Towards Accurate and Interpretable Sequential Prediction: A CNN & Attention-Based Feature Extractor

Jingyi Wang*
University of Electronic Science and Technology of China
Chengdu, China
jingyi.wang@std.uestc.edu.cn

Qiang Liu
RealAI and Tsinghua University
Beijing, China
qiang.liu@realai.ai

Zhaocheng Liu
RealAI
Beijing, China
zhaocheng.liu@realai.ai

Shu Wu†
Institute of Automation and Artificial Intelligence Research
Chinese Academy of Sciences
Beijing, China
shu.wu@nlpr.ia.ac.cn

ABSTRACT

With the influence of information explosion, there are more and more choices exposed to public view. Next item recommendation is being a significant and challenging task. Recently, attention mechanism, Convolutional Neural Networks (CNN) and other kinds of deep components are used to model user behaviors. However, the proposed models often fail to extract the feature of user behaviors in different time periods and the CNN-based models before are hard to make the used CNN interpretable. In this paper, we propose a **C**NN & **A**ttention-based **S**equential **F**eature **E**xtractor (CASFE) module to capture the possible features of user behaviors at different time intervals. Specifically, we import CNN to extract multi-level features of user behaviors with different time periods. After each CNN layer, we use attention module to emphasize the different effect of behaviors on the prediction result. Besides, the features we try to extract here have the similar concept and meaning with the hand-crafted features in Feature Engineering, which proves the validity of CASFE. Accordingly, CASFE becomes a general sequential feature extractor that can be used in various sequential prediction tasks. With Multi-Layer Perceptron (MLP), CASFE would be a state-of-the-art next item recommendation model. The model obtains good performance on *Last.fm_1K* dataset and *MovieLens_1M* dataset. Besides, as a compatible extractor module, it can also promote CTR prediction models as well as other sequential prediction tasks.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Machine learning*; *Neural networks*.

KEYWORDS

Sequential prediction; recommender systems; CTR prediction; sequential feature extractor

ACM Reference Format:
Jingyi Wang, Qiang Liu, Zhaocheng Liu, and Shu Wu. 2019. Towards Accurate and Interpretable Sequential Prediction: A CNN & Attention-Based Feature Extractor. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM'19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357887

1 INTRODUCTION

With the influence of information explosion, more and more choices are presented to the public. Accurate recommendation items would lead to higher user stickiness in general. So service platforms like music or shopping websites need effective sequential features of user behaviors to improve service quality. Accordingly, extracting sequential feature is a necessary and significant stage when modeling user behaviors. To conclude user features for sequential prediction, models try using collaborative filtering [18] or some deep learning modules. An example of applying these extracted features is *next item recommendation*, which is an important task in real-world recommender systems [28]. In this task, data is processed into sequences of historical items that users clicked or bought chronologically. The next time when a user logs in, some personalized items would be presented to him according to the features generated by recommendation algorithms on browsing history. In a word, the main task

*This work is done when Jingyi Wang worked as an intern at RealAI.
†Shu Wu is the corresponding author.

CIKM '19, November 3–7, 2019, Beijing, China

ACM ISBN 978-1-4503-6976-3/19/11…$15.00
https://doi.org/10.1145/3357384.3357887

of feature extracting here is to extract pivotal user features from sequential data. Based on the extracted features, systems can yield more accurate recommendation to users.

For extracting users' sequential features to achieve personalized recommendation, many recommendation algorithms import deep learning modules. There are many related models based on RNN [4, 7, 9, 14]. But their speed is usually slower than other models obviously. Attention mechanism and CNN are the two most commonly used modules due to their good performance. The importance of attention mechanism is illustrated in STAMP [17] and other related models. CNN is popular recently for its good performance on computer vision [8] and some other tasks [2, 21]. Caser [23] is a successful example of using CNN in the sequential recommender, which achieves a comparable result better than the previous RNN models. NextItNet [24] is introduced with dilated CNN and residual learning. But they are hard to give a reasonable interpretation of the information and features got from CNN so they are usually considered as black boxes. Besides, they need additional residual block [6] if a multi-layer CNN is used. Otherwise, the learning would be hard to continue because of gradient issues.

As we can see, few models could capture the possible sequential features during time periods with different fixed lengths. For example, if a person is used to watching a romantic movie every weekend but prefers comedy movies at other times, there is a great possibility that the proposed models would recommend a comedy. To solve this problem, this paper performs an appropriate combination of CNN and attention module as the main part of CASFE. With a multi-layer CNN, we not only expand the receptive field on sequential behaviors but also try to explore the possible periodicity of user behavior. Periodicity here means the possible features with time periods of some fixed lengths. For example, a user might be used to watching an inspiring movie once a week and might prefer a romantic movie every month with his/her partner. Afterward, we add attention module after each convolution layer to emphasize the different importance of sequential actions. In this way, CASFE focuses on important actions and reduces the focus on unimportant actions. Then we concat outputs of all attention layers and import MLP to get the final prediction scores of all alternative items. Accordingly, we can alleviate information loss and possible gradient problems because the results of all CNN layers are connected to the final result. So we don't need an additional residual neural network [6].

CASFE has good compatibility and interpretability. Besides next item recommendation, CASFE can also be applied to other sequential prediction tasks, e.g., CTR prediction, trail prediction, credit investigation. In CASFE, the interpretability is expressed by the attention weights after each CNN layers. In CNN, different filters correspond to different time periods and are responsible for exploring user features during different time periods.

The contributions of this work are as follows.

1. We propose a CNN & attention-based sequential feature extractor (CASFE) to extract latent multi-level user features from sequential behaviors. To the best of our knowledge, this is the first work that joints CNN and attention mechanism well at the level of the CNN layer in the sequential prediction field.
2. CASFE has good compatibility and interpretability. It can be applied to next item recommendation, CTR prediction, and other various sequential tasks. Attention network brings good interpretability.
3. We perform some experiments on the frequently used datasets Last.fm 1K and MovieLens 1M. The proposed CASFE module has been demonstrated state-of-the-art compared to other common recommendation models. It is also proved useful in some CTR prediction models.

2 RELATED WORK

In this section, we present an overview of the proposed recommendation models.

2.1 Rule-based Recommendation

In early works, items are recommended according to the popularity or category simply. Popularity-based recommendation MostPop used as a baseline in [24] couldn't be personalized. It always recommends the items that are clicked or bought most for any user no matter which category he prefers. The category-based recommendation still couldn't satisfy users' needs because it simply focuses on the categories users like but it isn't able to capture the user feature hidden in sequential behaviors.

2.2 Collaborative Filtering

Then collaborative filtering (CF) is proposed using all users' sequences. Neighborhood-based CF models recommend according to the similarity among users or items. The stored user-item ratings are directly used for new items. This can be done in two ways known as user-based or item-based recommendation [18].

With large datasets, it is difficult for neighborhood-based models to perform real-time recommendation. So here comes model-based CF models that use the historical data to train the model and then achieve the real-time recommendation with the model. For example, in Matrix Factorization (MF) model, the authors decompose the *user-item* rating matrix into the *user-feature* matrix and the *feature-item* matrix [10, 20]. In this way, we not only get users' preference and items' features but also reduce the dimensions of the matrix.

2.3 Deep Learning Models

Recently, deep learning modules are popular and frequently used in the session-based next item recommendation task.

2.3.1 RNN-based models. There are many related models based on RNN. For example, [7] adapts the RNN models to the recommender setting by introducing a new ranking loss function. Based on RNN, Yu et al. proposed a Dynamic

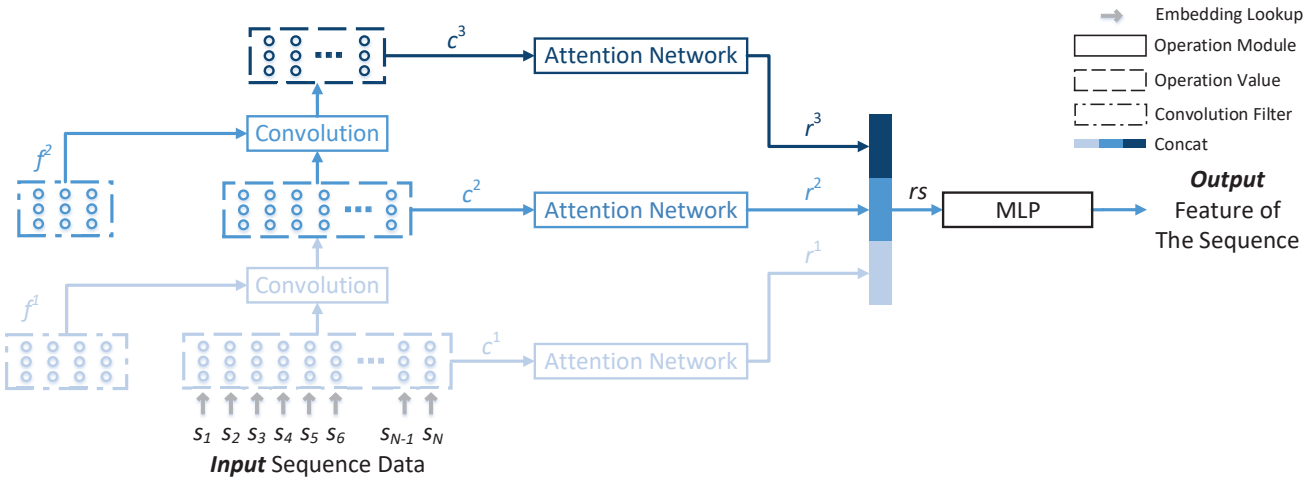


Figure 1: The architecture of the CASFE module. Darker blue means a deeper CNN layer. Filters with different lengths are responsible for extracting features during time periods of different lengths. Attention network is imported after each CNN layer to emphasize the important features of convolution results. Then the results of all attention networks are jointed together and inputted into MLP or other modules.

REcurrent bAsket Model (DREAM) [4] which makes use of the dynamic representation of a specific user and the global sequential features for next basket recommendation. Data augmentation and a method to account for shifts in the input data distribution are proposed to improve RNN-based models for session-based recommendations [22]. SASRec [9] succeeds in balancing Markov Chains models and RNN-based models. The context-aware recommendation has also been extensively studied [14]. Liu et al. proposed CA-RNN [14] which utilizes adaptive context-specific transition matrices for modeling varied transition effects. [15] extends RNN and proposes a novel method called Spatial Temporal Recurrent Neural Networks (ST-RNN).

2.3.2 Attention-based models. Attention module gives different weights to items in a sequence. Zhou et al. designed a local activation unit to adaptively learn the representation of user interests from historical behaviors with respect to a certain ad in DIN model [27]. In DIEN [26], the authors designed an interest extractor layer to capture temporal interests from history behavior sequences. Neural Attentive Recommendation Machine (NARM) [12] employs attention mechanism to capture the main purpose from the hidden states and combines it with the sequential behavior. STAMP proposed a novel attention mechanism in which the attention weights are enhanced with the users' current interests.

2.3.3 CNN-based models. CNN has achieved great performance on computer vision [8] and natural language processing [2]. It also makes an improvement in information retrieval [21], speech recognition [1] and image classification [11, 14]. CNN-based CCPM [16] can extract local-global key features from an input instance with varied elements, which can be implemented for not only a single ad impression but also a sequential ad impression. With horizontal and vertical convolution layers, Caser captures both general preferences and sequential patterns. With dilated convolution, NextItNet increases the receptive fields without relying on the pooling operation.

3 MODEL DESIGN

Details of CASFE are stated in this section. Then we give two examples that make use of CASFE in the recommendation field.

Users' behavior history would be recorded by the browsed platform as the dataset. For example, *Last.fm_1K* provides the dataset of chronological music listening history of a thousand users. The task is to build a model that analyzes the user behaviors and outputs scores of all alternative items. Then the platform would recommend the top items in the scores list. In this paper, the set of all candidate items in a certain recommendation circumstance is denoted by $\boldsymbol{I} = \{i_1, i_2, ..., i_{|\boldsymbol{I}|}\}$. All the known and predicted items belong to set $\boldsymbol{I}$. A sequence is defined as $\boldsymbol{S} = \{s_1, s_2, ..., s_{N-1}, s_N\}$ where $s_j \in \boldsymbol{I}\,(1 \leq j \leq N)$. N denotes the number of items in the sequence, i.e., the length of the sequence. $\boldsymbol{S_t} = \{s_1, s_2, ..., s_{t-1}, s_t\}$ is defined as the sub-sequence of $\boldsymbol{S}$, which is actually the prefix items of $\boldsymbol{S}$ with the length of $t\,(1 \leq t < N)$. So the task converts to calculating the prediction scores for s_{t+1} and recommending the top items in the scores list.

3.1 CASFE Module

The structure of CASFE is shown in Figure 1. It can be roughly divided into four parts as follows.

3.1.1 Input. We use $\boldsymbol{E} = \{e_1, e_2, ..., e_{N-1}, e_N\}$ to denote the embeddings of a sequence $\boldsymbol{S}$. $e_t \in \mathbb{R}^d, 1 \leq t \leq N$ denotes the d-dimension embedding expression of s_t in $\boldsymbol{S}$. $\boldsymbol{E} \in \mathbb{R}^{N \times d}$ is

got by looking up e_t in a look-up table that contains embeddings of all features.

3.1.2 CNN layers. CASFE takes $\boldsymbol{E}$ as the input which means $\boldsymbol{E}$ is set as the first layer of CNN c^1. The filter height of each CNN layer is d. So we can regard it as a one-dimension convolution like Figure 2. f^i denotes the convolution filter and c^i denotes the convolution result of the i-th CNN layer in Figure 1 and Figure 2. The convolution is calculated row by row as:

$$c^i_{j,k} = \sum_{l=k}^{k+f_w-1} c^{i-1}_{j,l} * f^{i-1}_{j,l}. \tag{1}$$

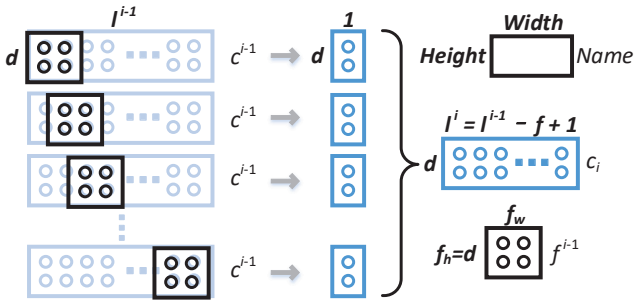


Figure 2: One-layer convolution with single feature map. The convolution here is calculated row by row.

3.1.3 Attention network. After each CNN layer, we perform a multi-layer attention net on the convolution result c^i to emphasize the difference among items and focus on the important actions. A three-layer attention network example from c^i to r^i is showed in Figure 3. The effect of the last layer is converting the matrix to a vector as the weights. Hadamard product is performed on two matrices with the same shape. The result also has the same shape as the two product items. Hadamard product means multiplying the corresponding elements with the same position in the two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$:

$$(A * B)_{i,j} = A_{i,j} * B_{i,j}. \tag{2}$$

With the reduce sum operation, we add the values each row of $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ as the element in the result $\boldsymbol{B} \in \mathbb{R}^{m \times 1}$ at the same row:

$$B_i = \sum_{j=1}^{n} A_{i,j}. \tag{3}$$

Then the results of all attention networks are connected together as $rs \in \mathbb{R}^{len \times 1}$. len is the length of rs that can be calculated by:

$$len = F \times d + C \times M \times F \times d \tag{4}$$

where F is the number of feature fields, C is the number of CNN layers and M is the number of feature maps.

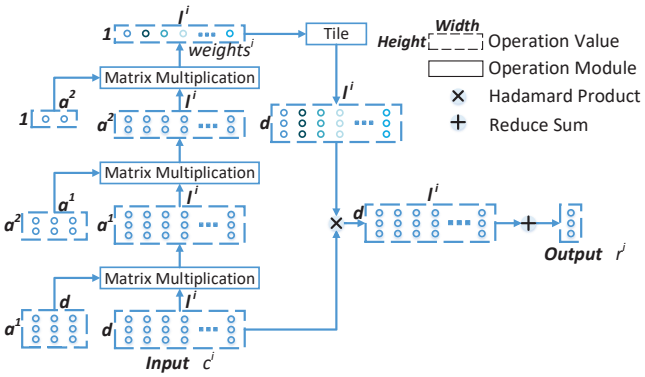


Figure 3: Three-layer attention net. Similar operations are performed on the result of each layer of convolution.

3.1.4 Output. Finally, we use MLP to explore the feature further. The calculation of a single hidden layer of MLP is:

$$y[i] = \begin{cases} rs & i = 0 \\ f(w[i]y[i-1] + b[i]) & i > 0 \end{cases} \tag{5}$$

where $w[i]$ is the weight matrix of the i-th layer with the height of $y[i]$ as the width, $b[i]$ is the bias and f is the activation function between layers. We use tanh here:

$$tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}. \tag{6}$$

The output of MLP is the feature of the sequence that CASFE extracts, which can be utilized in next item recommendation. With other networks instead of MLP, the result can also be used in the CTR prediction task.

3.2 CASFE in Next Item Recommendation

As a sequence feature extractor module, CASFE can be jointed with other data processing or further feature exploring modules. The combined model can be widely used in sequential prediction because of its outstanding performance in extracting hidden features in sequential data. For example, CASFE has a good performance in next item recommendation. After CASFE, we perform matrix multiplication on its output and the transposed embedding lookup table. With the recommendation grades of all possible items for the current sequence denoted as $z \in \mathbb{R}^{1 \times d}$ and the embedding dictionary denoted as $D \in \mathbb{R}^{|\boldsymbol{I}| \times d}$, the calculation is:

$$z = y \times D^T. \tag{7}$$

For an arbitrary item in the items set, the more similar its embeddings and the feature CASFE extracted are, the higher grades it would get through this operation. This is the essence of the multiplication and the items with top grades would be recommended. In order to calculate the loss for model training, we first perform softmax function on z as:

$$\hat{p}_i = \frac{e^{z_i}}{\sum_{j=1}^{|\boldsymbol{I}|} e^{z_j}}. \tag{8}$$

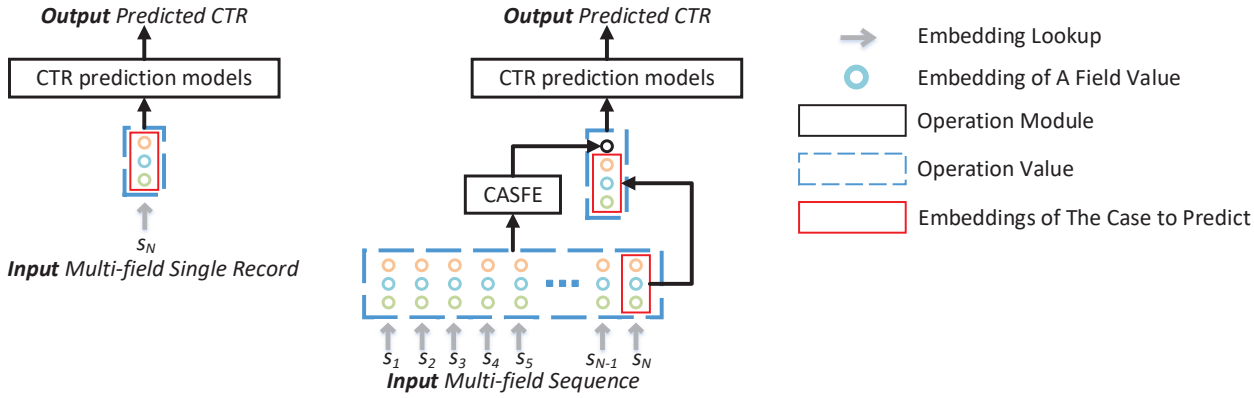


Figure 4: CTR prediction models without & with CASFE. The left architecture shows the original CTR prediction models. In the right one, the output of CASFE is jointed together with the origin input. The output of CASFE which indicates sequential information is used as a feature of the item to be predicted.

In this way, we convert the grades of all items into their probabilities. Then we use cross entropy as the loss function. The label, i.e., the actual item the user chose is first converted to a one-hot vector. For example, if the label is the third item, it would be converted to [0, 0, 1, ..., 0]. Here we denote the one-hot label vector as p. So the cross entropy is

$$Loss(\hat{p}) = -\sum_{i=1}^{|I|} p_i * \log(\hat{p}_i). \tag{9}$$

We perform some experiments on the combined model and the result is showed in Section 4.2.

3.3 CASFE in Broader Sequential Prediction Tasks

As stated in Section 1, CASFE not only works in next item recommendation but also can be applied to other sequential prediction tasks such as CTR prediction, trail prediction, and credit investigation. For example, we joint the output of CASFE and the input of CTR prediction models. In this way, we could use sequential data for the task instead of the original data organization and the origin models would be improved with CASFE.

As we know, applying CNN on the original data for CTR prediction directly doesn't work very well because different arrange orders of original features do not have different meanings[13]. So in CASFE, CNN is performed on the level of items from a macroscopic view. The specific approach and details of CASFE with CTR prediction models are shown in Figure 4. The left architecture in the figure is the origin CTR prediction approach and the right is the jointed one. As we can see, traditional models only use the given specific record. They are unable to take useful sequential information. Actually, this can be achieved by importing CASFE to deal with the sequential records before the specific case. The data inputted into CTR prediction models include the features embeddings of the last item S_N and the result of CASFE on the whole sequence. The output of CASFE is attached to the embeddings of S_N as a feature which signifies the sequential information. This combination mode of CASFE and CTR prediction models are beneficial for proving the compatibility of CASFE.

In CTR prediction, CASFE can promote predictive models by attaching its output to the input of origin models. In this process, the features extracted by CASFE have the similar concept and sense with the hand-crafted features in Feature Engineering, which is a good proof for the validity of the extracted features. Section 4.4 would show the details and results of experiments about CTR prediction models with & without CASFE. Besides, we validate the module on a private dataset about credit and get good results.

4 EXPERIMENTS

In this section, we carry out various experiments to prove the excellence of CASFE in sequential prediction. First, we compare its performance with the former state-of-the-art models in next item recommendation to prove its advanced property in the sequential field. Second, we test CASFE on the CTR prediction task trying to prove its compatibility. Then, results with different hyper-parameters are compared and analyzed. At last, we analyze the specific parameters in CASFE's attention layers to explain its good interpretability.

4.1 Datasets and Experiment Setup

4.1.1 Datasets. We carry out experiments on two frequently-used datasets. One is a music listening record of 1,000 users from *Last.fm_1K*[1]. The other is *MovieLens_1M*[2] with 1 million ratings from 6,000 users on 4,000 movies. N denotes the number of items in a sample sequence.

As for the *Last.fm_1K* music dataset, we process it by drawing 20,000 songs randomly for *Music_m* and 200,000 songs randomly for *Music_l*. Only records with drawn songs

[1]http://www.dtic.upf.edu/ ocelma/MusicRecommendationDataset/lastfm-1K.html
[2]https://grouplens.org/datasets/movielens/1m/

Table 1: Dataset statistics. Sequences in different datasets are counted.

	Music						MovieLens				
DataSet	m5	l5	l10	l20	l50	l100	n5	n10	n20	ca	cb
Sequences	0.616M	2.101M	1.050M	0.525M	0.209M	0.104M	0.113M	0.055M	0.026M	0.946M	0.946M

are used. With a sliding window of both size and stride of N on those records, we could get datasets of some fixed sequence lengths. In addition, records in a window, i.e. a sequence sample must have the same user and the time difference between the last two items must be less than 2 hours. In *Music_m*, N is 5 because this drawn dataset is not large. *Music_l* is converted to 5 datasets with N of 5, 10, 20, 50 and 100.

As for the *MovieLens* dataset, there are two different processing approaches for different tasks. The two approaches are still with the limiting conditions of the same user in a sequence. The first approach ignores the records with ratings of 4 and 5. We convert the original dataset into 3 datasets with N of 5, 10 and 20. Only the movie_id field is used as the feature. In this way, the generated dataset *MovieLens_n* could be used in next item recommendation. The second approach uses all relevant fields including 5 fields about users and 7 fields about movies. They are user_id, user_gender, user_age, user_occupation, user_zipcode, movie_id and movie_genres(1-6). The maximum quantity of genres of a movie is 6 so we have 6 fields for movie_genres and use a specific feature to fill the movies with genres less than 6. With a sliding window whose size is 10 and stride is 1, we get *MovieLens_ca*. For every window, i.e., every case in *MovieLens_ca*, we extract the last rating item in the window and get *MovieLens_cb*. The usage of these datasets would be instructed in specific experiments. Table 1 shows the numbers of sequences in these datasets.

Besides, we use subsequences in experiments on dataset *Last.fm_1K*. Subsequence means the prefix of a standard sequence. For example, in a sequence $\{s_1, s_2, s_3, ..., s_{10}\}$, we use $\{s_1, s_2, s_3, ..., s_9\}$ as the known sequence and s_{10} as the label to predict or assess. With subsequences, the origin sequence could derive more sequences like $\{s_{-1}, s_1, s_2, ..., s_9\}$, $\{s_{-2}, s_{-1}, s_1, ..., s_8\}$, etc. The sequence items with subscript less than 0 are nonexistent and are filled with a fixed feature like 0.

4.1.2 Evaluation metrics. We use 5 kinds of evaluation metrics in this paper. The specific metrics used in every experiment will be stated.

Recall@*K* is the hit rate of cases whose true items are in the top-K recommended list. It is equal to the proportion of hit cases in all test cases. $hit(i)$ is 1 if case i hits, otherwise it's 0. Recall@K can be calculated by

$$Recall@K = \frac{\sum hit(i)}{|the\ test\ set|}. \tag{10}$$

Mrr@*K* calculates the sum of the reciprocal of rankings. Rankings larger than K would be regarded as infinite so the reciprocal is 0. So Mrr@K can be calculated by

$$Mrr@K = \sum \frac{1}{rank(i)}. \tag{11}$$

NDCG@*K*[25] is short for Normalized Discounted Cumulative Gain. It's based on relevancy. Items with high relevancy have a greater impact on the results. NDCG@K is higher when high relevancy items have better rankings.

AUC is the area under the receiver operating characteristic curve (ROC). The horizontal axis of ROC is the false positive rate and the vertical axis is the true positive rate.

Log loss is a frequently used metric in CTR prediction. It evaluates the accuracy of the result and can be calculated as

$$Log\ loss = -\frac{1}{N}\sum y_i \log(p_i) + (1 - y_i)\log(1 - p_i). \tag{12}$$

4.2 Comparison against Baselines

In this subsection, we compare CASFE with the following baseline models:

MostPop. Mostpop always recommends the items that people clicked or bought most for any user no matter which category he prefers.

GRURec[7]. This architecture is based on RNN for next item recommendation. It imports GRU units for sequence modeling and uses pair-wise loss function.

Caser[23]. It abandons RNN structures, proposing instead a CNN-based model. Caser uses horizontal and vertical convolutional filters to capture sequential patterns at point-level and union-level.

STAMP[17]. It is a short-term attention/memory priority model. STAMP could capture users general interests from the long-term memory of a session while capturing users current interests from the short-term memory of the last-clicks.

NextItNet[24]. This model is formed of a stack of holed convolutional layers, which can efficiently increase the receptive fields without relying on the pooling operation.

The embedding size is 100 here for all the 6 models including CASFE. Some important parameters for CASFE should be illustrated here. We use a 2-layer CNN. For datasets whose sequence length is less than 10, the kernel size is [3,2]. For datasets whose sequence length is greater than or equal to 10, the kernel size is [5,5].

The comparison result of CASFE with the 5 baselines are shown in Table 2. The model of the best performance on a certain dataset with a certain metric is in boldface. As we can see, in the numerical performance on any dataset with any metric, it is CASFE that almost always achieves the best result. There are only several metrics that STAMP or NextItNet reaches the best. MostPop just recommends the items

Table 2: Performance comparison on next item recommendation

Metric	Model	Music						MovieLens		
		m5	l5	l10	l20	l50	l100	n5	n10	n20
Recall@5	MostPop	0.0078	0.0031	0.0027	0.0031	0.0020	0.0013	0.0281	0.0238	0.0156
	GRURec	0.3269	0.2414	0.2564	0.2689	0.2633	0.2603	0.0820	0.1191	0.0898
	Caser	0.2812	0.2353	0.2623	0.2659	0.2534	0.2400	0.2188	03125	0.1875
	NextItNet	0.4050	0.2250	0.3326	0.3501	0.3477	0.3342	0.2812	0.2500	0.2500
	STAMP	0.4057	0.3223	0.3464	0.3582	0.3643	0.3407	**0.3438**	0.3750	0.2812
	CASFE	**0.4091**	**0.3940**	**0.3572**	**0.3651**	**0.3698**	**0.3744**	0.2812	**0.3750**	**0.2812**
Mrr@5	MostPop	0.0041	0.0006	0.0014	0.0011	0.0009	0.0008	0.0140	0.0104	0.0077
	GRURec	0.2593	0.1910	0.1863	0.1899	0.1851	0.1851	0.0442	0.0635	0.0529
	Caser	0.2354	0.2021	0.2123	0.2121	0.1932	0.1870	0.1719	0.1698	0.1562
	NextItNet	0.3302	0.2690	0.2844	0.2896	0.2988	0.2904	0.2005	0.2188	0.1469
	STAMP	0.3345	0.2675	0.2854	0.2948	0.3017	0.2812	0.2078	**0.2485**	0.1807
	CASFE	**0.3402**	**0.2749**	**0.2974**	**0.3012**	**0.3050**	**0.3085**	**0.2448**	0.2302	**0.2292**
NDCG@5	MostPop	0.0050	0.0012	0.0017	0.0016	0.0012	0.0010	0.0197	0.0137	0.0096
	GRURec	0.2762	0.2036	0.2037	0.2088	0.2046	0.2039	0.0536	0.0772	0.0621
	Caser	0.2465	0.2209	0.2248	0.2255	0.2082	0.2003	0.1841	0.2038	0.1644
	NextItNet	0.3489	0.2361	0.2980	0.3014	0.3106	0.3079	0.2207	0.2269	0.1722
	STAMP	0.3522	0.2812	0.3006	0.3106	0.3174	0.2965	0.2405	**0.2775**	0.2056
	CASFE	**0.3574**	**0.2882**	**0.3123**	**0.3172**	**0.3212**	**0.3250**	**0.2571**	0.2659	**0.2426**
Recall@20	MostPop	0.0203	0.0094	0.0057	0.0083	0.0061	0.0049	0.0703	0.0736	0.0579
	GRURec	0.4203	0.2997	0.3456	0.3587	0.3612	0.3592	0.2207	0.2773	0.2344
	Caser	0.3125	0.3101	0.3199	0.3313	0.3265	0.3042	0.3125	0.4375	0.4062
	NextItNet	0.4952	0.2309	0.4017	0.4215	0.4102	0.4055	0.4375	0.3750	0.4062
	STAMP	**0.5055**	0.3912	0.4240	0.4377	0.4417	0.4151	0.4062	0.5000	0.4688
	CASFE	0.5003	**0.3940**	**0.4296**	**0.4416**	**0.4456**	**0.4481**	**0.4375**	**0.5938**	**0.4062**
Mrr@20	MostPop	0.0052	0.0011	0.0017	0.0017	0.0013	0.0012	0.0177	0.0147	0.0114
	GRURec	0.2690	0.1970	0.1948	0.1996	0.1953	0.1951	0.0568	0.0787	0.0668
	Caser	0.2399	0.1996	0.2184	0.2189	0.2009	0.1937	0.1801	0.1860	0.1766
	NextItNet	0.3397	**0.3278**	0.2976	0.3053	0.3086	0.3061	0.2206	0.2292	0.1564
	STAMP	0.3448	0.2746	0.2933	0.3029	0.3097	0.2892	0.2167	**0.2560**	0.1938
	CASFE	**0.3497**	0.2817	**0.3049**	**0.3091**	**0.3127**	**0.3165**	**0.2572**	0.2510	**0.2418**
NDCG@20	MostPop	0.0084	0.0029	0.0026	0.0031	0.0023	0.0020	0.0290	0.0272	0.0212
	GRURec	0.3033	0.2205	0.2292	0.2359	0.2331	0.2325	0.0919	0.1219	0.1029
	Caser	0.2569	0.2373	0.2416	0.2446	0.2296	0.2190	0.2098	0.2434	0.2252
	NextItNet	0.3752	0.2529	0.3107	0.3189	0.3197	0.3184	0.2703	0.2606	0.2100
	STAMP	0.3812	0.3012	0.3230	0.3336	0.3398	0.3173	0.2614	0.3107	0.2530
	CASFE	**0.3839**	**0.3073**	**0.3334**	**0.3394**	**0.3431**	**0.3465**	**0.2952**	**0.3274**	**0.2784**

that are clicked or bought most without considering the users' features. There is no doubt that its metrics always reach the worst. Compared to GRURec and Caser, NextItNet and STAMP have distinct progress. On these bases, CASFE has a greater improvement.

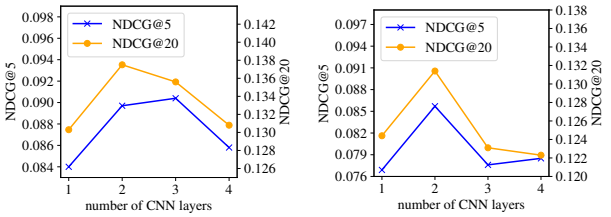
Caser and NextItNet also use CNN modules but they are weaker than CASFE. According to model structure, Caser and NextItNet only take the result of the last CNN layer. In this way, the explicit information in the middle layers is lost. Besides, they don't import attention mechanism to emphasize the difference among different items in a sequence. So we think these are the main reasons why Caser and NextItNet perform weaker than CASFE. As for STAMP, it does succeed in importing short memory and using a new attention network, but it's difficult for STAMP to capture the user's feature during some periods like CASFE. It performs the best among the 5 baselines.

4.3 Hyper-parameter Analysis

We perform experiments to study the impact of different hyper-parameters. In every group of the following experiments, we only change one parameter and keep others the same. The following experiments are performed mainly on *Music_m5*, *MovieLens_n5*, *MovieLens_n10* and *MovieLens_n20*. NDCG can reflect the overall effect better. So we use NDCG@5 and NDCG@20 here to compare the results.

4.3.1 The number of CNN layers. CNN is a main component of CASFE. To figure out the effect of the number of CNN layers C, we perform several experiments on dataset

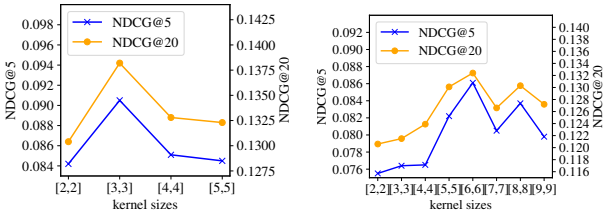
MovieLens_n10 and *MovieLens_n20*. The kernel sizes are all 3 in these experiments. The experiment results are shown in Figure 5. It shows that more CNN layers don't always lead to a better result. It is 2-layer CNN that can usually make the best result.



(a) MovieLens_n10 (b) MovieLens_n20

Figure 5: The performance with varying number of CNN layers. Two-layer CNN usually achieves the best result.

4.3.2 Size of kernels. The size of kernels indicates the length of the period phase we explore about user behaviors. These experiments are still performed on dataset *MovieLens_n10* and *MovieLens_n20*. There are two convolution layers, i.e., we would joint three attention results together. [3,3] means the kernels are $d \times 3$ and $d \times 3$. In Figure 6, [3,3] reaches the best on *MovieLens_n10*. But on *MovieLens_n20*, CASFE needs a longer kernel to get the best result. NDCG@5 and NDCG@20 keep rising from [2,2] to [6,6]. As the length of the sequence grows, the longer kernel may lead to a better result when trying to capture the periodic features of user behavior.

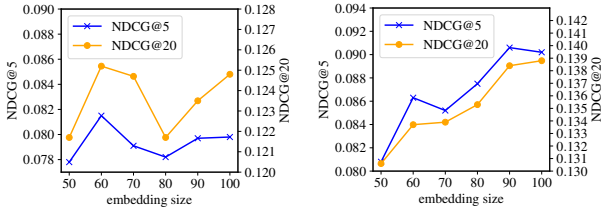


(a) MovieLens_n10 (b) MovieLens_n20

Figure 6: The performance with varying kernel sizes. Longer sequences may need longer filters.

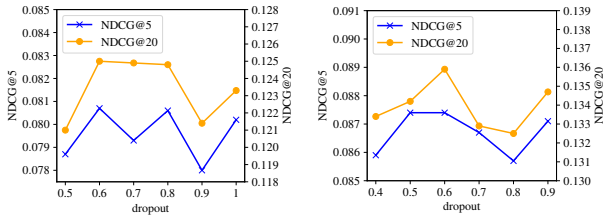
4.3.3 The embedding size. Figure 7 shows the variation of metrics with the embedding size d. Higher d doesn't lead to better results always. For the sequence length of 10, the best embedding size is 60. For the sequence length of 20, d of 90 reaches the highest result. Embedding size that is too high may lead to weak performance because of overfitting.

4.3.4 Dropout. During the learning process, dropout could optimize the artificial neural network by randomly ignoring some nodes of hidden layers. We also compare the experiment results with different dropout rates. The results are shown in Figure 8. The most proper dropout rate on this task is around 0.6.



(a) MovieLens_n5 (b) MovieLens_n10

Figure 7: The performance with varying embedding size d. Sequences of different lengths may need different embedding size.



(a) MovieLens_n5 (b) MovieLens_n10

Figure 8: The performance with varying dropout. The rate around 0.6 usually does the best.

4.4 Compatibility of CASFE on CTR prediction

As discussed in subsection 3.3, CASFE can make use of sequential data for the non-sequential CTR prediction task.

We joint the output of CASFE and the input of CTR prediction models to verify the compatibility of CASFE. Here we choose several CTR prediction models to prove the compatibility of CASFE. The following models are used:

IPNN [19]. PNN uses a product layer to capture interactive patterns between inter-field categories. IPNN is a kind of PNN that utilizes the inner product of vectors.

DNN [5]. This deep component is a feed-forward neural network. A data record (a vector) is fed into the neural network.

DeepFM [5]. This model consists of the deep component and the FM component that share the same input. The deep component is a feed-forward neural network and the FM component learns high-order feature interactions.

Experiments of models with and without CASFE are performed on dataset *MovieLens_1M* with different organization form, i.e., for the origin models IPNN, DNN and DeepFM we use *MovieLens_cb* which regards a single record as a case, and for the models with CASFE we use *MovieLens_ca* which regards a record to predict and several records before it as a case. AUC and Log loss are used as evaluation metrics. Table

Table 4: Attention visualization of CASFE for interpretability. Higher weights mean higher importance in the current layer. The experiment is performed on dataset MovieLens_n10.

movie id	1961	2020	527	318	593	296	2858	608	529	3006
movie poster										
movie category	Drama	Drama Romance	Drama War	Drama	Drama Thriller	Crime Drama	Comedy Drama	Crime Drama Thriller	Drama	Drama
attention weights 1	0.115	0.105	0.056	0.130	0.097	0.099	0.149	0.069	**0.180**	
attention weights 2	**0.210**	0.153	0.152	0.198	0.149	0.138				
attention weights 3	0.381	**0.469**	0.150							

Table 3: Compatibility of CASFE on CTR prediction

Model	AUC	Log loss	Improvement
IPNN	0.8010	0.4327	2.2‰
IPNN+CASFE	**0.8028**	**0.4253**	
DNN	0.8018	0.4381	1.4‰
DNN+CASFE	**0.8027**	**0.4301**	
DeepFM	0.8059	0.4191	5.8‰
DeepFM+CASFE	**0.8106**	**0.3837**	

3 shows the experiments results. The *Improvement* column is the ratio of AUC improvement brought by CASFE to AUC of the origin model. As we can see, the three models are all improved with CASFE. Though the improvement is not very large, it may be able to bring an excellent increase in online CTR [3, 13]. Compared to dataset *MovieLens_cb*, dataset *MovieLens_ca* has the same items to predict, but the sequential information is added through CASFE. Therefore, CASFE could improve the performance of the three typical CTR prediction models obviously. This fact proves that CASFE has good compatibility and is successful in extracting features from sequential data.

4.5 Interpretability of CASFE

As mentioned before, CASFE has good interpretability. Attention networks in CASFE introduce a good opportunity for providing explanations for such deep models. Because the height of kernels is equal to the embedding size, the convolution in CASFE is like one-dimension convolution more. Therefore, the results of different layers of CNN could be interpreted as the features of users' behaviors with different periods. For example, with a $d \times 3$ kernel as f^1 and a $d \times 2$ kernel as f^2, we can regard c^1 as the feature of every single day, c^2 as the feature of every three days with one day as the stride and c^3 as the feature of every six days with three days as the stride. After their respective attention network, r^1, r^2 and r^3 are jointed together as rs. In this way, the module is able to collect the features of user behaviors with one day, three days and six days as the periods. Furthermore, the result of the last layer of the attention network shows the difference among items in the responding convolution result.

In order to make the interpretability of CASFE comprehensible, we choose a representative case from dataset *MovieLens_n10* in the process of test. As shown in Table 4, we record the id, poster, and category of the ten movies in the sequence case and the attention weights of them except the last movie. The last three rows are the attention weights of different CNN-layer results. In attention weights 1, the weights of movie 2858 and movie 529 are the highest. But they don't appear in the highest one in attention weights 2 that concludes the first four movies. 0.210 is the highest weight in attention weights 2, but the highest one in attention weights 3 indicates the four middle values 0.153, 0.152, 0.198 and 0.149. Besides, the three continuous movies with the id of 2020, 527 and 318 don't have the highest weights in attention weights 1 but they all appear in the highest values of the last two attention weights. Considering the keynote of the three movies, they all try to explore the deep part of human nature and soul. A single one of them doesn't contribute to the user's feature most, but three continuous movies with similar keynote do need to be taken seriously. The recommended movie is *Good Will Hunting*, which also

has a similar keynote. These phenomena have proved that the sequence features presented in different convolution layers are different. But the conventional CNN-based models only make use of the result of the last convolution layer usually. In CASFE we joint the results of all the convolution layers to ensure that the information of any layer wouldn't be lost.

In this way, the CNN in CASFE not only emphasizes the receptive field but also succeeds in exploring features of the user behaviors with different time periods. This characteristic is very beneficial for the situation that users' behaviors have strong periodicity. From the perspective of feature engineering, CASFE can extract the useful feature during some different time periods automatically.

5 CONCLUSION

In this paper, we propose a CNN & attention-based sequential feature extractor (CASFE) module. The information of all CNN layers is used in order to capture the periodic features of user behavior. The deeper layer corresponds to longer time periods. We apply attention mechanism after each CNN layer to focus on the important features. At last, the results of each attention mechanism after the corresponding CNN layer are jointed together and inputted into MLP or other components to extract the user feature further. The experiments results against baselines on dataset *Last.fm_1K* and *MovieLens_1M* prove that CASFE has better performance on next item recommendation task. It also has good interpretability which can be proved according to the attention weights. Besides, as a compatible module, CASFE can also be applied in CTR prediction with models like DeepFM. With CASFE, sequential information can be used to the CTR prediction task so the origin models would be enhanced. In a word, CASFE is an accurate, interpretable and compatible module for sequential feature extracting and has better performance than existing models on the same tasks.

ACKNOWLEDGMENTS

This work is partially supported by National Natural Science Foundation of (61772528) .

REFERENCES

[1] Ossama Abdel-Hamid, Abdel rahman Mohamed, Hui Jiang, and Gerald Penn. 2012. Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition. In *ICASSP*. 4277–4280.
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. (2014).
[3] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. *CoRR* (2016).
[4] Yu Feng, Liu Qiang, Wu Shu, Wang Liang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *SIGIR*.
[5] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. (2017), 1725–1731.
[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. (2015).
[7] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *Computer Science* (2015).
[8] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. 2016. Densely Connected Convolutional Networks. (2016), 2261–2269.
[9] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. *CoRR* abs/1808.09781 (2018), 197–206.
[10] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* (2009), 30–37.
[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.
[12] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *CIKM*.
[13] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *WWW*.
[14] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-Aware Sequential Recommendation. In *ICDM*. 1053–1058.
[15] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the Next Location: A Recurrent Model with Spatial and Temporal Contexts. In *AAAI*. 194–200.
[16] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. 2015. A Convolutional Click Prediction Model. In *CIKM*. 1743–1746.
[17] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *SIGKDD*. 1831–1839.
[18] Xia Ning, Christian Desrosiers, and George Karypis. 2015. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. In *Recommender Systems Handbook*. 37–76.
[19] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-based Neural Networks for User Response Prediction. (2016), 1149–1154.
[20] Ruslan Salakhutdinov and Andriy Mnih. 2007. Probabilistic matrix factorization. In *NIPS*. 1257–1264.
[21] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grgoire Mesnil. 2014. A Latent Semantic Model with Convolutional-Pooling Structure for Information Retrieval. In *CIKM*. 101–110.
[22] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. *CoRR* (2016).
[23] Jiaxi Tang and Wang Ke. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*. 565–573.
[24] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *WSDM*. 582–590.
[25] Fajie Yuan, Xin Xin, Xiangnan He, Guibing Guo, Weinan Zhang, Tat-Seng Chua, and Jose M. Joemon. 2018. fBGD: Learning Embeddings From Positive Unlabeled Data with BGD. In *UAI*. 198–207.
[26] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. 2019. Deep Interest Evolution Network for Click-Through Rate Prediction. *CoRR* (2019), 5941–5948.
[27] Guorui Zhou, Xiaoqiang Zhu, Chengru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep Interest Network for Click-Through Rate Prediction. In *KDD*. ACM, 1059–1068.
[28] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. 2017. What to Do Next: Modeling User Behaviors by Time-LSTM. In *IJCAI*. 3602–3608.