items into user representations, this kind of methods narrow the semantic gap between users and items and achieved considerable results in recent years. [5] introduces RNN to model the sequential behaviors in the task of sponsored search. [20] proposes a modified RNN to learn the dynamic representations of users and global sequential behaviors, which facilitates next basket recommendation. [6] highlights the importance of modeling user's diverse interest and propose Deep Interest Network (DIN), which utilize an attention mechanism to extract user's interest with regard to the target item. [9] further investigates digging user's latent interest and capturing the evolution of the latent interest. [7], [21], [22] employ self-attention to model the user behavior sequence and achieve good results in recommender system, e-commerce recommendation and session based CTR prediction, respectively.

C. Relation Networks

Relation network (RN) provides a typical approach to machine relational reasoning in neural networks, and makes a significant breakthrough and achieves super-human performance in many tasks [11]. The main idea of the original RN is to fuse features from each entity-pair into one vector and can be described as the following function:

$$\mathrm{RN}(O) = f_\phi\left(\sum_{i,j} g_\theta(o_i, o_j)\right) \tag{1}$$

where the input is a set of *objects* O, while f_ϕ and g_θ are multi-layer perceptrons (MLPs). Intuitively, the g_θ plays the role of figuring out the relations between the object pairs.

Soon afterwards, [12] applies RN to multi-scale temporal relational reasoning in activity recognition. [13] improves object detection by simultaneously utilizing a relational module on a set of objects. [14] employs relation network for his meta-learning task and further investigate the learned non-linear distance metric for comparing items in few-shot learning and zero-shot learning.

III. Similitude Attentive Relation Network

The goal of Similitude Attentive Relation Network (SARN) is to predict the CTR given user's visited items, target item (the advertisement to be estimated) and context information, which can be expressed by the following function:

$$\hat{y} = f_{\mathrm{SARN}}(B, t, c) \tag{2}$$

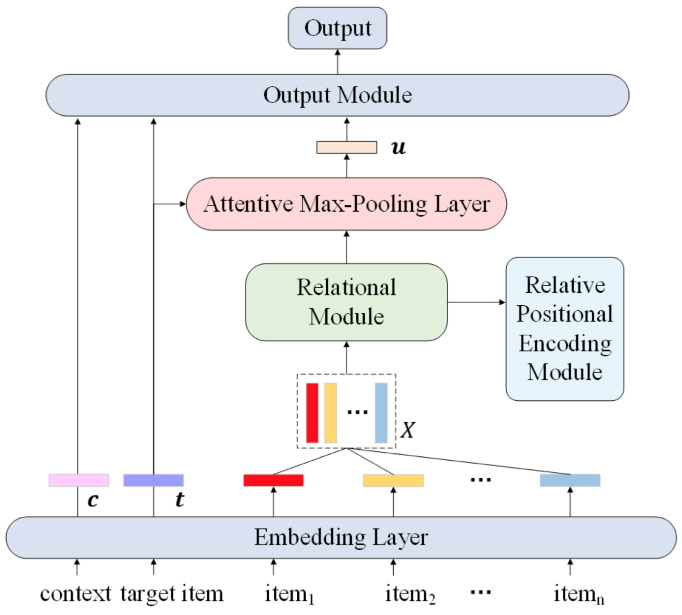


Fig. 2: Overall architecture of SARN.

where $B \in \mathbb{N}^n$ is a list of item ids that the user visited, n is the number of items in the list, $t \in \mathbb{N}$ is the id of target item and $c \in \mathbb{N}$ is the id of context information, such as the operation system on the user's device.

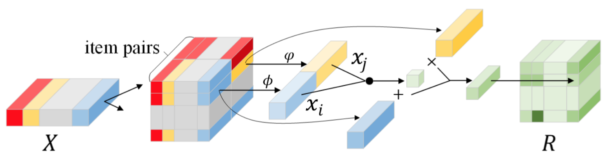
Next, we describe the whole architecture of SARN and our relational module.

A. The Overall Architecture of SARN

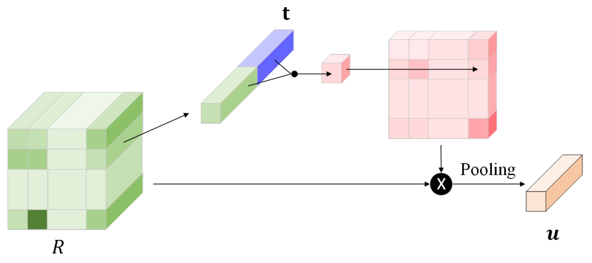
Like many other approaches of CTR prediction, our model creates embedding vectors from its input field, including the visited item list, target item and context information. As shown in fig. 2, SARN is comprised of several parts: embedding layer, relational module, attentive max-pooling layer, output module and extra relation encoding module. Next, we will introduce how we organize them together in detail.

1) Embedding Layer: The input of SARN is built upon prevalent approaches that use embedding mechanism to encode the sparse features into low dimensional vectors. More specifically, the embedding layer takes the user's visited item list, target item and context features as input and then transforms them into embedded item list $X \in \mathbb{R}^{n \times d}$, embedded target item vector $\boldsymbol{t} \in \mathbb{R}^d$ and embedded context vector $\boldsymbol{c} \in \mathbb{R}^d$ respectively, where n is the number of visited items and d is the dimensionality of embedding vectors.

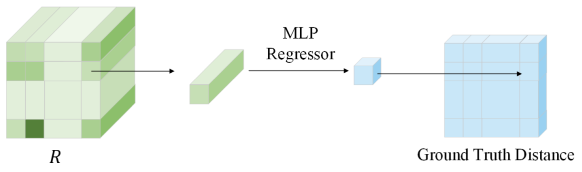
2) Relational Module: The relational module is the key module in SARN. It is a neural network cell g moving on each pairs of the items $(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In other words it considers items X as a set of nodes and links each nodes by executing relational reasoning on each item pairs. This module outputs the relational representations $R = G(X) = \sum_{i,j} g(\boldsymbol{x}_i, \boldsymbol{x}_j)$, with the shape of $n \times n \times d$. We will give more details about it in the next subsection.



(a)



(b)



(c)

Fig. 3: Three important components in our model. (a) Diagram of the relational module. It takes a list of item vectors as input, and makes a pair representation for every two items. (b) Diagram of Attentive Max-Pooling layer. Each attention score is utilized to scalar multiply the corresponding relation vectors before max-pooling. (c) Diagram of the relative positional encoding module.

3) Attentive Max-Pooling Layer: To extract user representation from the relational representations R, attentive max-pooling layer utilizes an attention mechanism similar to [6] and then summarizes the tensor into a vector $\boldsymbol{u}$ of $1 \times d$ by max-pooling over the first two axes.

4) Output Module: The output module takes target item vector $\boldsymbol{t}$, attentive user representation $\boldsymbol{u}$ and context vector $\boldsymbol{c}$ as input. Besides, we add max-pooling and average-pooling of X as additional features. It concatenates all the vectors together and feeds them into a MLP similar to [9]. The output is utilized to calculate negative log-likelihood loss function, which is defined as:

$$\mathcal{L}_{log} = -\frac{1}{N} \sum_{i=1}^{N} (y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)) \tag{3}$$

where y_i is the i^{th} label in the training set and $\hat{y}_i$ is the corresponding output of our model.

5) Relative Positional Encoding Module: Since the structure of the relational module ensures that it is invariant to the order of input sequence, we turn to encode the relative positional information into SARN. It employs a trainable MLP regressor, which takes a relation vector $g(\boldsymbol{x}_i, \boldsymbol{x}_j)$ as input and tries to predict the relative distance between them. The loss of predictions is added to the global loss function as a regularization term. Here mean square error is utilized to define the loss function of the MLP regressor:

$$\mathcal{L}_{reg} = -\frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}(\mathrm{MLP}(g(\boldsymbol{x}_i,\boldsymbol{x}_j)))-d(i,j))) \tag{4}$$

where the relative position of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $d(i,j)$, is the ground truth distance for this additional task.

Besides, we utilize the negative sampling trick in DIEN, which carries out an auxiliary loss $\mathcal{L}_{aux}$ [9]. The global loss function is:

$$\mathcal{L} = \mathcal{L}_{log} + \alpha\mathcal{L}_{reg} + \beta\mathcal{L}_{aux} \tag{5}$$

where α and is β are hyperparameters controlling the influence of relative positional information.

B. Relational Module

Formally, a relation R on a set X can be described as a subset of $X \times X$. If $(a,b) \in R$, we write aRb, which means a is connected to b with a relation R in a graph (X,R). Similarly, our relational module bridges two items in a user behavior sequence. We call the first item ”source node” and the other ”target node”. Unlike RNN passing information step by step along the time series, we design our module to pass information directly from source node to target node without the limit of spatial distance. Moreover, we hope that more information is passed when the source node has a close relationship to the target node in the semantic space. Therefore, our model needs the ability of relational reasoning to find out the relationship between item.

Inspired by the expression aRb mentioned above, we define our new relational module as follows:

$$g(\boldsymbol{x}_i,\boldsymbol{x}_j) = \boldsymbol{x}_i\mathcal{R}(\boldsymbol{x}_i,\boldsymbol{x}_j) + \boldsymbol{x}_j \tag{6}$$

where $\boldsymbol{x}_i, \boldsymbol{x}_j$ are learnable embedding of items, $\mathcal{R}$ plays the role of mining the relational knowledge between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Here our relational module mainly concentrates on finding the similarity between items, so we call our model similitude attentive relation network. By separating representation learning and relational reasoning into different parts, our relational module provides more flexibility when designing the pairwise interactions than MLP based module does. Besides, we apply vector-wise interaction in one of the custom operators, which avoids an element-wise fusion in our relational representation and may benefit the learning process [18]. Moreover, the new relational module provides more inspiration on creating new relational reasoning functions, that is firstly finding the relations between items and combine the item representations and relational information in an unsymmetrical manner.

The idea of relational module is to compare any two of the normalized item representations $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ using dot product, which reflects the cosine similarity between two learned representations. Fig. 3a shows the structure of our relational module. We define relational mapping functions $\phi(\boldsymbol{x}_i)$ and $\varphi(\boldsymbol{x}_j)$ equipped with an additional scaling function that scales vectors into unit vectors:

$$\phi(\boldsymbol{x}_i) = \frac{\mathrm{ReLU}\left(\boldsymbol{W}_\phi \boldsymbol{x}_i\right)}{\|\mathrm{ReLU}\left(\boldsymbol{W}_\phi \boldsymbol{x}_i\right)\|_2} \tag{7}$$

$$\varphi(\boldsymbol{x}_i) = \frac{\mathrm{ReLU}\left(\boldsymbol{W}_\varphi \boldsymbol{x}_i\right)}{\|\mathrm{ReLU}\left(\boldsymbol{W}_\varphi \boldsymbol{x}_i\right)\|_2} \tag{8}$$

Finally, we define similitude relational operation like this:

$$\mathcal{R}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right) = \phi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x}_j) \tag{9}$$

As shown in eq.(9), the similitude relation between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are computed by a dot product between $\phi(\boldsymbol{x}_i)$ and $\varphi(\boldsymbol{x}_j)$.

In conclusion, this module learns the relations by transforming item representation pairs into two vectors and calculating the cosine similarities, then fuse item representations using learned relations in a vector-wise level.

TABLE I: Statistics of datasets

Dataset	Users	Items	Cates.	Interactions
Electronics.	192,403	63,001	801	1,689,188
Books.	630,668	367,982	1,600	8,898,041
Taobao.	979,533	4,068,211	9,406	89,716,264
Online Ad.	224,308	4,237	23	2,718,078

C. More Discussion on RN and SARN

Many practices of RN model the universal relation on X using $g(\boldsymbol{x}_i,\boldsymbol{x}_j)$, where $\boldsymbol{x}_i,\boldsymbol{x}_j \in X$ and g is an MLP. That is to say, RN essentially describes a fully connected graph, where X represents the nodes and the edges correspond to the output of g, which is usually an MLP shared by item pairs. In these cases, the relational modules can be described by the following function:

$$g_\theta(\boldsymbol{x}_i,\boldsymbol{x}_j) = \mathrm{MLP}_\theta\left(concat(\boldsymbol{x}_i,\boldsymbol{x}_j)\right) \tag{10}$$

Parameter sharing scheme among edges indeed helps MLP achieve good performance in the frame of RN in various tasks. However, in our CTR prediction task, it still remains an open question that whether MLP is the most effective way to represent the edges in RN framework. According to recent research in relational reasoning, we need to explicitly model the entities and relations, and find a way for computing their interactions [10]. Now think about eq.(1) again, the RN finally takes the sum of g_θ as the input of the next-stage computation, that means only the summary of the edges are taken into account, while the nodes don't take part in it directly. In this case, the MLP not only plays the role of relational reasoning, but also bears the responsibility of passing necessary information of item representations. Therefore, the interactions between item representations and relational representations are deeply fused in the MLP. Nevertheless, MLP seems to be not so good at modeling high-order feature interactions in CTR prediction [18], [23]. Besides, MLP is an indivisible module, making it hard to introduce inductive biases to various tasks.

TABLE II: Overall Performance Comparison.

Model	Electronics.		Books.		Taobao.		Online Ad.
	AUC	LogLoss	AUC	LogLoss	AUC	LogLoss	AUC
Wide&Deep	0.8444	0.2448	0.7599	0.2849	0.8111	0.2742	0.6351
PNN	0.8498	0.2402	0.7603	0.3585	0.8217	0.2525	0.6421
GRU	0.8511	0.2535	0.7960	0.2870	0.8411	0.2541	0.6300
ATRank	0.8548	-	0.7533	-	0.8806	-	0.6494
DIN	0.8523	0.2731	0.7708	0.6352	0.8850	0.2140	0.6315
DIEN	0.8682	1.7522	0.8554	1.1258	0.8732	0.4387	0.6175
RN	0.8702	0.2320	0.8423	0.2457	0.8682	0.2347	0.6359
SARN	**0.8715**	**0.2283**	**0.8750**	**0.2261**	**0.9032**	**0.1947**	**0.6608**

IV. EXPERIMENTS

To verify the effectiveness of SARN, we conduct extensive experiments on two public datasets and an online advertising dataset. We first introduce necessary information about the datasets. Next, the setup in our experiments is provided, including evaluation metric, competitors and hyperparameter settings. Then we compare the proposed methods with competitors on three datasets with different settings. Furthermore, we show the visualization of the learned relations.

A. Datasets

We utilize three datasets: Amazon Dataset*, Taobao Dataset† and our online advertising dataset, which vary in terms of time span, category/item ratio, user number and interaction sparsity.

1) Amazon Dataset.: Amazon Dataset is a widely used benchmark dataset in this field [24]. It stores a huge number of product reviews and metadata in several subsets, spanning from May 1996 to July 2014. We choose two subsets named Electronics and Books to evaluate and analyze our proposed method. To make sure that each user has enough interactions, we adopt the 5-core setting for the dataset, which means each user and item have at least 5 interactions in the dataset.

2) Taobao Dataset.: This is a public dataset collected from Taobao user behavior data [25]. It records 987,994 users' behaviors during November 25 to December 03, 2017. In this paper, we also adopt the 5-core setting and only consider click behaviors, which account for the majority of all the behaviors. Similarly, each user-item interaction consists of user ID, item ID, category ID and timestamp.

3) Online Advertising Dataset.: The online advertising dataset is newly collected from the logs of a demand-side platform (DSP) company in Japan‡. Unlike the world's largest e-commence marketplace such as Amazon or Taobao running a unified online advertising platform, DSP receives advertisements from a variety of advertisers and displays them to users in different publishers (websites). Therefore, users and advertisements in this online advertising dataset are essentially from different sources, which introduces more uncertainty and is hard to construct a knowledge graph for it. The scheme of the online advertising dataset is quite similar to Amazon Dataset's. The only difference is that it records context information for each impression.

The statistics of datasets is shown in Table I. Since user-item interactions in the original datasets cannot be fed into the model directly, we need to reconstruct the datasets and split them into training sets and test sets. We form the datasets by collecting each user's visited items and sort them with regard to the timestamp when the interaction happens. Each sample contains features such as visited item list, target item and context information (for online advertisement dataset). The visited item list represents the user, and the target item is the candidate to be displayed. Since the public datasets do not contain negative samples, we conduct negative sampling by replacing the target item with a random item. As for splitting the training set and test set, we try two settings:

- Split by user: each user's samples (including positive and negative samples) are ether located in training set or test set. This method evaluates model's generalization performance on different users.
- Split by time: interactions before a certain timestamp are utilized to construct the training set, while others are for the test set. This method shows the generalization performance on the future data.

For public datasets, experiments are conducted on the split-by-user setting similar to [9]. We put approximately 80% of the samples into the training set, while the rest is in test set. Note that this kind of setting may be different from original papers, so the result is different. The reason we choose this setting is that we want to better measure the model's generalization ability on unseen users. For online advertising dataset, since there are real negative samples in the logs, we choose split-by-time setting to evaluate our model, which is closer to the actual scenario. Over 99% of negative samples in the training set are randomly dropped to keep the balance of classes. When generating the test set for online advertising dataset, we pour all the impressions after the certain timestamp into it.

B. Setup

1) Evaluation Metrics: We employ Area Under the ROC Curve (AUC) as one of the evaluation metric, which is believed to be suitable for CTR predictions because it is scale-invariant, threshold-invariant and insensitive to imbalanced data. Since we cannot obtain smooth ROC Curve using limited number

*http://jmcauley.ucsd.edu/data/amazon/

†https://tianchi.aliyun.com/dataset/dataDetail?dataId=649

‡https://www.mvrck.co.jp/

TABLE III: Performance over different maximum sequence lengths.

Model	Taobao. length=10	length=20	length=30	length=40	length=50	Books. length=10	length=20	length=30
ATRank	0.8664	0.8757	0.8661	0.8786	0.8806	0.7553	0.7520	0.7533
DIN	0.7837	0.8331	0.8647	0.8652	0.8850	0.7579	0.7701	0.7708
DIEN	0.7238	0.7814	0.8372	0.8560	0.8806	0.8499	0.8545	0.8554
RN	0.8665	0.8737	0.8797	0.8828	0.8682	0.8173	0.8305	0.8423
SARN	**0.8684**	**0.8786**	**0.8899**	**0.8975**	**0.9032**	**0.8622**	**0.8664**	**0.8750**

of samples in our practical tasks, we approximately calculate AUC using the coordinates of critical points on the ROC curve:

$$\mathrm{AUC} = \frac{1}{2}\sum_{i=1}^{m}(x_{i+1} - x_i)(y_{i+1} + y_i) \tag{11}$$

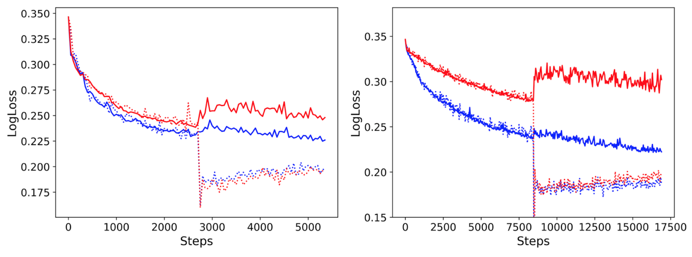
where m is the number of critical points and (x_i, y_i) is the coordinate of the i_{th} critical point.

Another evaluation metric is LogLoss, which is depicted in eq.(3). It measures the distance between the logits and the true label, so the uncertainty of logits is taken into consideration.

2) Competitors: We compare our method with both typical CTR methods and recently proposed methods.

- **Wide&Deep** [16]: Wide&Deep Model is a benchmark for deep learning based CTR predictions. It combines a linear module and MLP layers together. We employ an additional average pooling operation on visited items as additional historical information.
- **PNN** [17]: This method uses an embedding layer followed by a product layer to extract high-order features of the categorical data.
- **GRU**: Following the idea of [5], we utilize embedding mechanism and a GRU layer to model the user's visited item list.
- **ATRank** [7]: ATRank applies a modified self-attention mechanism to model user behaviors. It also encode relative timestep information into the item representations.
- **DIN** [6]: DIN is a representative behavior sequence based model, which is designed to concentrate on historical behaviors that are related to the target item with an attention mechanism.
- **DIEN** [9]: DIEN models user's dynamic interests using two-layer GRU with an attention mechanism. It encodes the neighborhood information using an auxiliary task.
- **RN**: We transplant RN to this task by replacing the relational module with an original RN. The only difference is that its relational module directly outputs $\mathcal{R}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, instead of $\boldsymbol{x}_i\mathcal{R}(x_i, x_i) + \boldsymbol{x}_j$ as depicted in eq.(6).

3) Hyperparameter Settings: For almost all the models, the embedding size of user, item, category and context is 18, which is the same with DIEN's. To keep consistent with the original setting, ATRank's embedding size is 64. Each item embedding vector and category embedding vector are concatenated together as the basic item representation. The number of hidden units in our relational module is 36, which is the same with the basic item representation's. We utilize Adam optimizer to train all the models, with decaying learning rate starting from 0.01 and batch size equal to 128, except for ATRank, which is trained by stochastic gradient descent (SGD) optimizer with decaying learning rate starting from 1.0 and the batch size equal to 32, because we find it performs better for ATRank with original settings. As for the MLP that outputs the final logits, we choose fully connected layers with the shape of $(50, 10, 2)$ for our model, while others are based on the code released by the authors.



(a) Amazon Electronics. (b) Amazon Books.

Fig. 4: LogLoss during training. The red lines represent the LogLoss of our model without relative positional encoding module, and the blue lines represent the LogLoss of our model with it. The solid lines represent the LogLoss on the test set, while the dotted lines represent it on the training set.

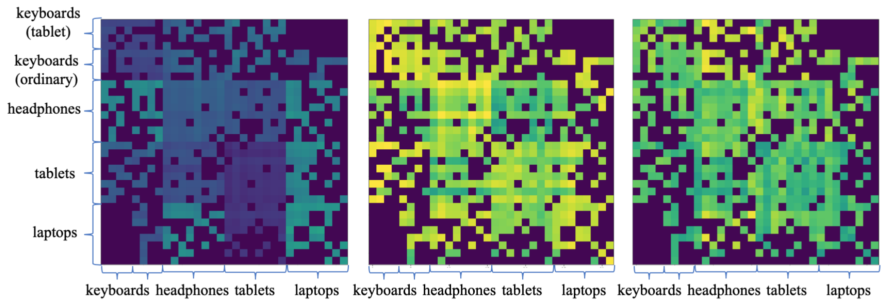
C. Performance Comparison

We first verify the effect of SARN on both public datasets and online advertising dataset. Since the test data in online advertising dataset is highly imbalanced, we don't utilize LogLoss to evaluate the models because it is sensitive to imbalanced data. Each result in Table II is the average value of ten experiments.

We find that behavior sequence based methods perform significantly better than others. That may be because there is no comprehensive side information on both users and items in our dataset, which increases the difficulty of finding the collaborative signals. Recent models employ attention mechanism to extract more helpful information, and they improve the performance in many cases. We then compare our modules with RN. It seems that RN based methods generally achieve better results. Benefiting from new relational modules, our SARN shows better performance than RN, especially on Taobao dataset.

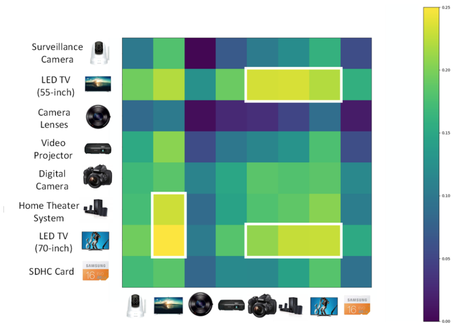
D. Study of SARN

To have a better knowledge of SARN, we conduct two series of controlled experiments. First, we study the influence of the



Step 0, AUC 0.50 Step 1000, AUC 0.83 Step 5000, AUC 0.87

(a) The evolution of relations during training.



Step 5400, relations of a user

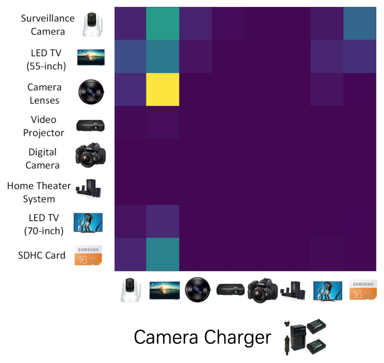
(b) Case study on a user after training.

Fig. 5: Visualization of the relations

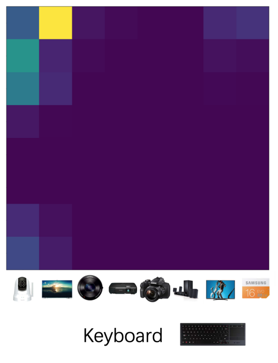
maximum length of behavior sequences to the CTR prediction models. Then the effect of encoding relative position is verified.

1) Different Maximum Length.: In general, different maximum length not only changes the amount of historical information provided to the models, but also affects the interaction sparsity of the dataset, because the previous interactions are discarded if the user visited too many items. The model needs to learn proper representations for inactive users with few interactions [19]. We conduct experiments on Taobao dataset and Amazon Books dataset, which record more items for each user, so that we can control the maximum length in a larger range. As we can see in Table III, SARN outperforms other models especially when the maximum length is relatively short. When the length is 50 on Taobao dataset, all the models achieve relatively higher results. It seems that with relational module, our model can learn better representations for users with less visited items. One interesting thing is that, with the growth of the length, the AUCs of DIN and DIEN increase progressively, while ATRank and SARN's AUCs do not increase so fast. The reason why ATRank show similar property with SARN may be because the dimensions of inner states in ATRank and SARN are proportional to the square of the length, while the dimensions of inner states in DIN and DIEN are proportional to the length itself.

2) Effect of Relative Positional Encoding Module: During our experiment, we find that encoding extra relations helps a lot in terms of improving model performance. Controlled experiments are then further conducted on SARNs with and without extra relation encoding module on Amazon datasets. The training curves in fig. 4 record the LogLoss during two epochs. We can find that after the first epoch, the training loss of both models decreases dramatically, which means they are likely to overfit the training set. Without the help of extra relation encoding module, it performs badly on the test set, especially on the Amazon Books dataset. To some degree, the extra relation encoding module regularizes the learning process, making SARN more stable and more likely to avoid overfitting.



(a) Attention scores (positive)



(b) Attention scores (negative)

Fig. 6: Attention scores of a user.

E. Visualization

To give a deep insight of SARN, we try to visualize the relations in both global level and individual level.

Since relation is the inner product between two item representations, we can construct a big heatmap indicating the relations between any two items by summing all the relations calculated from the dataset, obtaining the average value of each relation and integrating them into the big heatmap. Since there are too many items in Amazon Electronics dataset to display in our paper, we select popular items in the categories of *tablets*, *laptops*, *keyboards*, and *headphones*.

Fig.5a shows the evolution of relations during training. Both axes represent the list of items grouped by categories, and the pixels indicate the relations between the corresponding items on x-axis and y-axis. At the beginning, there are clear boundaries between category blocks and the values of all the relations are small. That is because the two transformations in eq.(9) are randomly initialized with small values and do not make much difference on item representations, so the relations between items are largely affected by the categories. With the process of training, the boundaries disappear a little, and the highlight area increases, especially along the diagonal. At last, the area near the diagonal become significantly lower and only a small number of relations have relatively high value. Moreover, the highly related items do not always share the same category, while items in the same category may also have different highly related items. This phenomenon may indicates that during training, our model firstly increases the values of relations in a wide range, and then deactivate those relations that are not so important, such as the relations between items themselves.

Next, we visualize user *A1ZU55TM45Y2R8* in Amazon Electronics dataset. Fig. 5b depicts relations of a subset of the items him/her visited. Note that the relations of some item itself is not necessarily to be high, this is because the similarity is calculated in a semantic space. And it is interesting that TVs probably share higher relations with home theater systems and digital cameras. This phenomenon may indicate that TV plays a core role in the family.

As for the attention scores on relations, in fig. 6, we can find that for positive candidate (Camera Charger), the highest attention locates on a relation from TV to Camera lenses, which is quite relevant to the Camera Charger, while the highest attention for the negative candidate (Keyboards) locates on irrelevant items. Therefore, after attentive max-pooling layer, the final user representation is quite close to the positive candidate and not so close to the negative candidate.

V. Conclusions

In this paper, we investigate the possibility of introducing relational modules into CTR prediction. The User interests behind their behavior sequences are highly complicated and need more specific methods to represent the underlying structures. Based on this observation, SARN is proposed, which models a user from pairwise relations and extract relational features explicitly. Experiments show that SARN has the potential to improve the performance and interpretability of CTR prediction models.

References

[1] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, "Practical lessons from predicting clicks on ads at facebook," in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 2014, pp. 1–9.

[2] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *In Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 191–198.

[3] R. Catherine and W. Cohen, "Personalized recommendations using knowledge graphs: A probabilistic logic programming approach," in *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 2016, pp. 325–332.

[4] N. Hariri, B. Mobasher, and R. Burke, "Context-aware music recommendation based on latenttopic sequential patterns," in *In Proceedings of the 6th ACM Conference on Recommender Systems*. ACM, 2012, pp. 131–138.

[5] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu, "Sequential click prediction for sponsored search with recurrent neural networks," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2014.

[6] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1059–1068.

[7] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," in *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, 2018.

[8] Q.-H. Chen, S.-M. Yu, Z.-X. Guo, and Y.-B. Jia, "Estimating ads' click through rate with recurrent neural network," in *ITM Web of Conferences*, vol. 7. EDP Sciences, 2016, p. 04001.

[9] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai, "Deep interest evolution network for click-through rate prediction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 5941–5948.

[10] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, and R. Faulkner, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[11] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap, "A simple neural network module for relational reasoning," in *Advances in neural information processing systems*, 2017, pp. 4967–4976.

[12] B. Zhou, A. Andonian, A. Oliva, and A. Torralba, "Temporal relational reasoning in videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 803–818.

[13] H. Hu, J. Gu, Z. Zhang, J. Dai, and Y. Wei, "Relation networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3588–3597.

[14] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 1199–1208.

[15] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, M. Shanahan, V. Langston, R. Pascanu, M. Botvinick, O. Vinyals, and P. Battaglia, "Deep reinforcement learning with relational inductive biases," in *Proceedings of the International Conference on Learning Representations*, 2019.

[16] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*. ACM, 2016, pp. 7–10.

[17] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," in *Proceedings of the 16th International Conference on Data Mining*. IEEE, 2016, pp. 1149–1154.

[18] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1754–1763.

[19] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua, "KGAT: knowledge graph attention network for recommendation," in *KDD*, 2019, pp. 950–958.

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

[21] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in alibaba," *arXiv preprint arXiv:1905.06874*, 2019.

[22] Y. Feng, F. Lv, W. Shen, M. Wang, F. Sun, Y. Zhu, and K. Yang, "Deep session interest network for click-through rate prediction," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[23] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *In Proceedings of the 27th International Joint Conference on Artificial Intelligence*, ser. IJCAI'18. AAAI Press, 2018, pp. 2227–2233. [Online]. Available: http://dl.acm.org/citation.cfm?id=3304889.3304969

[24] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *In Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2015, pp. 43–52.

[25] H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, and K. Gai, "Learning tree-based deep model for recommender systems," in *In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 1079–1088.