

# Neural Feature Embedding for User Response Prediction in Real-Time Bidding (RTB)

Enno Shioji<sup>1,2</sup> and Masayuki Arai<sup>2</sup>

<sup>1</sup>Adform A/S, Wildersgade 10B, DK-1408 Copenhagen, Denmark

<sup>2</sup>Graduate School of Science and Engineering, Teikyo University, Japan

<sup>1</sup>Enno.Shioji@adform.com, <sup>2</sup>arai@ics.teikyo-u.ac.jp

**Abstract.** In the area of ad-targeting, predicting user responses is essential for many applications such as Real-Time Bidding (RTB). Many of the features available in this domain are sparse categorical features. This presents a challenge especially when the user responses to be predicted are rare, because each feature will only have very few positive examples. Recently, neural embedding techniques such as word2vec which learn distributed representations of words using occurrence statistics in the corpus have been shown to be effective in many Natural Language Processing tasks. In this paper, we use real-world data set to show that a similar technique can be used to learn distributed representations of features from users web history, and that such representations can be used to improve the accuracy of commonly used models for predicting rare user responses.

## 1 Introduction

Predicting the probability of user response such as click, conversion etc. given an ad-impression is crucial for many advertisement applications, such as Real-Time Bidding (RTB). Because of its efficiency, linear models such as logistic regression are the most widely used for this purpose[15]. The models are commonly trained on sparse categorical features such as user agent, IDs of visited websites etc., which are encoded as sparse binary features via one-hot encoding[15]. One of the prominent problem with these models is the sparsity of data. Especially when feature interaction is used, the feature representation becomes extremely sparse, making it difficult to exploit the features efficiently. Moreover, traditionally the industry has focused on predicting clicks, but recently the focus has shifted to optimizing for other, much rarer user responses like conversions, which exacerbates this problem[4]. We refer to this problem as feature sparsity problem.

A similar issue has been recognized in Natural Language Processing (NLP)[3]. Many mainstream models rely on bag-of-words representation, which suffers from the same issue outlined above. Recently, neural embedding techniques known as word2vec, paragraph2vec etc. that map words and documents into low-dimensional vector space has been shown to yield state-of-the-art results in various NLP tasks[10, 8]. In this approach, occurrence statistics in the corpus

is used learn distributed word representations that are much more amenable to generalization.

In this paper, we use real-world data set to show that a similar technique can be applied to user response prediction in RTB. Similar to the situation in Natural Language Processing, a large amount of user web history can be used to learn high quality feature representations, which can then be used to predict (rare) user responses. The technique was shown to improve the accuracy of commonly used models, especially when labeled data was scarce.

## 2 Related Works

Various methods have been employed to address the feature sparsity problem. For example, higher order category information derived from human annotation, or from the data via unsupervised methods such as topic modelling, clustering etc.[17, 18] has been used to improve generalization. Other techniques such as counting features can also help by allowing rare features to contribute jointly[5].

Another category of solutions involve embedding sparse categorical features into low-dimensional vector space. Various feature transformation methods that yield dense features has been investigated in conjunction with deep neural networks resulting in improvement over major state-of-the-art models[17]. Zhang et al. also investigated a framework they refer to as implicit look-alike modelling, in which entities like users, web-pages, ads etc. are mapped into a latent vector space using both general web browsing behavior and ad response behaviour data[16].

In this paper, we report initial results of applying a feature transformation technique similar to neural word embedding to user response prediction in RTB. The technique has been successfully applied to other domains, such as product recommendation [11, 2]. The technique shares the benefits of its counterpart in NLP, such as the ability to encode feature sequences, the ability to incrementally update the embeddings with new data, and the availability of numerous improvements and extensions that have been developed since its advent. The result opens up exciting opportunities to apply techniques that have been successfully used with neural word embeddings, such as deep neural networks.

## 3 Neural Feature Embedding for User Response Prediction

We first provide a brief overview of the neural word embedding technique developed by Mikolov et al[9]. We consider one of its simplest form, the Continuous Bag-of-Word Model (CBOW) with a single context window. Given a word  $t$  in the corpus and a surrounding word  $c$ , we parametrise  $\theta$  such that the conditional probabilities  $p(t|c; \theta)$  is maximized for the corpus.  $p(t|c; \theta)$  can be modelled using soft-max as follows:

$$p(t|c; \theta) = \frac{e^{v_t \cdot v_c}}{\sum_{c' \in C} e^{v_t \cdot v_{c'}}} \quad (1)$$

where  $v_t$  and  $v_c \in R^n$  are vector representations for  $t$  and  $c$ , and  $C$  is the set of all available contexts.  $n$  is a hyper-parameter that determines the size of the embedding, and is chosen empirically. Note that we use a distinct representation for target and context, following the literature. This objective is straightforward but expensive to calculate. To alleviate this problem, a technique called negative sampling[9] is used, wherein random pairs of  $(t, c)$  is sampled from the corpus, assuming they are wrong. This yields the following objective:

$$\arg \max_{\theta} \sum_{(t,c) \in D} \log \frac{1}{1 + e^{-v_t \cdot v_c}} + \sum_{(t,c) \in D'} \log \left( \frac{1}{1 + e^{v_t \cdot v_c}} \right) \quad (2)$$

where  $D$  is the set of all target-context pair in the corpus and  $D'$  are randomly generated  $(t, c)$  pairs. The objective is now cheap to calculate.

In this paper, we consider a dataset consisting of ad impressions. When an ad is shown to a user, some of the browsing history of that user is available as sequence of content IDs. It is thus relatively straightforward to apply techniques such as CBOW[9], skip-gram[9] etc. to this data. For this experiment we chose to discard the sequence of the content IDs and only use the co-occurrence information. More specifically, we generated our positive  $(t, c)$  pairs by randomly sampling content IDs from the set of content IDs the user had consumed at the time of the impression, and our negative pairs randomly from the corpus. It is known that the probability distribution of such sampling influences the quality of the embeddings[10], but we used a uniform distribution for this initial experiment. We then used the resulting content embeddings as features in our user response model, for which we use logistic regression.

## 4 Experiment and Discussion

### 4.1 Dataset

We used a real-world RTB dataset provided by Adform. Each record in the data corresponds to an ad-impression, and is ordered chronologically. The record consists of a binary label that indicates whether the user subsequently clicked the ad (`click`), and a set of content IDs (`content_ids`) the user had consumed in the past 30 days, up to the time of the impression. The data was taken from Adform’s impression logs of July 2016. Records for which no `content_ids` were available were filtered out. Further, negative examples were down-sampled at a rate of 0.01 as the data is extremely imbalanced. After the down-sampling, there were 5.0M examples in total, with 1.1M positive examples. There were 891K distinct content IDs. A newer, larger version of the dataset with additional fields has been published [14]. The `content_ids` correspond to feature `c9` in this dataset.

## 4.2 Experiment Protocol

The experiment consisted of an unsupervised stage and a supervised stage.

**Unsupervised stage.** Content embeddings were learned from `content_ids` as described above. I.e. the `click` field was discarded and not used for this stage. Out of the 5.0M data instances, the oldest 4.0M were used for this stage. We trained the embeddings with varying embedding sizes  $n$  ( $2^{k \in [1..7]}$ ). Tensorflow[1] was used to implement this stage.

**Supervised stage.** In the supervised stage, binary classifiers that predicts `click` were trained, using different features (see below). For all experiments, Logistic Regression with L2 normalization was used. Out of the remaining 1.0M data instances, the newest 30% (300K) were held-out as validation dataset. The training was done with varying amounts of data (0.3K, 1K, 10K, 100K) that were randomly sampled from the remaining data (700K). To evaluate the performance of the models, area under the ROC curve (AUC) was used, which is a commonly used metric for evaluating user response prediction models in RTB[15]. Grid search was performed with varying regularization strength ( $10^{k \in [-2..1]}$ ) and embedding size, and the best result was used as measurement. scikit-learn[12] was used for the implementation. Below is the list of features we compared:

- SB: Sparse Binary. `content_ids` were encoded as Sparse Binary features via one-hot encoding. This is our baseline.
- DR: Distributed Representation. Each dimension of the resulting embeddings were scaled by its maximum absolute value. For each `content_id` in `content_ids`, the corresponding embedding was looked up and the mean of the embeddings were used as the feature vector. The resulting feature vector had thus the same length  $n$  as the embeddings.
- SB+DR: Sparse Binary and Distributed Representation. The feature vector of SB and DR were concatenated.

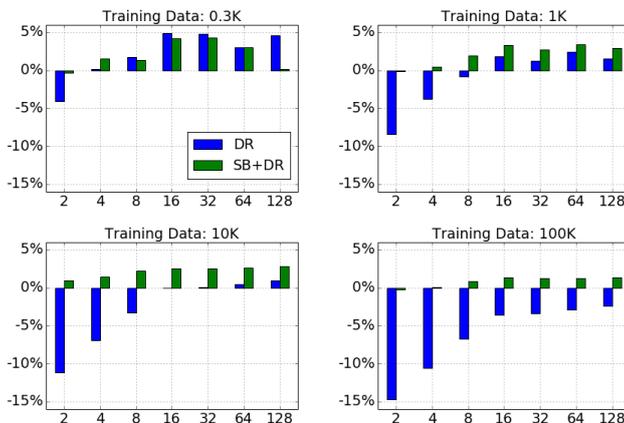
## 4.3 Performance Comparison and Discussion

Table 1 shows the best results obtained for each condition using the aforementioned grid-search. The results of SB+DR and DR is compared against SB (our baseline). DR outperforms SB when training data is scarce. SB+DR outperforms SB in all conditions, especially stronger when training data is scarcer. This is likely because when training data is scarce, the sparsity issue is more acute and thus the ability to generalize across features has a larger effect. However, when large amount of data is available, the lower-dimensional feature representation of DR likely limits the degree of differentiation between individual content IDs. When SB and DR is concatenated, both advantages can be preserved.

Figure 1 shows the difference in AUC from the SB baseline for DR and SB+DR, for varying embedding sizes ( $n$ ). Increasing  $n$  improves AUC, but the return diminishes after about 16 dimensions.

**Table 1.** AUC performance of click prediction

Training Data	SB+DR		DR		SB
	AUC	AUC Lift	AUC	AUC Lift	AUC
0.3K	60.93%	4.33%	<b>61.53%</b>	4.90%	56.60%
1K	<b>65.15%</b>	3.39%	64.20%	2.44%	61.76%
10K	<b>67.42%</b>	2.87%	65.47%	0.92%	64.55%
100K	<b>69.65%</b>	1.34%	65.94%	-2.37%	68.31%

**Fig. 1.** AUC Performance difference from baseline against embedding size  $n$ 

## 5 Conclusion

In this paper, we reported initial results of applying neural feature embedding technique to user response prediction in RTB, using a real-world dataset. To our best knowledge, this is the first time this technique was applied to this problem. We have demonstrated that the technique can improve performance of commonly used model in the industry, especially when labeled data is scarce and when thus the feature sparsity problem is most acute. The fact that large amount of data can readily be used for training of the feature embeddings, and that the commonly used logistic regression can be used at prediction time make the result ideal for industrial implementation.

The result also opens up exciting opportunities to apply improvements and techniques that have been developed around neural word embeddings, such as incorporating global context, using multiple representations per word[6], optimizing the embeddings for a specific supervised task using target labels[7], using a global log-bilinear regression instead of the earlier local context window methods[13], applying deep neural networks on the embeddings etc.

## References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P.A., Vasudevan, V., Warden, P., Wicke, M., Yu, Y., Zhang, X.: Tensorflow: A system for large-scale machine learning. CoRR abs/1605.08695 (2016)
2. Barkan, O., Koenigstein, N.: Item2vec: Neural item embedding for collaborative filtering. CoRR abs/1603.04259 (2016)
3. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. *J. Mach. Learn. Res.* 3, 1137–1155 (Mar 2003)
4. Dalessandro, B., Hook, R., Perlich, C., Provost, F.: Evaluating and Optimizing Online Advertising: Forget the Click, But There are Good Proxies. *Social Science Research Network Working Paper Series* (Oct 2012)
5. He, X., Pan, J., Jin, O., Xu, T., Liu, B., Xu, T., Shi, Y., Atallah, A., Herbrich, R., Bowers, S., Candela, J.Q.n.: Practical lessons from predicting clicks on ads at facebook. In: *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. pp. 5:1–5:9. ADKDD’14, ACM, New York, NY, USA (2014)
6. Huang, E.H., Socher, R., Manning, C.D., Ng, A.Y.: Improving word representations via global context and multiple word prototypes. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*. pp. 873–882. Association for Computational Linguistics (2012)
7. Labutov, I., Lipson, H.: Re-embedding words. (2013)
8. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. CoRR abs/1405.4053 (2014)
9. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
10. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. CoRR abs/1310.4546 (2013)
11. Nedelec, T., Smirnova, E., Vasile, F.: Content2vec: Specializing joint representations of product images and text for the task of product recommendation. *Unpublished Manuscript* (2017)
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 2825–2830 (2011)
13. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: *Empirical Methods in Natural Language Processing (EMNLP)*. pp. 1532–1543 (2014)
14. Shioji, E.: Adform click prediction dataset. *Harvard Dataverse* doi:10.7910/DVN/TADBY7 (2017)
15. Wang, J., Zhang, W., Yuan, S.: Display advertising with real-time bidding (RTB) and behavioural targeting. CoRR abs/1610.03013 (2016)
16. Zhang, W., Chen, L., Wang, J.: Implicit look-alike modelling in display ads: Transfer collaborative filtering to CTR estimation. CoRR abs/1601.02377 (2016)
17. Zhang, W., Du, T., Wang, J.: Deep learning over multi-field categorical data: A case study on user response prediction. CoRR abs/1601.02376 (2016)
18. Zhang, W., Yuan, S., Wang, J.: Real-time bidding benchmarking with ipinyou dataset. CoRR abs/1407.7073 (2014)