Predictive Analysis by Leveraging Temporal User Behavior and User Embeddings

Charles Chen[1], Sungchul Kim[2], Hung Bui[3], Ryan Rossi[2], Eunyee Koh[2], Branislav Kveton[2] and Razvan Bunescu[1*]
[1]Ohio University, [2]Adobe Research, [3]DeepMind
[1]{lc971015, bunescu}@ohio.edu
[2]{sukim, rrossi, eunyee, kveton}@adobe.com
[3]bui.h.hung@gmail.com

ABSTRACT

The rapid growth of mobile devices has resulted in the generation of a large number of user behavior logs that contain latent intentions and user interests. However, exploiting such data in real-world applications is still difficult for service providers due to the complexities of user behavior over a sheer number of possible actions that can vary according to time. In this work, a time-aware RNN model, TRNN, is proposed for predictive analysis from user behavior data. First, our approach predicts next user action more accurately than the baselines including the n-gram models as well as two recently introduced time-aware RNN approaches. Second, we use TRNN to learn user embeddings from sequences of user actions and show that overall the TRNN embeddings outperform conventional RNN embeddings. Similar to how word embeddings benefit a wide range of task in natural language processing, the learned user embeddings are general and could be used in a variety of tasks in the digital marketing area. This claim is supported empirically by evaluating their utility in user conversion prediction, and preferred application prediction. According to the evaluation results, TRNN embeddings perform better than the baselines including Bag of Words (BoW), TF·IDF and Doc2Vec. We believe that TRNN embeddings provide an effective representation for solving practical tasks such as recommendation, user segmentation and predictive analysis of business metrics.

CCS CONCEPTS

• ;

KEYWORDS

Recurrent Neural Networks; user behavior modeling; representation learning

ACM Reference Format:
Charles Chen[1], Sungchul Kim[2], Hung Bui[3], Ryan Rossi[2], Eunyee Koh[2], Branislav Kveton[2] and Razvan Bunescu[1]. . Predictive Analysis by Leveraging Temporal User Behavior and User Embeddings. In *Proceedings of (CIKM'18).* ACM, New York, NY, USA, 8 pages.

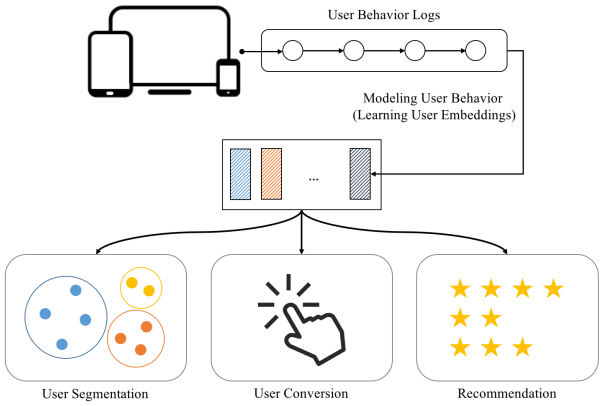


Figure 1: Overview of our approach; The user embeddings generated by our approach can be used in predictive analysis such as user segmentation, click/conversion prediction, user churn, and users' preferred items prediction.

1 INTRODUCTION

With the rapid growth of mobile devices, a large number of behavior logs are generated while people use their devices to access webpages or mobile services. This leads to the task of understanding and leveraging user behavior patterns. Accordingly, there have been prior attempts to leverage user behavior logs and extracting latent interests for solving real-world problems such as personalization [19, 31, 35] and conversion prediction [34]. More recently, user behavior logs have been used for inducing user representations that reflect users' hidden intention or temporal patterns [7, 30]. The resulting user representations can benefit look-alike modeling strategies for new customer acquisition [20]. In this work, we introduce time-aware RNNs (TRNN) for modeling user behavior and computing user embeddings from user behavior logs (Figure 1). Behavior logs generally contain attributes such as timestamps, user tags, and session information that have been underutilized in previous work. The proposed TRNN leverages temporal information extracted from event timestamps, which is shown experimentally to lead to substantial improvements in prediction accuracy. Training the TRNN model on user behavior logs has two important outcomes:

*Most of this work was carried out while the first author was an intern at Adobe Research, and Hung Bui was at Adobe Research.

CIKM'18, ,

- A model for **predicting next user actions**, which can help marketers provide a more seamless user experience.
- **User embeddings**, computed based on the sequence of RNN states over the user behavior log.

First, in the next action prediction task, experimental results show that our model outperforms bi-gram and tri-gram models, conventional RNNs, as well as two recently introduced time-aware RNN models that exploit a variation of LSTM cells to handle periodicity of event sequences: DeepCare[23] and TLSTM[2]. A byproduct of training TRNN to predict next user actions is the corresponding sequence of states. When aggregated into fixed-size embeddings, they can be used as effective representations of users in many other tasks in digital marketing. We demonstrate their general utility as user embeddings by evaluating them in two additional prediction tasks:

- **Conversion prediction**: We are interested in whether the users will convert (for example, make a purchase) in the future. Conversion prediction models enable marketers to discover and analyze specific user behavior–conversion patterns.
- **Preferred product prediction**: It provides insight to the marketers regarding product popularity and allows the users to be provided with personalized product recommendations.

In these two practical tasks, experimental results show that TRNN embeddings perform better than the Bag of Words (BoW), BoW + TF·IDF and Doc2Vec baselines. Similar to how word embeddings benefit a wide range of task in natural language processing (NLP), the learned user embeddings are general and could be used in a variety of tasks in the digital marketing area.

The rest of this paper is organized as follows. Section 5 provides a survey of related literature on the topic of user behavior modelling and representation learning. Section 2 describes the datasets we use to evaluate our approach. In Section 3, we introduce our model TRNN for user behavior modeling and representation learning. Section 4 describes the experimental evaluation of our method. We conclude the paper with a summary of our contributions and thoughts on future work.

2 USER BEHAVIOR DATASETS

We create two datasets of user behavior by extracting user action logs from corporate websites, while excluding bots according to predefined bot rules based on user-agent-string, IP address with wildcard match and IP range match. To handle the cross-device environment, we define user actions by concatenating 'device' and 'page name' describing specific paths to mobile/web pages that users visited. There are three different devices: desktop, smartphone, and tablet. Table 1 shows an example of user event sequence after anonymizing the company's identity.

Each user is represented by a sequence of actions, where each action is associated with a timestamp. In preprocessing, we remove duplicate user actions that were repeated within 5 seconds and filter out infrequent actions that occurred less than 100 times during a month. The number of remaining users and unique user actions are shown in Table 2 for each of the two datasets: CompanyA and CompanyB. The CompanyA dataset contains more unique user actions than CompanyB dataset.

1451923253 | tablet:XYZ 3.2.1 (50)
1451923258 | mobile:App/201512161
1451923261 | mobile:App/us/en/myplans/landing/phone
1451923267 | tablet:App/us/en/home
1451923275 | mobile:App/us/en/photopass
1453310277 | tablet:App/us/en/explore/thingstodo/home
1453310605 | mobile:App/us/en/tools/modifyselectpeople

Table 1: An example of the user behavior log where each user action is associated with a timestamp.

Datasets	# of users	# of actions
Company A	106,616	10,000
Company B	54.155	1,764

Table 2: Number of users and number of unique actions in our datasets; user behavior logs extracted from corporate websites or mobile services.

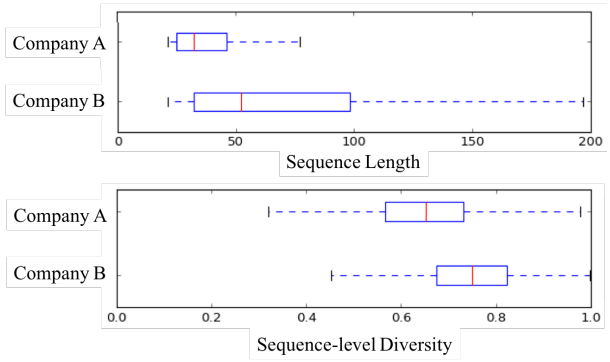


Figure 2: Boxplot of 1) sequence length and 2) sequence-level diversity for two datasets. The diversity of actions in a sequence is defined as the ratio of the number of unique user actions over the length of sequence.

For a better understanding of the datasets, we perform a statistical analysis in terms of the length of user action sequences and the diversity of actions in the sequences. For each sequence, the diversity of actions is computed as the number of unique user actions over the length of sequence. According to the analysis results, CompanyA dataset generally contains shorter sequences than CompanyB dataset, and has lower sequence diversity. This can make user modeling relatively easier. A more detailed analysis with quantitative results will be given in the experiments section. In this paper, "action sequence" and "event sequence" are used interchangeably.

3 USER BEHAVIOR MODELING VIA TRNNS

A RNN processes a sequence of vectors $\mathbf{S} =< \mathbf{x}_1, .., \mathbf{x}_T >$, where $\mathbf{x}_t$ is the input at time step t, such that the hidden state $\mathbf{h}_t \in \mathcal{R}^n$ at time step t is computed recursively from the previous hidden state and the current input, as follows:

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}) \quad (1)$$

where f is a non-linear function (e.g. a logistic sigmoid). To alleviate the vanishing gradient problem, our framework uses LSTM networks wherein the simple non-linearity f is replaced with more complex LSTM units [15].

Figure 3 shows an overview of our time-aware RNN model. First, as described in Section 2, we represent each user's behavior log as an event sequence. Different from other types of sequential data such as text, user actions are time-stamped, which enabled us to additionally consider the time difference between consecutive actions.

We observed that the time differences between user events belong to a wide range of values and can influence the likelihood of particular user actions. As an example, consider that a user purchase consists of three steps as follows: (Step 1) Visit an item, (Step 2) Add an item to wishlist, and (Step 3) Place an order. Assuming the user has gone through steps 1 and 2, the likelihood of actually taking step 3 depends on the time difference between the last two steps. If this time difference is small, it is likely that the user will actually purchase the item. However, if step 2 is taken after more than one hour, it is unlikely the user will go through step 3.

We build an action vocabulary $\mathcal{A}$ consisting of all the unique user actions in the dataset. We use $|\mathcal{A}|$ to denote the size of the action vocabulary. In the following, we use 'action set' and 'action vocabulary' interchangeably. We then compute the time difference between consecutive actions as well as the time elapsed between an action and the beginning of the session. There are two types of inputs at each step: the one-hot representation for the user action and the two timestamp-based inputs: 1) the time difference $\Delta t_i^{(1)}$ between the current action and the last action, and 2) the time $\Delta t_i^{(2)}$ elapsed between the beginning of the session and the current action. $\Delta t_i^{(1)}$ and $\Delta t_i^{(2)}$ are defined by:

$$\Delta t_i^{(1)} = \begin{cases} 0 & \text{if } i = 0 \\ \min((t_i - t_{i-1})/t_{sess}, 1.0) & \text{otherwise.} \end{cases} \quad (2)$$

$$\Delta t_i^{(2)} = \begin{cases} 0 & \text{if } i = 0 \\ (t_i - t_0)/(t_{max} - t_0) & \text{otherwise.} \end{cases} \quad (3)$$

where t_{sess} is the threshold for sessionizing behavior logs (6 hours in this work) and t_{max} is the timestamp for the last user action in the sequence. Then the LSTM input gate $\mathbf{z}_i$, forget gate $\mathbf{f}_i$ and output gate $\mathbf{o}_i$ at step i are computed as follows:

$$\mathbf{x}_i = [(\mathbf{E}\mathbf{a}_i)^T; \Delta t_i^{(1)}; \Delta t_i^{(2)}]^T \quad (4)$$

$$\mathbf{z}_i = \sigma(\mathbf{W}\mathbf{x}_i + \mathbf{U}\mathbf{h}_{i-1} + \mathbf{b}) \quad (5)$$

$$\mathbf{f}_i = \sigma(\mathbf{W}_f\mathbf{x}_i + \mathbf{U}_f\mathbf{h}_{i-1} + \mathbf{b}_f) \quad (6)$$

$$\mathbf{o}_i = \sigma(\mathbf{W}_o\mathbf{x}_i + \mathbf{U}_o\mathbf{h}_{i-1} + \mathbf{b}_o) \quad (7)$$

where $\mathbf{a}_i \in \mathcal{R}^{|A|}$ is the one-hot representation for the i^{th} user action in the sequence, $\mathbf{E} \in \mathcal{R}^{m \times |A|}$ is a set of action embeddings for all the user actions; m is the dimension of the action embeddings; $\mathbf{x}_i \in \mathcal{R}^{m+2}$; $\mathbf{W}, \mathbf{W}_f, \mathbf{W}_o \in \mathcal{R}^{n \times (m+2)}$, $\mathbf{U}, \mathbf{U}_f, \mathbf{U}_o \in \mathcal{R}^{n \times n}$ are trainable matrices; n is the dimension of the hidden state; $\mathbf{b}, \mathbf{b}_f, \mathbf{b}_o \in \mathcal{R}^n$ are bias terms and $\sigma(\cdot)$ is the logistic sigmoid.

The memory cell $\mathbf{c}_i$ and the hidden state $\mathbf{h}_i$ at current time step i are updated as follows:

$$\mathbf{g}_i = \phi(\mathbf{W}_c\mathbf{x}_i + \mathbf{U}_c\mathbf{h}_{i-1} + \mathbf{b}_c) \quad (8)$$

$$\mathbf{c}_i = \mathbf{f}_i \odot \mathbf{c}_{i-1} + \mathbf{z}_i \odot \mathbf{g}_i \quad (9)$$

$$\mathbf{h}_i = \mathbf{o}_i \odot \phi(\mathbf{c}_i) \quad (10)$$

where $\mathbf{W}_c \in \mathcal{R}^{n \times (m+2)}$ and $\mathbf{U}_c \in \mathcal{R}^{n \times n}$ are weight matrices, $\mathbf{b}_c \in \mathcal{R}^n$ is the bias term, $\phi(\cdot)$ is the hyperbolic tangent and $\odot$ indicates the element-wise multiplication.

Figure 3 shows the overall architecture of our model. The action embeddings can be pre-trained using an external model such as word2vec, or can be trained simultaneously with the other parameters in our model.

3.1 User Behavior Modeling

In order to predict the next user action based on the output of LSTM, the model computes a probability distribution over all the user actions in the action vocabulary $\mathcal{A}$ at time step i:

$$\mathbf{p}_i = [p(y_i = a|\mathbf{x}_i)]_{1 \le a \le |A|} = softmax(\mathbf{W}_s\mathbf{h}_i) \quad (11)$$

where y_i is the predicted next user action, $\mathbf{W}_s \in \mathcal{R}^{|A| \times n}$ is the weight matrix and $\mathbf{p}_i \in \mathcal{R}^{|A|}$ indicates the probability distribution for the predicted next user action. The loss E_i at time step i is given by the following equation:

$$E_i = -\ln \mathbf{p}_i[y_i] = -\ln p(y_i|\mathbf{x}_i) \quad (12)$$

The model is optimized by minimizing the total loss averaged over all the actions appearing in the training dataset:

$$E = \frac{1}{|D|} \sum_{\mathbf{S} \in \mathcal{D}} \frac{1}{T} \sum_{1 \le i \le T} E_i \quad (13)$$

where $\mathbf{S} =< \mathbf{x}_1, .., \mathbf{x}_T >$ represents a user action sequence, T is the length of $\mathbf{S}$ and $|D|$ is the number of training examples.

Once the model is trained, the next user action $\hat{y}_i \in \mathcal{A}$ is predicted as follows:

$$\hat{y}_i = \arg \max_{1 \le a \le |A|} p(y_i = a|\mathbf{x}_i) \quad (14)$$

Finally, the user embedding $\mathbf{u} \in \mathcal{R}^n$ corresponding to the user event sequence $\mathbf{S}$ is calculated as max pooling of all RNN states.

3.2 Sequence-level Dropout

Dropout is a regularization technique that reduces complex co-adaptation on training data, in order to improve the generalization performance of (deep) neural network models [26]. Our user behavior data contains noise and a large number of duplicate actions. Even after doing preprocessing as described in Section 2, there exist many duplicated user actions in individual sequences. On webpages or mobile applications, it is common that users click the same banner or button frequently. In addition to that, user behavior does not strictly adhere to a particular order. People frequently go back and forth between pages, and even directly move to the page by typing the path. These factors can have a negative impact on modeling user behavior. To alleviate this problem, we apply dropout to input sequences as described in Figure 4 where the circles shown with dashed contours are randomly selected to be dropped before the sequence is fed into the model for training.

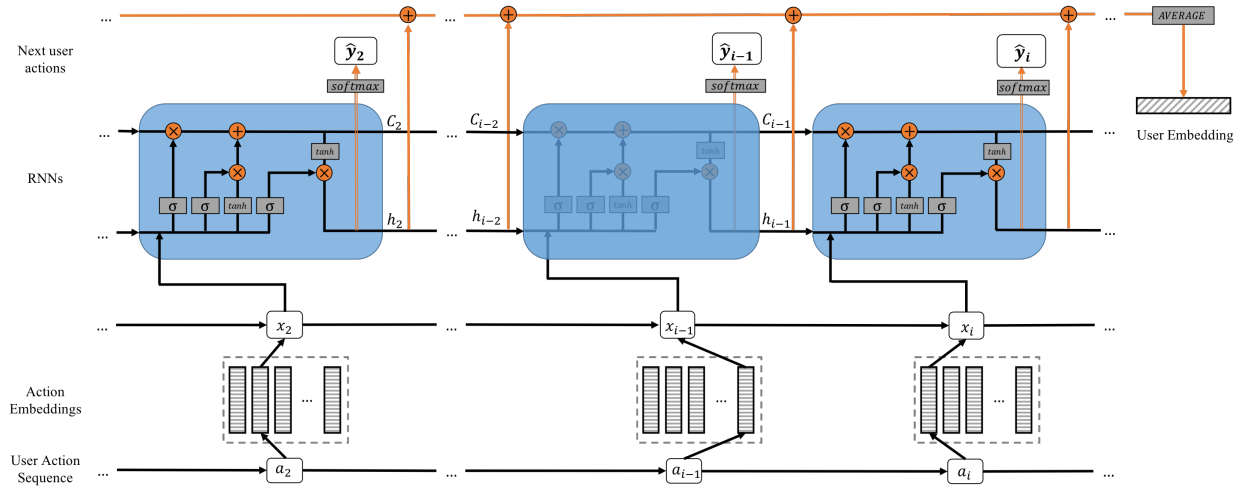


Figure 3: Our framework to leverage RNN-based user embeddings for predictive analysis.

Methods	Datasets	0%	20%	40%	60%	80%
RNN	CompanyA	0.3250 (0.00%)	**0.3314 (1.95%)**	0.3149 (-3.11%)	0.3286 (1.11%)	0.3110 (-4.31%)
	CompanyB	0.2787 (0.00%)	0.2875 (3.14%)	**0.2948 (5.78%)**	0.2943 (5.59%)	0.2870 (2.97%)
TRNN	CompanyA	0.4613 (0.00%)	0.4619 (0.13%)	**0.4623 (0.20%)**	0.4584 (-0.64%)	0.4537 (-1.65%)
	CompanyB	0.3573 (0.00%)	**0.3574 (0.02%)**	0.3565 (-0.22%)	0.3558 (-0.43%)	0.3524 (-1.37%)

Table 3: Impact of dropout on next action prediction. Validation accuracy on our datasets. The numbers in the parentheses show the performance improvement from using 0% of dropout in each user action sequence.

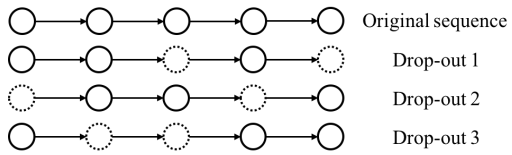


Figure 4: Sequence-level dropout. The circles shown with dashed contours are randomly selected to be dropped before the sequence is fed into the model for training.

Table 3 shows the top-1 accuracy with varying dropout ratio on our datasets, in the task of predicting next user action. The results show that in most cases the sentence-level dropout is more beneficial for conventional RNNs. Specifically, the performance improvement by dropout is 1.95%, and 5.78% for CompanyA and CompanyB, respectively. In contrast, the TRNN does not benefit as much from the sentence-level dropout, which results in less than 1% performance improvement. It can be attributed to that most of generalization performance achieved by sentence-level dropout are already learned by treating temporal factors. As an example, most duplicated actions are likely occurred after few seconds. In contrast to that, main pages are likely visited after quite amount of time - new session after previous session. Nevertheless, TRNNs perform better than RNNs in most cases, regardless of sentence-level dropout.

Method	Cov	Acc@1	Acc@5	MRR
Bigram	0.2825	0.4287	0.6086	0.5221
Trigram	0.6270	0.4285	0.5946	0.4699
DeepCare	0.3959	0.4349	0.6459	0.5320
TLSTM	0.4270	0.4317	0.6433	0.5293
RNN	0.8434	0.3314	0.6119	0.4401
TRNN	0.8214	**0.4623***	**0.6953***	**0.5690***

(A) CompanyA

Method	Cov.	Acc@1	Acc@5	MRR
Bigram	0.5909	0.3209	0.583	0.4589
Trigram	0.9041	0.3418	0.5922	0.4699
DeepCare	0.3588	0.3219	0.6245	0.4704
TLSTM	0.3446	0.3163	0.6253	0.4702
RNN	0.7352	0.2977	0.6446	0.4636
TRNN	0.6631	**0.3572***	**0.6833***	**0.5012***

(B) CompanyB

Table 4: Accuracy of top-k actions predicted by models

4 EXPERIMENTAL EVALUATION

4.1 Prediction of Next User Action

Motivation: From the perspective of professional marketers, accurate predictions of users' next behavior would allow them to provide a more seamless user experience. In order to comparatively

evaluate the performance of our model in this setting, we also experimented with a number of different baselines, and computed for each the average action-level top-k accuracy.

Experimental setting: We randomly split the datasets described in the Table 2 as follows: 80% of the user event sequences are used for training, while the remaining 20% are used for testing. A prediction is true if the actual action is included in the top-k predictions and false otherwise. The accuracy is defined as the number of correct prediction over the total number of actions within the test set. We compute the average accuracy of top-k predictions at the action level as well as the Mean Reciprocal Rank (MRR) which is a conventional way to measure the quality of ranking[1]. Additionally, we compute coverage (Cov.) as the ratio of number of unique user actions that are predicted over the length of user actions sequence. Note that we cannot argue that higher or lower coverage is always better. However, when related with accuracy it can provide a better understanding of the behavior of each prediction model. We compute the average accuracy and predicted action coverage for the our datasets using the proposed TRNNs as well as the baselines.

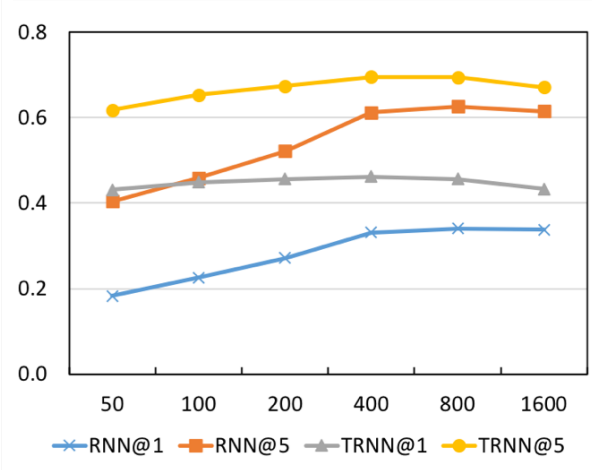
Baselines: In our experiments, we use n-gram probability models as one baseline. N-gram models estimate the probability of the n^{th} user action based on the co-occurrence with the previous $(n-1)$ actions. The second baseline is the DeepCare model introduced by Pham et al. [23]. The third baseline we compare with is TLSTM [2]. Both of these two models exploit the timestamps associated with sequential data, and are thus appropriate to use with our datasets. We also use conventional RNNs as one baseline.

Results: According to the results shown in Table 4, the TRNN model outperforms the bi-gram and tri-gram models, and the conventional RNNs by a large difference in terms of accuracy and Mean Reciprocal Rank (MRR), although the actual improvements vary depending on the complexity of dataset and the size of the action set. The results indicate that in the context of predicting next user actions, leveraging both sequential and temporal information is crucial.

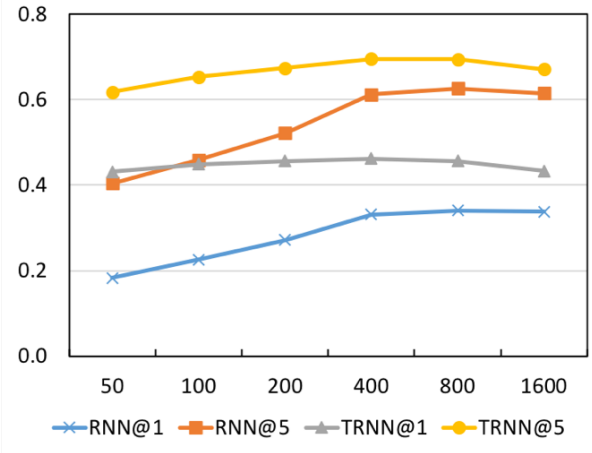
TRNN also achieves better performance compared to the recently proposed time-aware models DeepCare and TLSTM. In contrast with their claim that avoiding alteration of the current input's effect by using temporal information is helpful for modeling sequential data, the results show that allowing complex alteration of the input with temporal information is important for modeling user behavior patterns. This could also be an evidence of the differences between user behavior data and clinical records. User behavior logs from websites or mobile services usually contain much more actions or events. In terms of temporal information, user behavior data contains more noise and smaller time differences between consecutive actions, and accordingly, user actions show more complex patterns which cannot be captured by the notion of general periodicity.

In terms of the coverage, bi-gram model covers the smallest amount of the user actions compared to other models on CompanyA dataset. However, using the entire set of user actions does not guarantee good performance. For example, on CompanyB dataset, tri-gram shows worse performance than TRNN, but the predicted actions cover up to 90% of the entire user action set.

[1]We used McNemar's test to statistically assesses the accuracy of compared models in the first task and t-test for the other metrics including MRR. '*' indicate that the result is statistically significant from the next nearest run at confidence level of 98% ($\alpha = 0.02$)



(A) Company A



(B) Company B

Figure 5: Performance of next user action prediction with different size of embeddings (x-axis is embedding size and y-axis is the top-1 accuracy)

We conducted experiments to explore the impact of the dimension of action embeddings on the model performance (Figure 5). Generally, the larger the action embedding dimension, the more accurate the results are, but the ratio of performance improvement decreases as the dimension increases. Conventional RNNs are influenced by the embedding size more than TRNNs. However, although RNNs show larger gains from larger-size embeddings, TRNNs perform better than RNNs in most cases even with a smaller dimension for their embeddings. Conventional RNNs obtain performance comparable to TRNNs only when the size of vocabulary is limited.

4.2 Prediction of User Conversion

Motivation: Except for previous specific user events, marketers could also be interested in predicting long-term user behavior which

Method	Dim.	RIG	AUC
BoW	10000	-0.4909	0.7367
BoW + TF·IDF	10000	-0.4908	0.7072
Doc2Vec	400	-0.7564	0.6356
RNN	400	-0.5950	0.7434
DeepCare	400	-0.5313	0.7681
TLSTM	400	-0.4990	0.7817
TRNN	400	**-0.4802***	**0.7872***

CompanyA

Table 5: Performance of models on user conversion prediction.

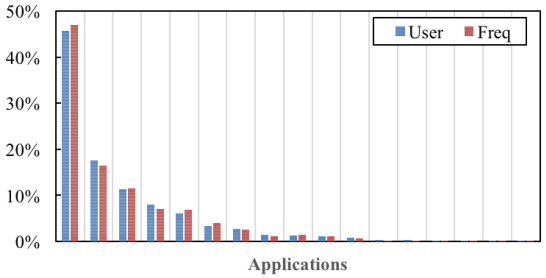


Figure 6: Ratio of active users and accumulated frequency per application

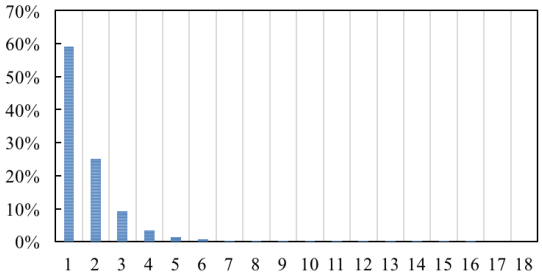


Figure 7: Ratio of users per the preferred application count

cannot be directly defined based on specific user actions, such as purchasing items, or (un-)subscribing services. To evaluate our approaches in this setting, we provide the user embeddings computed by TRNNs as inputs to a linear classifier trained on the task of user conversion. Although conversion could have different definitions according to the target domain, we focus on user purchase, which is available in our dataset CompanyA.

Experimental setting: In order to train the classifier, we specify the label as 1 if the user will make a purchase after t_T, the last time point in the event sequence for this user, 0 otherwise. Conventionally, the distribution of user conversion labels is highly skewed towards negative labels. Therefore, we weight the loss with $|loss_{pos}|/|\mathcal{P}|$ for positive labels and $|loss_{neg}|/|\mathcal{N}|$ for negative labels where $\mathcal{P}$ and $\mathcal{N}$ represent the user set with positive and negative labels, respectively. To measure the performance of the classification model, we conduct 5-fold cross-validation with Area Under ROC Curve (AUC) and Relative Information Gain (RIG) as the evaluation metrics, which is the conventional practice in previous work on click prediction [9, 29, 34].

Classification Model: Given a user embedding $\mathbf{u} \in \mathcal{R}^n$, we train a logistic regression model by minimizing the following cross-entropy loss:

$$E = -\sum_{\mathbf{u} \in \mathcal{P} \cup \mathcal{N}} y \log p(\hat{y} = 1|\mathbf{u}) + (1 - y) \log p(\hat{y} = 0|\mathbf{u}) \qquad (15)$$

where $p(\hat{y}|\mathbf{u}) = 1/(1 + \exp(-\mathbf{w}^{\mathbf{T}}\mathbf{u}))$. w is the vector of parameters of the logistic regression model. Note that we can use any other linear or non-linear model as well.

Baselines: As a baseline, to represent each user, we first use Bag-of-Words (BoW) and combine it with TF·IDF as the second baseline BoW + TF·IDF. In these two cases, the dimension of user representation is equal to the size of entire action vocabulary $|\mathcal{A}|$. Additionally, we compare against the representation computed from conventional RNNs and Doc2Vec[2] [18]. Note that Doc2Vec learns embeddings of sequences by predicting a target action based on its neighbors. We also use DeepCare[23] and TLSTM[2] as baselines. We obtain the user embeddings from these two methods by taking max-pooling of all RNN states. In this work, we use 400 as the dimension of user embeddings for TRNNs, DeepCare and TLSTM.

[2]Implementation details for Doc2Vec are as follows: Dimension of the embeddings is 400. We use PV-DM training algorithm with window size 5 and negative sampling where 10 noisy actions are drawn.

Results: Table 5 reports the overall RIG and AUC of all embedding approaches. Note that BoW with or without TF·IDF performs better than Doc2Vec on CompanyA dataset. It shows that although sequential information is important for modeling user behavior, it is difficult to capture through co-occurrences with partial ordering in Word2Vec-like models [21]. In addition to that, in some cases, it is effective to consider entire corpus to capture the complex nature of user embedding rather than to compress them to latent representations without considering temporal information. In contrast to that, compared with all of the baselines including BoW, BoW + TF·IDF, Doc2Vec, conventional RNNs and time-aware models such as DeepCare and TLSTM, our model TRNN shows better performance. We believe that such improvements in conversion prediction have the potential to lead to significant revenue increment in the digital marketing area.

4.3 Preferred Mobile Applications

Motivation: Different from CompanyA dataset, CompanyB dataset is collected from user behavior logs on multiple channels including mobile applications, SNS and display advertisements. Among them, to further evaluate the quality of embeddings computed by our approach in the context of recommendation systems, we define a multi-class classification task based on mobile application logs, which have been recorded whenever users launch one of this company's mobile applications. Largely, there are 18 different applications without version and platform information. We assume that individual users will show correlated user behavior even at the different channels.

Experimental setting: Figure 6 shows the distribution of the number of users and the cumulative frequency. Among the 18 applications, we select the top-8 most popular applications as our target applications in this task. We compute the number of launches per individual user for each of the applications served by this company, and use this statistics to generate label information for the multi-class classification task. More specifically, we conduct experiments in two different scenarios. First, we predict the most preferred application whose evaluation result will be referred to as **Acc-1**. In this case, each user only has one most preferred product. However, as shown in Figure 7, although the majority of users are interested in one application, still about 30% users are using more than one applications. To obtain more precise results based on this observation, we perform another experiment of predicting a set of applications that users have used, whose evaluation result will be represented as **Acc-All**. In this scenario, there could be more than one positive true label since a user may have used more than one product. More formally, given a set of user embeddings, we train and use a multi-class classifier to predict user preferred application(s) on the CompanyB dataset. We compare our approaches with the baselines described in the Section 4.2, because this task is also based on user embeddings computed from user behavior logs.

Classification Model: We use logistic regression with multinomial output [14] in this evaluation. Although one-vs-one and one-vs-rest settings with a set of binary classifiers have also been used for multiclass classification, multinomial logistic regression generally shows more accurate results and is faster to train on the larger scale dataset.

Results: Similar to the evaluation in Section 4.2, we use 400 as the dimension of user embeddings for TRNNs. For comparison purpose, we use both 400 and 800 as the embedding size for Doc2Vec and the conventional RNNs. For BoW and BoW + TF·IDF, the dimension of the user representation is equal to the size of the action set.

Table 6 shows the experimental results. In terms of Acc-1, the performance of TRNNs is 71.18%. It outperforms BoW, BoW + TF·IDF, Doc2Vec and conventional RNNs. Similar to the conversion prediction task, Doc2Vec shows the lowest accuracy, whereas TRNN models produce more than 30% improvement in accuracy with the same size of the user embeddings.

In terms of Acc-All, the accuracy of the proposed time-aware TRNN model is 70.54%, which shows that it performs better than all other baselines. Again, the results show that Doc2Vec has the lowest accuracy while TRNN brings a clear performance gain of up to 45%. The results demonstrate that modeling sequential patterns with TRNN leads to user embeddings that obtain good prediction results in downstream predictive analysis tasks.

Method	Dim.	Acc-1	Acc-All
BoW	1764	0.6708	0.6864
BoW + TF·IDF	1764	0.6702	0.6943
Doc2Vec	400	0.5559	0.4982
Doc2Vec	800	0.5467	0.4837
RNN	400	0.6997	0.6879
RNN	800	0.6985	0.6908
TRNN	400	**0.7118***	**0.7054***

Table 6: Accuracy of predicting users' preferred applications.

5 RELATED WORK

User Behavior Modeling: There have been a long stream of approaches that aim to analyze and leverage user behavior, so called User Behavior Modeling, to solve real-world problems in many areas such as recommender systems [11, 17], social networks [13], and online ads [1, 33]. Such approaches aim at providing seamless and personalized user experiences, which from the marketer's points of view is directly related to their revenue as well. To do this, Ashish et al. [20] proposed a classifier for Look-alike Modeling on tail campaigns which predicts conversion based on a sequence of events showing user behavior. More recently, Longqi et al [30] proposed a distributed representation learning model that produces user representations from Photoshop, and evaluate them in multiple applications. Although there is significant prior work in understanding user behavior and accurately predicting relevant target variables, user behavior modeling has been focused mostly on optimizing performance on a specific task for which ground truth labels were provided. As such, the user behavior that is captured in prior work is not general enough to be transferred to other types of decision making problems.

Representation Learning: Representation learning has been used to improve the performance on a wide variety of natural language processing (NLP) tasks. The main idea is to learn a dense fixed-size vector for each word in the vocabulary – the word embedding – as a byproduct of training a model to predict the context words of a target word and vice versa [21]. Models for learning word embeddings have been successfully extended for learning distributed representations of sentences, paragraphs, and entire documents [5, 18]. Their success in NLP tasks has led to their widespread use in many other fields [11, 16, 27]. More recently, Cristobal et al. [8] proposed to learn patient embeddings by applying ML techniques on tensors which consist of manual representations of a sequence of clinical events per user, whereas Edward et al. [7] introduced a Med2Vec model that learns embeddings for medical codes and visits from large-scale EHRs.

Recurrent Neural Networks: RNNs [3, 25] provide a powerful machine learning technique for learning the underlying structure in sequential data. They have been applied to many areas such as natural language processing [4, 10, 24], machine translation [6, 28], text classification [32] and image generation[12, 22]. Although RNNs can theoretically represent arbitrary time dependencies within sequences, they frequently suffer from vanishing and exploding gradient problems. To alleviate the vanishing gradient problem, Long Short-Term Memory (LSTM) [15] was proposed where gates are used to control the contribution of the long-term memory vs. the short-term behavior due to recent input data. Subsequently, Cho et al. [6] proposed the Gated Recurrent Unit (GRU) which includes a reset gate and an update gate that control how much the hidden unit remembers or forgets while processing a sequence. More recently, in order to handle temporal sequential data, time-aware RNN models such as DeepCare[23] and TLSTM[2] have been proposed. In these models, the gate structure of the standard LSTM

cell is changed and temporal information is used to control the contribution of long-term memory versus short-term memory. In contrast, instead of modifying the internal structure of standard LSTMs, we augment the input representation with two additional features: the time elapsed since the previous action, and the time elapsed since the start of the user session. For user behavior modeling such as predicting next user action, experimental results show that the proposed time-aware RNN (TRNN) outperforms both DeepCare and TLSTM. Furthermore, instead of using an auto-encoder, we create user representations from merging all of the RNN states over the sequence of user actions and demonstrate their utility in two predictive analysis tasks.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a time-aware RNN (TRNN) model for predicting user behavior and learning user embeddings from event sequence data. The contributions of this work are two-fold: 1) A sequential learning model that leverages temporal information and session information in order to model user behavior, and 2) A framework that induces user embeddings from event sequence data and uses them for multiple prediction tasks.

To evaluate our approaches, we first conducted experiments on the task of predicting next user actions, using different datasets. According to the results, TRNNs outperform bi-gram amd tri-gram models, as well as conventional RNNs and two recently proposed time-aware RNN models. Second, we generate a set of user embeddings with TRNN. To evaluate the obtained user embeddings, we train external classifiers to predict user conversion on CompanyA dataset, and user preferred mobile applications on CompanyB dataset. In both tasks, the TRNN-based embeddings perform better than baseline embedding approaches such as BoW, TF·IDF and Doc2Vec, and also better than the conventional RNN-based embeddings. The results demonstrate that our approach has the potential to contribute significantly to user behavior analysis, and benefit other types of prediction models on event sequence data.

TRNN leverages temporal user behavior in order to create user embeddings. However, given a user behavior sequence, user actions associated with different temporal information may have different levels of importance in the overall user embedding. To capture this aspect, we will investigate memory augmented neural networks. We also plan to explore methods within the variational inference family for learning user representations.

REFERENCES

[1] Josh Attenberg, Sandeep Pandey, and Torsten Suel. Modeling and Predicting User Behavior in Sponsored Search. In *KDD*. 1067–1076.

[2] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. 2017. Patient Subtyping via Time-Aware LSTM Networks. In *KDD*. 65–74.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks* 5, 2 (1994), 157–166.

[4] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* (2015).

[5] Yu Chen and Mohammed J Zaki. 2017. KATE: K-Competitive Autoencoder for Text. *arXiv preprint arXiv:1705.02033* (2017).

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[7] Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. 2016. Multi-layer representation learning for medical concepts. In *KDD*. 1495–1504.

[8] Cristóbal Esteban, Danilo Schmidt, Denis Krompaß, and Volker Tresp. 2015. Predicting sequences of clinical events by using a personalized temporal latent embedding model. In *ICHI*. IEEE, 130–139.

[9] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-Scale Bayesian Click-Through Rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. In *ICML*.

[10] Alex Graves and others. 2012. *Supervised sequence labelling with recurrent neural networks*. Vol. 385. Springer.

[11] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *KDD*. ACM, 1809–1818.

[12] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623* (2015).

[13] Ido Guy, Naama Zwerdling, David Carmel, Inbal Ronen, Erel Uziel, Sivan Yogev, and Shila Ofek-Koifman. 2009. Personalized Recommendation of Social Software Items Based on Social Relations. In *RecSys*. 53–60.

[14] Joseph M Hilbe. 2011. Logistic regression. In *International Encyclopedia of Statistical Science*. Springer, 755–758.

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

[16] Douwe Kiela and Léon Bottou. 2014. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics.. In *EMNLP*. 36–45.

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37.

[18] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*. 1188–1196.

[19] Lei Li and Tao Li. 2013. News Recommendation via Hypergraph Learning: Encapsulation of User Behavior and News Content. In *WSDM*. 305–314.

[20] Ashish Mangalampalli, Adwait Ratnaparkhi, Andrew O Hatch, Abraham Bagherjeiran, Rajesh Parekh, and Vikram Pudi. 2011. A feature-pair-based associative classification approach to look-alike modeling for conversion-oriented user-targeting in tail campaigns. In *WWW*. ACM, 85–86.

[21] 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.

[22] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759* (2016).

[23] Trang Pham, Truyen Tran, Dinh Phung, and Svetha Venkatesh. 2016. Deepcare: A deep dynamic memory model for predictive medicine. In *PAKDD*. Springer, 30–41.

[24] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664* (2015).

[25] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, and others. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.

[26] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15, 1 (2014), 1929–1958.

[27] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *ICML*. 843–852.

[28] Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017. Sentence embedding for neural machine translation domain adaptation. In *ACL*, Vol. 2. 560–566.

[29] Chenyan Xiong, Taifeng Wang, Wenkui Ding, Yidong Shen, and Tie-Yan Liu. 2012. Relational Click Prediction for Sponsored Search. In *WSDM*. 493–502.

[30] Longqi Yang, Chen Fang, Hailin Jin, Matthew D. Hoffman, and Deborah Estrin. 2017. Personalizing Software and Web Services by Integrating Unstructured Application Usage Traces. In *WWW Companion*. 485–493.

[31] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Zi Huang. 2014. A Temporal Context-aware Model for User Behavior Modeling in Social Media Systems. In *SIGMOD*. 1543–1554.

[32] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898* (2017).

[33] W Zhang, L Chen, and J Wang. 2016. Implicit look-alike modelling in display ads: Transfer collaborative filtering to ctr estimation. arXiv preprint. *arXiv* 1601 (2016).

[34] 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. In *AAAI*. 1369–1375.

[35] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H. Chi. 2015. Improving User Topic Interest Profiles by Behavior Factorization. In *WWW*. 1406–1416.