Improving Native Ads CTR Prediction by Large Scale Event Embedding and Recurrent Networks

Mehul Parsana
Microsoft Corporation
Redmond, WA
mparsana@microsoft.com

Krishna Poola
Microsoft Corporation
Redmond, WA
krishnap@microsoft.com

Yajun Wang
Microsoft Corporation
Sunnyvale, CA
yajunw@microsoft.com

Zhiguang Wang*
Microsoft Corporation
Redmond, WA
zhigwang@microsoft.com

ABSTRACT

Click through rate (CTR) prediction is very important for Native advertisement but also hard as there is no direct query intent. In this paper we propose a large-scale event embedding scheme to encode the each user browsing event by training a Siamese network with weak supervision on the users' consecutive events. The CTR prediction problem is modeled as a supervised recurrent neural network, which naturally model the user history as a sequence of events. Our proposed recurrent models utilizing pretrained event embedding vectors and an attention layer to model the user history. Our experiments demonstrate that our model significantly outperforms the baseline and some variants.

ACM Reference Format:
Mehul Parsana, Krishna Poola, Yajun Wang, and Zhiguang Wang. 2018. Improving Native Ads CTR Prediction by Large Scale Event Embedding and Recurrent Networks. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Native advertisement is a new type of online advertisement which attracts significant attention in recent years, especially due to the effectiveness of Facebook ads [2], Yahoo Gemini [4] and Bing Intent Ads [1]. In native ads scenarios, the advertisements replicate the look and feel of the organic contents and are placed interchangeably with them. It has been very successful in providing good user experience and effective return of investment for advertisers [3].

One distinctive characteristic of native ads, comparing with traditional search ads, is the lack of strong user query intent. In search ads, the search query from user input plays a dominant role in optimizing the serving system. In native ads, however, it is important to infer users' intent from their history of online activities.

We propose an event embedding scheme to map the events from users' browsing activities we collected from our advertisers to a latent space. The browsing history consists of URLs the user visited as well as titles and simple descriptions. With our event embedding scheme, the sparse activity data is embedded into a fixed length vector with weak supervision by the user browsing history only.

We apply our event embedding scheme to the native ads click through rate prediction problem. In each request, for each advertisement that is selected, we need to compute the click through rate, i.e., how likely the advertisement will be clicked by the user in this request. The CTR estimation is very critical in selecting and ranking the advertisements as well as downstream optimization processes. We employ recurrent neural network models to predict the CTR from users' browsing activities. We show that by applying the event embedding with our designed recurrent attentional model, we can achieve the performance that is significantly better than the baseline and other variants.

2 RELATED WORK

Deep neural networks has been applied in personalized recommendation systems. He et al [14] combine neural networks and matrix factorization machine to model user-item interactions. They model the user's preference implicitly by generating the latent vector jointly with the item matrix with a supervised loss. Long short-term memory (LSTM) based sequence modeling is applied in [22] for news recommendation. They generates user representations with an recurrent neural networks from user news browsing histories. Their embedding layer consists of a denoising autoencoder [28]. The embedding layer is not applicable in our setting as we do not have rich content information for the users' past browsing activities. Deep interest network (DIN) [31] is proposed to estimate click through rates for advertisements in e-commence site. Their event embedding layer is trained together with the user modeling. Also their model is not sequential based, as their model cannot learn from long term history. Recurrent Neural Network is used in [30] for click prediction in search ads scenario with only users search history.

There are limited work in user browsing event embedding especially in the context of modeling user history. Our approach is inspired by the word2vec [19] and the metric learning with Siamese

*The authors are listed in alphabetical order.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

networks [9], which is successfully applied in a lot of natural language processing and computer vision tasks. In particular, the word vector is usually trained outside of the actual NLP tasks. The framework of word embedding is also utilized to train the item vector in achieving good performance in genre classification for Windows 10 applications [6]. Their embedding is ID based, thus unable to handle unseen items. The model also discards the temporal information which is very important in our setting. A character level CNN is proposed to learn a URL embedding in to detect malicious URLs [26]. It trains the embedding using a binary classifier specific to the task and hence is not applicable in our problem. DSSM is proposed in [16] to train the word embedding through click on the query-document pairs, which usually requires huge number of labeled data.

Piece-wise Linear Models (LS-PLM) [12] and factorization machine (FM) [25] models can be viewed as a class of networks with one hidden layer, which first employs embedding layer on sparse inputs and then imposes special designed transformation functions for output, aiming to capture the interactive relationships among features. As the scale of feature and sample becoming larger and larger, the CTR prediction model has evolved from shallow to deep structure in recent years. Wide&Deep [7] and the YouTube Recommendation CTR model [10] extend the idea of factorization machine by replacing the transformation function with feed forward networks, which improve the model capability. They follow a similar model structure that stacking an embedding layer before fully connected layer with pooling layer to integrate the multi-dimension hidden vectors, which is now more likely to be the de facto standard. Our baseline model follows this kind of model structure.

3 LATENT EVENT REPRESENTATION

As mentioned earlier, one crucial component in our models is an event embedding scheme that embeds each user browsing event into a fixed dimensional space. In this section, we describe the scheme in detail.

For each user browsing activity e_u^t from user u at time t, we define it as a triplet

$$e_u^t = \{U, T, D\},$$

where U is the URL, T is the title of the URL page and D is a simple description where we received from our advertiser websites. A user session $S = \{e_1, e_2, e_k\}$ is defined as a sequence of events the user visits in a session. [1] We train a Siamese network [9] to embed events $\{e_t\}$ into a low dimensional latent space.

3.1 Network Structure

The Siamese networks learn the latent embeddings of the user events by enforcing the distance of two consecutive events to be close. The input are events $\{e_i, e_j\} \in S$ which are visited by one user consecutively in a session S. We employ negative sampling on random event pairs $\{\tilde{e_i}, \tilde{e_i}\}$ to get the same number of negative samples.

The Siamese networks consists of two identical neural networks, each taking one of the two input events. The last layers of the two networks are then fed to a contrastive loss function , which calculates the similarity between the two events.

Our networks is formulated as:

$$\begin{aligned} \tilde{X_i} &\sim I(e_i) \qquad e_i = \{U_i, T_i, D_i\} \\ \tilde{X_j} &\sim I(e_j) \\ h_i &= f(W\tilde{X_i} + b) \\ h_j &= f(W\tilde{X_j} + b) \\ \theta &= \arg\min L(h, X)) \end{aligned}$$

where I is the embedding function. We describe it in detail as follows.

We use similar mechanisms to vectorize U, T and D. For URL U, we break it into fixed number K tokens, e.g., 20 tokens. (The domain is a separate token. If there are less than K tokens, we pad them with some null values.) Each token is then hashed into a fixed large space M, e.g., 10^5. We maintain a dictionary of size M, in which each value is an embedding vector of fixed dimension L. For each selected token in U, we extract its corresponding embedding value from the dictionary and we pad K such vectors into a final vector to represent U. So URL U is embedded into a vector of dimension $K \cdot L$. We apply the same mechanism to T and D with different embedding dictionaries. The final embedded vector is of dimension $3 \cdot K \cdot L$. Some characteristic of this embedding mechanism:

- **Unseen events.** Since our embedding is based on tokens, we can naturally handle unseen events, e.g., with new URLs. (Some websites use GUID in the URLs to make every visit with unique URL.)
- **Same tokens.** The same token appears in URL, title and description might be embedded into different vectors.
- **Cross-site correlation.** We are able to understand the similarity between different websites from the tokens in the URLs, titles and descriptions.

The hidden representation, h, is mapped from the embedding through the network with the activation function, $f(\cdot)$ as ReLu [21]. The weight matrix W and bias b are shared across the input pairs.

The contrastive loss function, $L(\cdot, \cdot)$ is defined as

$$\begin{aligned} L(h, X) & \qquad (1) \\ &= \frac{1}{2} \cdot y \cdot D(h_i, h_j) \\ &+ \frac{1}{2} \cdot (1 - y) \cdot \max\{0, m - D(h_i, h_j)\} \end{aligned}$$

$D(h_i, h_j)$ is the distance between the embedding pair $\{h_i, h_j\}$. Distance function D can be customized to fulfill specific tasks. Here we use $L2$ distance. m is a margin value which is greater than 0. Having a margin allows us to discard the loss from pairs are too far away and focus on the negative sample pairs that are marginally close in the current latent space.

y is the label to indicate if the input pair is a positive sample, i.e., visited consecutively in one user session, or a negative one, i.e., sampled randomly from all events. Notice our loss function will enforce the embedding to pull consecutive events together, in the same time to push negative sampled events pairs to be further apart in the latent space.

The hidden representation layer h and the embedding layer I both learn the semantic representations of the user events. Another

[1] In practice, we take all consecutive events that are within 30 minutes apart as a session.

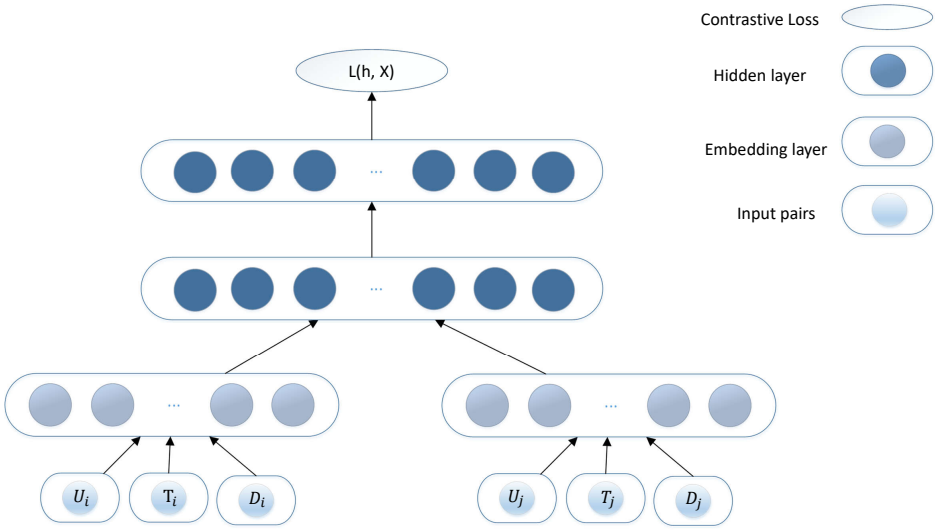


Figure 1: Siamese network structure to learn the event embedding.

benefits of our Siamese network structure is, whenever we need the embedding, we can easily plug the whole networks as a module into the big neural network framework. To keep the contribution clear and facilitate the evaluation, we use outputs of the the embedding layer I as the event vectors.

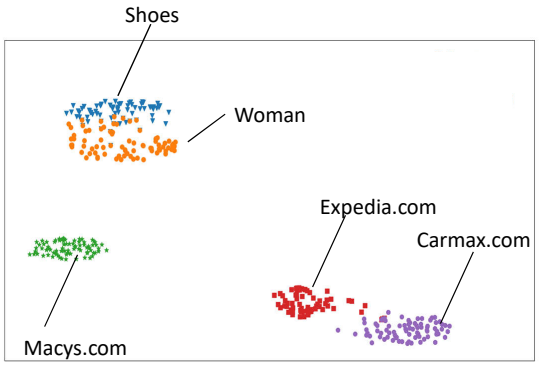
The Siamese network structure is illustrated in Figure 1. We use two hidden layers instead of one in practice to leverage the different data sources, which give us lower loss during training.

Another option is to train the event embedding with word2vec e.g., Skip-gram model [20]. However, we do not see meaningful improvement over the Siamese networks. We plan to further investigate in word2vec based embedding approaches.

3.2 Qualitative Evaluation

It is a very challenging task to measure the quality of our event embedding scheme. We qualitatively inspect the embedding results as follows. As the user behavior data provide no category information, we manually selected several pivots (or say, centroid). For each pivot, we retrieve the relevant embedding representations and find the top-k nearest neighbors in the embedding space. This is motivated by the assumption that a useful representation would cluster the domain/tokens with similar semantic contents. We generate a subset that contains the top 80 events for selected pivots. We applied t-SNE [18] with a $L2$ norm kernel to reduce the dimensionality of the item vectors to 2. Then, we color each pivots and their neighbors.

As shown in the left graph in Figure 2, every selected pivots and their associated clusters are clear and well defined. 'Shoes' and 'Woman' are close to each other with some overlap, which is reasonable as the shoes are a quite big category in female consumer goods. 'expedia.com' and 'carmax.com' are also near with each other. Moreover, the nearest neighbors trained have strong coherence. We pulled the neighbors beyond the top 100 in each



Event embedding co-trained with the event triplet $E(U, T, D)$.

Figure 2: Visualization of embeddings of the user events.

embeddings, the neighbors are still showing reasonable correlation to the pivots.

In Table 1, we show the examples of the selected pivots and their top nearest neighbors. The URL domains and words are clustered respectively although we trained them by projecting onto a common hidden layer together. The embeddings are learning the semantics that encodes the user browsing interest and intentions well, like Glove [24] or DSSM [16]. Especially, the embeddings learns the specific patterns in user browsing history, like for the centroid 'woman', 'hair', 'beauty' and even 'sale' are among the closest neighbors as they also appear often when the user click and browse. Note that our method and scenario are correlated but also different with DSSM.

Table 1: Demo of the event embedding results. Given a pivot domain or words, we extract their nearest neighbor

Pivot	Nearest Neighbors
www.carmax.com	www.edmunds.com, www.toyota.com, www.subaru.com, www.carfax.com, www.mbusa.com, automobiles.honda.com, www.chevrolet.com
www.macys.com	www.ebates.com, www.jcpenney.com, www.sephora.com, www.target.com,www.ulta.com, www.potterybarn.com, www.gap.com, www.urbanoutfitters.com, www.dsw.com, www.overstock.com
www.expedia.com	www.kayak.com, www.priceline.com, www.orbitz.com, www.hotels.com, www.starwoodhotels.com www.travelocity.com, usa, www.delta.com, www.airbnb.com
Woman	petite, lace, kids, gifts, baby, christmas, dresses, hair, plus, beauty, shoes, pants, boots, makeup, outerwear, girls, checkout, boys, sale, shoe
Shoes	boots, women, womens, apparel, mens, dress, clothing, shirt, kids, jeans, jackets, beauty, shirts, wool, sets, face, checkout, jacket, girls, black

- DSSM trains the embedding using query-document pairs. It contains strong supervision from the search query and clicks. In native ads, as we have no related query, our methods trained the embedding that is weakly supervised by the pure user browsing history, which is more realistic and adequate in native advertising.
- DSSM is able to represent short sentences as the query-answer pair always appears in such continuous word sets. In native ads, either the URLs, title or short description in the user browsing history has meaningful syntactic structures. Thus we proposed the Siamese network structure instead of Skip-gram. Our method is able to embed both the words and URL domains.

4 CLICK THROUGH RATE PREDICTION

In this section, we apply our embedding results to the CTR prediction problem for native ads. In particular, for each request $\{E_u^{-t}, a_t\}$ where E_u^{-t} is the sequence of activities for the user u up to time t and a_t is the advertisement we displayed to user u. Our objective is to predict the probability that user u will click on a_t in this request. Figure 4 outlines the information flow.

We describe several variations to model user activities from the browsing history of the user as the input to the CTR prediction tasks. First, we formulate our problem and a simple word-based baseline method and discuss the issues that they have. We then describe our methods of using latent embedding of user events, as was explained in the previous section.

4.1 Problem Formulation

Let E_u be the entire sequence of the events for user u and E_u^{-t} be the sequence of events of user u up to time t. We are given a set of triplets $\{< u_t, a_t, y_t >\}$, that user u_t is served with advertisement a_t at time t. y_t indicates that the u_t clicked on a_t (if 1) or not (if 0). Our objective is:

$$\arg\min \sum_t L_\theta(y_t, \tilde{y}_t; E_{u_t}^{-t}, a_t) \tag{2}$$

Where L is the loss of predicted click $\tilde{y}$ and the real click y given user browsing events $E_{u_t}^{-t}$ up to time t. θ is the model parameters.

Based on this formulation, the most important part of the model is to represent the event history E_u^{-t} for user u. In the following sections, we describe a few different choices to model the user event history.

4.2 Single Event Representation

We explore three different event representations.

(1) **Bag of Word (BoW).** The bag-of-word based model introduced in [22] are proposed in our experiment as the baseline. An event $e = \{U, T, D\}$ is represented by a collection of words tokenized from the triplets. The user state is learned from a collection of words included in the last event (last-1 event baseline) before time t or a fixed number of events (last-N event) of the user before time t. We use a feed forward networks to learn a non-linear mapping on the collection of the words to represent the user history.
(2) **Co-trained Event Embeddings (Embedding).** We use the network structure in Section 3 to embed the events. However, the weights of the embedding layers is co-trained together with the overall network for our CTR task.
(3) **Pretrained Embedding.** The embedding vector is pretrained by the Siamese networks in Section 3.

BoW is simple and fast to calculate, but suffers from the sparseness of the representation. The co-trained event embedding has a large number of weights and hence the model complexity is very high. In our CTR prediction task, the labeled data is much smaller than the users' browsing history, which makes the co-training of the embedding vector a lot challenging. On the other hand, the pre-trained event embedding is trained in self-supervised manner with sufficient labeled data. As we can see later, pretrained embedding has more generalization power.

Now we describe how to represent users' history based on the single event representations.

4.3 Bag-of-Events Model (BoE)

Following the popular model structure introduced in [10, 27, 32], our base bag-of-events model has two major steps:

(1) Use one of the single event embedding approach in Section 4.2.

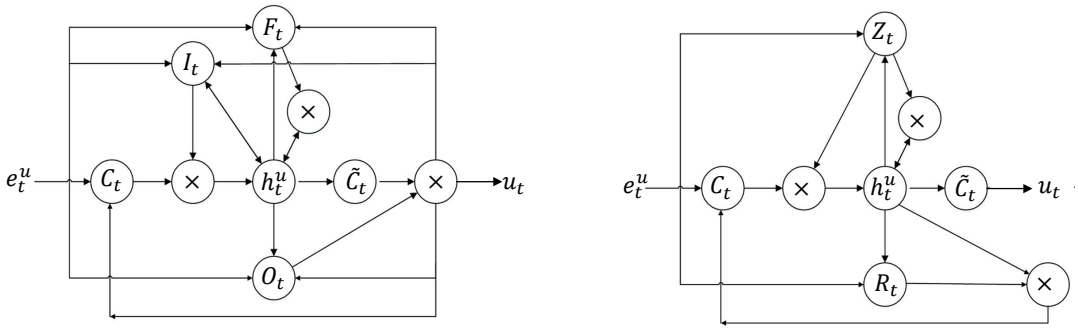


Figure 3: LSTM unit (left) and GRU unit (right).

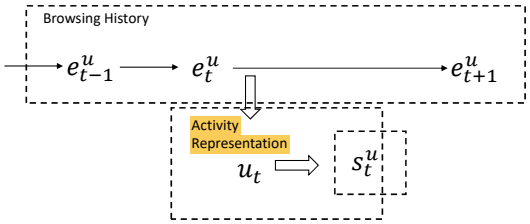


Figure 4: Flowchart of browsing history and clicks.

(2) Apply a feed forward networks with ReLU to fit the click response. Notice that the input contains multiple user events, we add an averaging pooling layer to summarize the sequence and get a fixed size vector.

We use BoE(K) to represent the bag of events model with last K events included before the pooling layer.

A decay model is proposed in [22] as a baseline where α is a parameter vector that represents the strength of time decay. However, the decay model is not showing any gain in their experiments. Since our (attentional) recurrent model is focusing on learning the temporal dependencies, we skip this variation in our experiments.

4.4 Recurrent Neural Network Models

Given that the user's history consists of an event sequence, it is natural to consider recurrent neural networks (RNN) to model the user. More specifically, in such RNN networks, the user state s_u^t is determined by the event e_t^u and previous state s_u^{t-1} as:

$$s_u^t = f(e_u^t, s_u^{t-1}).$$

In order to learn long temporal dependencies and avoid gradient vanishing and explosion problems, we employ two popular RNN networks: the long short-term memory networks (LSTM) [15] and the gated recurrent networks [8].

4.4.1 Long short-term Memory Network. The LSTM unit is formulated as

$$\begin{aligned}
I_t &= \phi(W_I^{in} e_t^u + W_I^{out} u_{t-1} + W_I^V h_{t-1}^u + b_I) \\
F_t &= \phi(W_F^{in} e_t^u + W_F^{out} u_{t-1} + W_F^V h_{t-1}^u + b_F) \\
O_t &= \phi(W_O^{in} e_t^u + W_O^{out} u_{t-1} + W_O^V h_{t-1}^u + b_O) \\
C_t &= \tau(W_C^{in} e_t^u + W_C^{out} u_{t-1} + b_C) \\
h_t^u &= I_t \odot C_t + F_t \odot h_{t-1}^u \qquad (3) \\
\tilde{C}_t &= \tau(W_{\tilde{C}}^V h_t^u + b_{\tilde{C}}) \qquad (4) \\
u_t &= O_t \odot \tilde{C}_t \qquad (5)
\end{aligned}$$

where $\phi(\cdot)$ is element-wise sigmoid function. h_t^u is the memory state. The information flows from the input of the user browsing browsing activity to output (user state). The input e_t^u is encoded using pretrained event embedding vectors in section 3 or can be randomly initialized and trained with the CTR networks. The event vector flow merged into the previous hidden and decoded to the event vector space as the user state.

The browsing information flow is controlled by three gates, input gate I_t, forget gate F_t, and output gate O_t. The idea is that the input gate filters unnecessary input browsing activity to construct a user state, for example the concept drift due to the sudden interest change. The forget gate represents the decline in interest by the user, which is actually a more adaptive time decay effect than the exponential discount factor. The output gate filters the contents that should not be focused on in the next session.

4.4.2 Gated Recurrent Network. The gated recurrent unit (GRU) was proposed in [8] to make each recurrent unit adaptively capture the dependencies of different time scales, which is another effective learning structure to avoid gradient vanishing and explosion. The

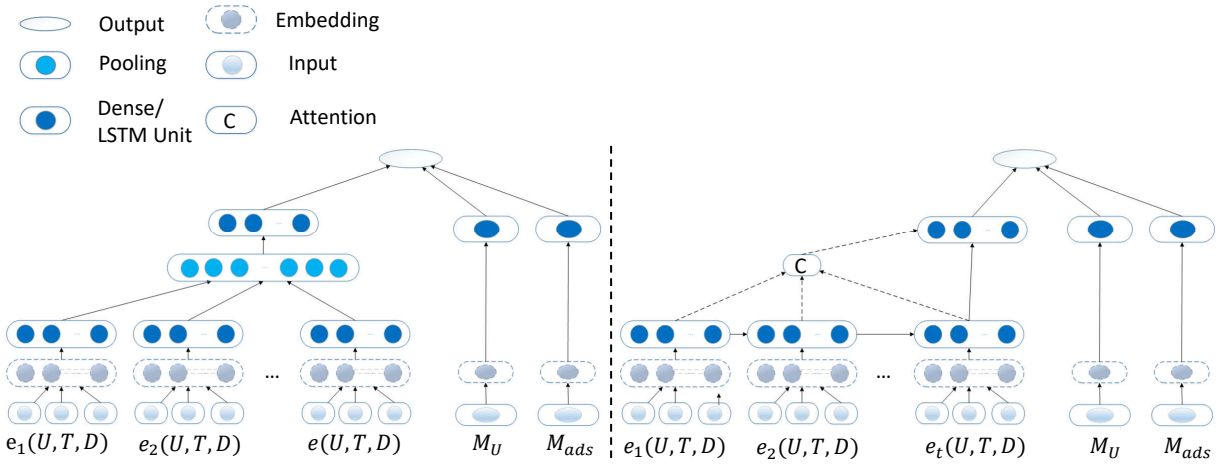


Figure 5: Network structure of the bag-of-events and the recurrent (attentional) model.

GRU unit is formulated as:

$$Z_t = \phi(W_Z^{in} e_t^u + W_Z^V h_{t-1} + b_Z)$$
$$R_t = \phi(W_R^{in} e_t^u + W_R^V h_{t-1} + b_R)$$
$$C_t = \tau(W_C^{in} e_t^u + W_C^{out}(R_t \odot h^{t-1}) + b_C)$$
$$h_t^u = Z_t \odot C_t + (1 - Z_t) \odot h_{t-1}^u \quad (6)$$
$$\tilde{C}_t = \tau(W_{\tilde{C}}^V h_t^u + b_{\tilde{C}}) \quad (7)$$
$$u_t = \tilde{C}_t \quad (8)$$

The important difference is that the gate Z_t in this model plays the role of two gates, i.e., input and forget gate in the LSTM-based model, which happens to make the upper bound of the hidden state h as $\max(||h_0^u||_\infty, sup_x|\tau(x)|)$. Thus, for any length of the long input sequences, the upper bound tends to be limited by a constant. In practice, LSTM model is always trained with gradient clipping [23] where GRU unit is always trained without clipping. Figure 3 shows the structure of LSTM unit and GRU unit.

Note that except for one original GRU layer, we add another fully connected layer as suggested by [22], which makes the network easy to adopt the proposed attention model.

4.4.3 Attention Network. Attention model is proposed in [5, 13] in machine translation and sequence learning tasks, and later successfully applied to recommendation systems [29].

We build a context vector c_i that depends on a sequence of annotations $(h_1, \cdots, h_t)$ to which a recurrent encoder maps the input browsing events. Each h_i contains information about the whole input sequence with a strong focus on the parts surrounding the i-th event in the input sequence. To better capture the different contribution scale of various contextual events, the attention vector learns the integration weights on each E_i instead of using the last state only.

The context vector c_i is computed as the weighted sum of the state sequences h_i

$$c_i = \sum_{j=1}^{t} \alpha_j h_j \quad (9)$$

The weight α of state h_j is computed by

$$\alpha_{ij} = \frac{\exp e_j}{\sum_{j=1}^{t} \exp(e_j)}$$
$$e_j = \tau(h_j)$$

Where τ is the hyper-tangent function. The context vector directly computes a soft attention, which allows the gradient of the cost function to be back-propagated through. Indeed, the context vector is a weighted sum of all the annotations as computing an expected annotation, which assign the probability that the target event e_i^u is assigned to the final user states. All the three models are illustrated in Figure 5.

4.5 Experiments and Results

4.5.1 Dataset. To train the Siamese network for event embedding, we sampled approximately one week user browsing history and build about 600 million event pairs which has been visited consecutively from our browsing data in November, 2017. To train the CTR model, we random sampled about 100 million browsing and click entries. About 60 million is used as training data, leave the remaining 40 million as validation and test set evenly.

For CTR prediction, we extract the last $K = 10$ browsing events before the clicks/non-clicks on the advertisement to model the user intention. We also test the baseline that only uses the last event. The user meta data M_u, e.g., user ID and the advertisement meta data M_{ads}, e.g., title, description, URL are used to predict the click. The full dataset has about 0.36% events has label 1 (clicked).

4.5.2 Evaluation Metrics. To measure CTR prediction quality, we compute the area under receiver operator curve (AUC) [11] as the labels/clicks are highly skewed. For the clicks in test set, we calculate AUC using the predicted ranking formed by the estimated click probability output by the model and the ground-truth label, which is the presence or absence of clicks on the native advertisement. We report final AUC as an average and standard deviation over five runs.

4.5.3 Experiment Setting and Result. We fully compare different baseline settings and our proposed recurrent models with pretrained event embeddings. The embedding vector size are fixed at 128. All the dense layer have 256 units. We trained all the neural networks with Adam [17] with the initial learning rate was 0.0001 and mini-batch is 128. The dimensions of internal state h_t^u is 256 in both LSTM and GRU. Dropout ratio is 0.5 at the last layer in all the neural networks models to alleviate overfitting. These parameters are determined using the validation dataset. We used the gradient clipping technique with bound 5 to avoid gradient explosion in LSTM. GRU did not cause gradient explosion when this technique was not used in these experiments.

Table 2: Results of the click prediction. Values indicate average of metrics and 99% confidence intervals in five runs.

Model	AUC
BoW - BoE(1)	0.623 ± 0.004
BoW - BoE(10)	0.631 ± 0.005
Pretrained embedding - BoE(1)	0.630 ± 0.005
Pretrained embedding - BoE(10)	0.642 ± 0.006
Embedding - LSTM	0.679 ± 0.005
Embedding - GRU	0.683 ± 0.004
Pretrained embedding - LSTM	0.695 ± 0.003
Pretrained embedding - GRU	0.697 ± 0.002
Pretrained embedding - LSTM - attention	**0.699 ± 0.005**
Pretrained embedding - GRU - attention	**0.708 ± 0.004**

Table 2 lists all the AUC metrics with different experiment settings. BoE is short the Bag-of-Events model. The brackets (1) indicates that we use the only last user event to model the user activities. 'Embedding' means we randomly initialize and train the embedding within the click prediction training procedure. We also test the case to use the pretrained embedding to represent the user events.

For the Bag of Events models, the model with last 10 events improves over the single event case. Also by employing the pretrained embedding, we are able to improve the AUC slightly further.

In the second half of Table 2, RNN based models, e.g., LSTM and GRU achieve significantly better results than Bag-of-Events models. This indicates that the RNN networks generally can better summarize sequential user activities. We also observe that the pretrained embedding is able to improve the AUC for both LSTM and GRU models. The attention layer can further improve the performance. We notice that GRU models consistently outperform LSTM networks in all our settings.

In Table 3, we explore how the length of the user activities would impact the performance of the models. We see if the length is short, the performance is slightly worse due to the insufficient learning of the user states. However, the models hardly benefit from the long user history beyond 10 events. We plan to investigate further how to model long term history more effectively.

Table 3: AUC score of the recurrent model with different user event history length.

Model	AUC
Pretrained embedding - LSTM - attention (5)	0.695 ± 0.002
Pretrained embedding - LSTM - attention (10)	0.699 ± 0.003
Pretrained embedding - LSTM - attention (20)	0.698 ± 0.003
Pretrained embedding - GRU - attention (5)	0.702 ± 0.004
Pretrained embedding - GRU - attention (10)	0.711 ± 0.004
Pretrained embedding - GRU - attention (20)	0.711 ± 0.003

One interesting observation we have is the generalization power of the pretrained event embedding model. From Table 4, while training AUC decreases with pretrained embedding vectors, the testing AUC actually increases. This matches our intuition as we could have overfitting issue when co-training the event vectors.

Table 4: The difference of the mean AUC score between training and test with different models.

Model and AUC	Training	Test
Embedding - GRU	0.746	0.683
Pretrained embedding - GRU	0.735	0.697
Pretrained embedding - GRU - attention	0.741	0.711

5 CONCLUSION

In this paper, we propose a large-scale event embedding scheme to encode the user browsing history by training a Siamese network with weak supervision on the user consecutive events. With the pretrained embedding, we explore recurrent neural network models to model the user history, thus to improve the CTR prediction in native advertising.

We found that the our embedding scheme is able to learn reasonable semantics of both domains and words by training from the URL, title and description of the user events.

We also show the improvements in the native advertising CTR prediction task by using recurrent neural network models with the pretrained embedding and attention layer to model the user history. Our experiments showed that the event embedding and our designed recurrent attentional models significantly outperform the baseline and other variants.

REFERENCES

[1] [n. d.]. Bing Intent Ads. https://advertise.bingads.microsoft.com/en-us/solutions/ad-products/bing-intent-ads. ([n. d.]).
[2] [n. d.]. Facebook Ads. https://www.facebook.com/business/products/ads. ([n. d.]).
[3] [n. d.]. NATIVE ADS VS BANNER ADS. https://www.sharethrough.com/resources/in-feed-ads-vs-banner-ads/. ([n. d.]).
[4] [n. d.]. Yahoo Gemini Ads. https://gemini.yahoo.com/advertiser/home. ([n. d.]).

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

[6] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *Machine Learning for Signal Processing (MLSP), 2016 IEEE 26th International Workshop on*. IEEE, 1–6.

[7] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. ACM, 7–10.

[8] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[9] Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 539–546.

[10] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.

[11] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[12] Kun Gai, Xiaoqiang Zhu, Han Li, Kai Liu, and Zhe Wang. 2017. Learning Piecewise Linear Models from Large Scale Data for Ad Click Prediction. *arXiv preprint arXiv:1704.05194* (2017).

[13] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 173–182.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[16] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*. ACM, 2333–2338.

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

[18] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints* (Jan. 2013). arXiv:cs.CL/1301.3781

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[21] Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*. 807–814.

[22] Shumpei Okura, Yukihiro Tagami, Shingo Ono, and Akira Tajima. 2017. Embedding-based News Recommendation for Millions of Users. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1933–1942.

[23] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*. 1310–1318.

[24] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

[25] Steffen Rendle. 2010. Factorization machines. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. IEEE, 995–1000.

[26] Joshua Saxe and Konstantin Berlin. 2017. eXpose: A Character-Level Convolutional Neural Network with Embeddings For Detecting Malicious URLs, File Paths and Registry Keys. *arXiv preprint arXiv:1702.08568* (2017).

[27] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 255–262.

[28] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and Composing Robust Features with Denoising Autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML '08)*. ACM, New York, NY, USA, 1096–1103. https://doi.org/10.1145/1390156.1390294

[29] Shoujin Wang, liang Liu, Longbin Cao, Defu Lian, and wei Liu. 2018. Attention-based Transactional Context Embedding for Next-Item Recommendation.. In *AAAI*.

[30] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. 2 (04 2014).

[31] G. Zhou, C. Song, X. Zhu, X. Ma, Y. Yan, X. Dai, H. Zhu, J. Jin, H. Li, and K. Gai. 2017. Deep Interest Network for Click-Through Rate Prediction. *ArXiv e-prints* (June 2017). arXiv:stat.ML/1706.06978

[32] Guorui Zhou, Chengru Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xingya Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. 2017. Deep interest network for click-through rate prediction. *arXiv preprint arXiv:1706.06978* (2017).