Deep Learning for Online Display Advertising User Clicks and Interests Prediction*

Zhabiz Gharibshah[1], Xingquan Zhu[1], Arthur Hainline[2], and Michael Conway[2]

[1] Dept. of Computer & Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431, USA
{zgharibshah2017, xzhu3}@fau.edu
[2] Bidtellect Inc., Delray Beach, FL 33483, USA
{arthur,mike}@bidtellect.com

Abstract. In this paper, we propose a deep learning based framework for user interest modeling and click prediction. Our goal is to accurately predict (1) the probability that a user clicks on an ad, and (2) the probability that a user clicks a specify type of campaign ad. To achieve the goal, we collect page information displayed to users as a temporal sequence, and use long-term-short-term memory (LSTM) network to learn latent features representing user interests. Experiments and comparisons on real-world data shows that, compared to existing static set based approaches, considering sequences and temporal variance of user requests results in an improvement in performance ad click prediction and campaign specific ad click prediction.

1 Introduction

Computational advertising is mainly concerned about using computational approaches to deliver/display advertisement (Ad) to audiences (*i.e.* users) who might be interested in the Ad, at the right time [1]. The direct goal is to draw users' attention, and once the Ads are served/displayed on the users' device, they might take actions on the Ads and become potential buying customers. Due to the sheer volumes of online users and the advertisements, and users have different background and interests, not to mention their changing habits and interests, finding users interests is often the key to determine whether a user is interested in an Ad.

In display advertising, because AdExchange often only passes very limited information about the user [2], such as user device type, user agent, page domain name and URL, etc., in order to predict user interests, the industry commonly relies on generative modeling. Historical data is used to build tree-structured models whose parameters are used to derive the CTR value of the new impression. Common generative models include CTR hierarchy trees [3] or hierarchical Bayesian frameworks [4]. One inherent advantage of the generative model is that the model provides transparent interpretability for business to understand which factor(s) contribute the most to the CTR values. However, due to the limitations of the models, such methods can normally estimate

*This research is sponsored by Bidtellect Inc. and by US National Science Foundation through Grant No. CNS-1828181.

only a handful of parameters (*e.g.* using a number of selected factors to split the tree hierarchy), and are unable to consider many rich information from users, publishers, and websites for accurate CTR estimation.

Different from generative models, the increasing popularity of machine learning, particularly deep learning, has driven a set of predictive modeling methods, which treat user clicks as binary events, and uses supervised learning to train a classifier to predict the likelihood of an impression being clicked by users [5], including some deep neural networks based CTR estimation methods [6]. Such methods normally work on tens of thousands of features, and are often more powerful than generative models.

In this paper, we propose to consider temporal user information to estimate user clicks and user interests. We generalize these two problems as a binary classification task (for user click prediction) and a multi-class classification task (for user interest prediction). More specifically, we collect users' page visits as a temporal sequence, and train deep LSTM (long-term short-term memory) networks to make predictions.

2 LSTM Network for User Clicks and Interests Modeling

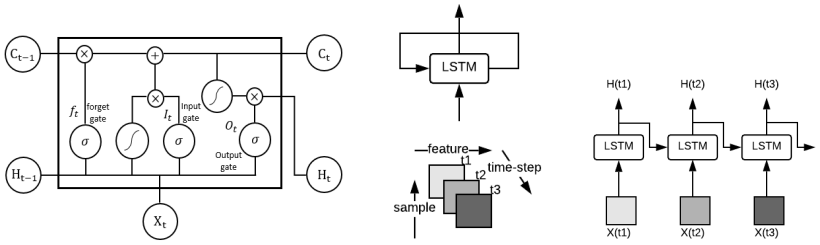
2.1 Problem Definition

let U be the a set of users $\{u_1, u_2, u_3, ..., u_n\}$ and R be a set of events. Each event denoted by $r_{u_j}^{t_i}$ represents the occurrence that an advertisement is displayed to a user u_j in a specific context at time t_i. In this case, the event is encoded as a real-valued vector ($r_{u_i}^{t_i} \in \mathbb{R}^d$). The context in display advertising industry is the page visited by the user which is in turn described by a hierarchy of page category IDs corresponding to various contextual information with different level of granularity [7, 8].

The set of all pre-defined page categories is denoted by $\mathbb{C}$ equals to $\{c_1, c_2, ..., c_{|\mathbb{C}|}\}$ where $|\mathbb{C}|$ is the number of categories. For a page visited by user u_j at time-step t_i, its page categories can be shown in the form of array like $[c_1, c_2, c_3, ...]$. For each user $u_j \in U$, we take the history of web pages visited by the user and denote it by $r_{u_j} = \{r_{u_j}^{t_1}, r_{u_j}^{t_2}, ..., r_{u_j}^{t_m}\}$. Because of the variety in the number of websites visited by users, we have $r_{u_j} \in \mathbb{R}^{m \times d}$ where m is the maximum sequence length. Thus, given the historical records of all users as $R = \{r_{u_1}, r_{u_2}, \cdots, r_{u_n}\}$ where $R \in \mathbb{R}^{n \times m \times d}$, $d < |\mathbb{C}|$, our **objective** is using historical user activities as the chronological sequence of requests before an arbitrary time-step t_i to predict (1) the probability that a user may interact with an Ad at t_i by generating a click response, and (2) predict which campaign Ad the user might click.

2.2 LSTM for User Modeling

Recurrent Neural Networks(RNN) is an extension of feed forward networks and has been successfully applied in various sequence data analysis and temporal sequence modeling [9]. While traditional RNN networks are unable to learn all term dependencies in sequences because of vanishing or exploding gradient problem [10], long short-term memory(LSTM) networks were introduced which uses special multiple-gate structured



(a) LSTM cell architecture (b) Single layer LSTM network (c) Unfolded single layer LSTM network

Fig. 1: LSTM cell and network architecture. (a) shows the detailed view of an LSTM cell, (b) shows a single layer LSTM network with respect to the input, represented in a 3rd-order tensor [sample, feature, time-step], and (c) shows the unfolded structure of the single layer LSTM network in (b). In our research, we stack three LSTM layers to form a deep LSTM network (detailed in the experimental settings).

cell to replace hidden layer nodes. Using LSTM cells in these networks has been shown as an efficient way to overcome these problems [10].

Two significant challenges in online display advertising to model user response and user interest using deep learning approaches like LSTM networks are that the collection of online user behavior data are (1) in multi-variant categorical form because each page may belong to one or multiple categories, and (2) user sequences of historical data may have different lengths because users' responses and actions vary over time. They result in multi-length sequences, where data points of each time-step may also include variant features. More specifically, in our model, the historical data collected for user modeling contains page category IDs of the pages that a user visited during a short period of time. For a user at a particular time-step, we have an array of category IDs of the page visited by the user discussed in Problem Definition section. Such IDs are represented as $[c_1, c_2, \cdots]$ which are in different lengths. Table 1 shows the sample of input sequential data used for user modeling.

One-Hot-Encoding with Thresholding To handle multi-length page categories as the features to describe each visited page, we use one-hot-encoding to represent them as sparse binary features. For each user, we have a sequence of visited pages attributed by a couple of page category IDs that correspond to their content. Therefore, each time-step can be shown as binary vector with length equals to the maximum number of categorical variables where 1 indicating the presence of each possible value from the original data. For example, in Table 1, at time-step t_1 the visited page of user u_2 is described by an array of page category IDs as features can be shown as $[0, 1, 1, 0, 0, 0, ..., 0]$. The dimension of vectors for time-step is determined by the number of unique page category IDs in the dataset that in our example it equals to $|\mathbb{C}| = 18$.

Concatenating these vectors generates a matrix with high dimensionality. Therefore, for features like page category IDs with high cardinality, using one-hot-encoding usually leads to extra computational costs. In the past, much research has been done to

work with such sparse binary features [11, 12]. To address this problem and in order to reduce the dimension of these vectors, we used an alternative to encode more frequent page category IDs based on a threshold based approach. In this case, page category IDs are sorted based on the number of their occurrences. Those with repetitions more than the user-defined threshold will be kept for the next parts.

Table 1: Schema of the data representation. We represent each audience(user) and his/her actions as multi-dimensional temporal sequence. Each row in the table denotes an audience, and t_1,t_2,$\cdots$,t_n denotes temporal order of the sequence (if $i>j$, then t_i happens after t_j). c_1,c_2,$\cdots$, c_m denotes the IAB tier-2 page category of the webpage visited by the users. $\langle$**click**$\rangle$ denotes an Ad click event from the audience. Not all sequences results in click events.

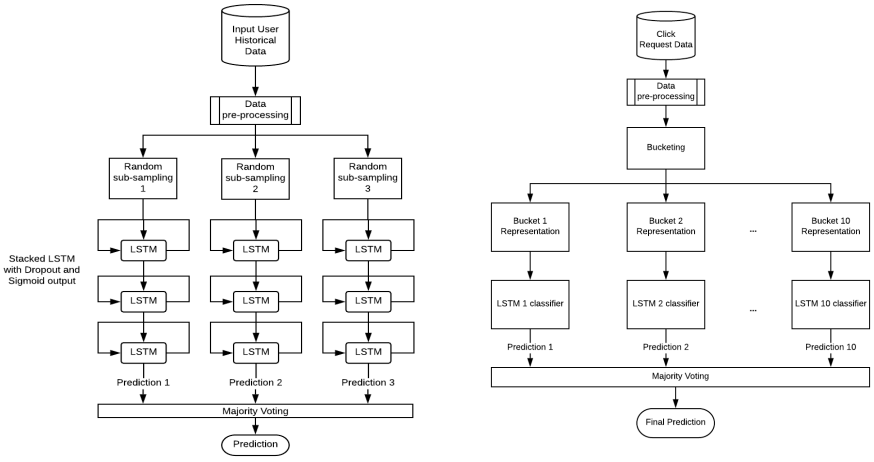
	Temporal order of audience response					
User	t_1	t_2	t_3	t_4	...	t_n
u_1	$\mathrm{r}_{u_1}^{t_1} = [c_1, c_2, c_3]$	$\mathrm{r}_{u_1}^{t_2} = [c_1, c_3]$	$\mathrm{r}_{u_1}^{t_3} = [c_1, c_4, c_5, c_6]$ $\langle$**click**$\rangle$	-	-	-
u_2	$\mathrm{r}_{u_2}^{t_1} = [c_2, c_3]$	$\mathrm{r}_{u_2}^{t_2} = [c_4]$ $\langle$**click**$\rangle$	-	-	-	-
u_3	$\mathrm{r}_{u_3}^{t_1} = [c_{10}, c_7, c_3, c_{20}]$	$\mathrm{r}_{u_3}^{t_2} = [c_1, c_3, c_{15}]$	$\mathrm{r}_{u_3}^{t_3} = [c_6, c_{12}, c_{22}, c_{24}, c_1, c_3]$	-	-	-
u_4	$\mathrm{r}_{u_4}^{t_1} = [c_8, c_{14}, c_{30}]$	$\mathrm{r}_{u_4}^{t_2} = [c_2, c_6]$	$\mathrm{r}_{u_4}^{t_3} = [c_{11}, c_{16}, c_{21}]$	$\mathrm{r}_{u_4}^{t_4} = [c_4, c_7, c_{11}]$ $\langle$**click**$\rangle$	-	-
...	.	.	.	.	.	.

Bucketing and Padding The variable length of sequences, like samples in Table 1, is another technical challenge. To handle sequences of any length and capture short and long dependencies in input data, padding with constant value (*e.g.* inserting zeros) is a straightforward strategy to make input dimensions fixed. However, applying this approach to train LSTM with wide range of sequence lengths is not only computationally expensive it also adds extra zero values resulting in bias in outcomes and changes input data distribution. Therefore, we propose to combine padding and bucketing to best utilize temporal information in sequences without inserting too many padding symbols.

To combine bucketing and padding, we construct several buckets in training samples, where sequences in each bucket have the same lengths corresponding to the range of sequence length in the dataset. Each sample is assigned to one bucket corresponding to its length. In this case, padding of samples mitigates to inside of buckets being used just for assigned sequences as much as necessary to fit into the bucket. The most important item in the sequence of request page categories is the last item that corresponds to the possible user click response, we use pre-padding approach. It means that each short sample inside buckets with the length lower than bucket size is pre-padded to become a sample with length equal to the maximum length in that bucket.

Following the idea, we designed an ensemble learning method for multi-class classification task. Rather than using the original splitting to generate buckets as the representative subset of samples in input space, we just use the sequence length of buckets to indicate one representation of samples through truncating time-steps. It means that for each representation, all samples in input data are trimmed to the selected sequence length by removing some time-steps from the beginning of sequences. Then in order to obtain classification, we build one LSTM model for each representation. The final

result of classification is generated by applying majority voting as the result merger of all models.



(a) User click prediction framework (b) User interest prediction framework

Fig. 2: The proposed User click prediction (a) and user interest prediction (b) frameworks. Given user request and response historical data, user click prediction aims to train stacked LSTM classifiers to predict whether a new user is going to click an Ad or not, *i.e.* a binary classification task; and user interest prediction will predict Ad campaign a new user is going to click, *i.e.* a multi-class classification task.

2.3 LSTM Based User Click Prediction Framework

Figure 2(a) briefly describes the structure of our proposed method for user click prediction problem as a binary classification. It includes the stacked LSTM model consisting of three LSTM layers followed by one fully connected layer with sigmoid activation to combine the output of hidden neurons in previous layers to predict click instances. In this case, the loss function is defined as the weighted binary cross entropy which aims to maximize the probability of correct prediction. The weight introduced in this function allows a trade-off between recall and precision in both classes to mitigate the negative effect of the class imbalance problem in our task:

$$L = 1/N \times \sum_{j=1}^{N} (y_i \times -log(p(x_i)) \times w + (1 - y_i) \times -log(1 - p(x_i))) \tag{1}$$

where N is the number of samples in training set. $y_i \in [0, 1]$ is target label and $p(x_i) \in [0, 1]$ is the network output, which represents the likelihood that how likely the sample x_i has a click response at the end. w is the coefficient which determine the cost of positive error relative to the misclassification error of negative ones.

2.4 LSTM Based User Interest Prediction Framework

Figure 2(b) outlines the model for for user interest prediction. It is defined as multi-class classification to classify the number of clicks in 10 different advertising campaigns. The number of buckets are defined uniformly over the range of sequence length in the dataset. Then, for each bucket, one representation of data is generated by trimming all longer samples and pre-padding shorter samples to the selected sequence length. Then prediction is made by following the ensemble learning approach. In this Figure, LSTM block follows the structure mentioned in Fig. 2(a) except the last layer having softmax activation function. In this case, the objective function is similar to equation (1) when w=1. It is actually an unweighted categorical cross entropy loss function in which $p(x_i)$ is the output of the network after softmax layer.

3 Experiments

3.1 Benchmark Data

We pulled out data from our industry partner's bidding engine, and prepared two datasets to validate user click and user interest prediction.

Post-View Click Dataset: This dataset is mainly used for validating binary user click prediction. We pulled 5.6 million users' request records from 1 day log events. These anonymous records include a chronological sequence of various request categories which represent user browsing interactions. In this case, there are two types of positive and negative responses from users where success occurs, if a post-view click takes place at end of a chain of visited impressions. Because of rarity of positive responses (click) in digital advertising, this dataset suffered from severe class imbalance problem. Thus, to deal with this issue, we use random down-sampling to get 10:90 positive:negative post sampling class distribution on the dataset.

Multi-Campaign Click Dataset: This dataset includes historical records with positive response in post-view click dataset. The positive response in this case is defined as user clicks on different type of campaigns (we used 10 types of campaigns). This dataset is mainly used for validating multi-class user interest prediction.

One issue we encountered in these datasets is that the sequence lengths are severely skewed where a large proportion of sequences are very short in length even less than 3 time-steps. Our bucketing and padding combined approach, introduced in Section 3.2, is specially designed to handle this challenge.

3.2 Experimental Settings and Performance Metrics

We implemented 7 methods for comparisons. All models were implemented through Tensorflow and CUDA to take advantage of using GPU and trained by Adam optimization as a variant of gradient descent. The remaining models are built using Scikit-learn library in Python. For data preprocessing, we convert input sequential data to binary vector by one-hot-encoding and get rid of less frequent categorical campaign IDs. we use a threshold to keep those categories with more 1,000 occurrence in our dataset. To control overfitting problem in neural networks early stopping mechanism is used to

stop after 10 subsequent epochs if there is no progress on the validation set. Dropout rate was set at 0.4 for neural networks. For the rest of methods L2 regularization is used in training process. All experiments are evaluated based on 5-fold cross validation.

We use the Area Under Receiver Operating Characteristics Curve (AUC) as the major evaluation metric because it shows the model accuracy of ranking positive cases versus negative ones. We also employ accuracy, F_1-measure, precision, and recall as additional performance metrics.

3.3 Performance Comparison

User Click Prediction Results As a binary classification task, the performance of proposed method is compared with SVM, Random Forest, Logistic Regression in addition a variant of convolutional neural network (CNN) [13]. Since input data has an extremely imbalanced class distribution with around 5,646,569 no-click user sequences (negative samples) versus 31,144 click user sequences (positive samples), we use random undersampling and ensemble learning to build the model in Fig. 2(b).

In our proposed method, three dimensional input data($\mathbb{R}^{n\times m\times d}$) are passed into the network where m and d are 70 and 153 corresponding to more frequent sequence lengths and the most frequent number of page category IDs. For remaining methods, the initial 3d input data is projected to the plane of $\mathbb{R}^{n\times d}$ by adding up values in sequence length dimension. Then, each model is trained by minimizing weighted binary cross entropy shown in Eq. (1). By default, we use cost ratio as 5 for positive samples because of the effectiveness seen in our experiments. Table 2 reports the result of prediction using different methods. As our proposed method pays more attention to history of requested pages before click, having higher performance in our proposed method shows the importance of this feature in click prediction.

Table 2: User click prediction results (binary classification task)

Method	Precision	Recall	F_1-measure	AUC	Accuracy
NaiveBayes	0.2638	0.2827	0.2729	0.6004	0.8583
Random Forest	0.3001	0.2758	0.2875	0.6045	0.8713
Logistic Regression	0.3353	0.2995	0.3164	0.6189	0.8782
Linear SVM	0.3534	0.2682	0.3050	0.6086	0.8850
SVM	**0.3815**	0.0333	0.0613	0.5138	0.9039
CNN	0.3492	0.4323	0.3862	0.6742	0.8707
LSTM	0.3111	**0.5196**	**0.3892**	**0.7001**	**0.8466**

User Interest Prediction Results For multi-class user interests prediction task, we compare proposed approach with NaiveBayes, Random Forest (with 100 tree estimators), Logistic Regression and two version of SVM with linear and RBF kernels. The input data are click samples used in previous task for click response prediction. Considering the AUC performance, the overall results in Table 3 illustrate that the proposed method in this classification task outperforms the others. It shows the effective of LSTM networks in detecting the correlation of sequential data and click response.

Table 3: User interest prediction results (multi-class classification task)

Method	Precision	Recall	F_1-measure	AUC	Accuracy
NaiveBayes	0.2486	0.2940	0.2713	0.6708	**0.4016**
Random Forest	0.3640	0.3401	0.3697	0.7404	0.3697
Logistic Regression	0.3911	0.2425	0.3240	0.7124	0.2233
Linear SVM	0.3858	0.2680	0.3369	0.7113	0.2415
SVM	0.1037	0.0654	0.0691	0.4593	0.0691
LSTM	0.3942	0.2628	0.3312	0.6994	0.2239
LSTM_Bucketing_Padding	**0.4076**	**0.3401**	**0.3835**	**0.7599**	0.3835

4 Conclusion

In this paper, we proposed a new framework for click response and user interest prediction using LSTM based deep learning. By combing padding and bucketing to learn binary user click prediction and multi-class user interest prediction, our method allows sequences to have variable lengths and variable number of dimensions and can maximally leverage temporal information in user sequences for learning. Experiments and comparisons on real-world data collected from industry partner show that our method is able to encode useful latent temporal information to predict user response and interests.

References

1. X. Zhu, H. Tao, Z. Wu, J. Cao, K. Kalish, and J. Kayne, *Fraud Prevention in Online Digital Advertising*. Springer International Publishing, 2017.
2. IAB, "Openrtb api specification version 2.5," *https://www.iab.com/guidelines/real-time-bidding-rtb-project/*, 2016.
3. H. Liu, X. Zhu, K. Kalish, and J. Kayne, "Ultr-ctr: Fast page grouping using url truncation for real-time click through rate estimation," *Proc. of IEEE IRI Conf.*, pp. 444–451, 2017.
4. L. Q. Ormandi Robert, Yang Hongxia, "Scalable multidimensional hierarchical bayesian modeling on spark," *Proc. of the 4th Intl. Conf. on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Sys., Programming Models and App.*, vol. 41, pp. 33–48, 2015.
5. W. D. Li Cheng, Lu Yue and P. Sandeep, "Click-through prediction for advertising in twitter timeline," *Proc. of ACM SIG KDD*, 2015.
6. H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for ctr prediction," *Proc. of the Intl. J. Conf. on Arti. Intell.*, pp. 1725–1731, 2017.
7. O. Chapelle, E. Manavoglu, and R. Rosales, "Simple and scalable response prediction for display advertising," *ACM Trans. on Intell. Sys. & Tech.*, vol. 5, no. 4, pp. 61:1–61:34, 2015.
8. D. Agarwal, R. Agrawal, R. Khanna, and N. Kota, "Estimating rates of rare events with multiple hierarchies through scalable log-linear models," *Proc. of the KDD Conf.*, 2010.
9. Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015.
10. L. Zhang, S. Wang, and B. Liu, "Deep learning for sentiment analysis : A survey," *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, 2018.
11. S. Rendle, "Factorization machines," *ICDM. IEEE*, p. 9951000, 2010.
12. Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," *IEEE Intl. Conf. on Data Mining*, pp. 1149–1154, 2016.
13. K. Yoon, "Convolutional neural networks for sentence classification," *arXiv:1408.5882*, 2014.