Purchase Prediction based on Recurrent Neural Networks with an Emphasis on Recent User Activities

Quanyu PIAO†, Joo-Young LEE††, and Tetsuya SAKAI†

† Faculty of Science and Engineering, Waseda University
3–4–1 Okubo, Shinjuku-ku, Tokyo 169–8555, Japan
†† Wider Planet, Inc.
KFAS Building 8F, 211, Teheran-ro, Gangnam-Gu, Seoul, Korea
E-mail: †quanyu.piao@ruri.waseda.jp, ††jooyoung@widerplanet.com, †††tetsuyasakai@acm.org

Abstract Existing purchase prediction models for E-commerce are often limited in accuracy when they overly concentrate on specific user activities. Hence, we propose three methods of aggregating user activities in over a time sequence in such a way that recent activities are emphasized compared to old ones. The compression of the original data and reduction of training time is significant, and the result shows that the models trained by the Fibonacci Aggregation give the best comprehensive performance of the three aggregating methods.

Key words Recommendation System, Sequential Data, Machine Learning, Recurrent Neural Networks

1 INTRODUCTION

Advertisement recommendation is an important part in the E-Commerce field. The quality of the recommendation will highly affect user behaviors. Whether the user will click and buy will give a tremendous impact on profits of the advertisers. The user behaviors include time information (timestamp), commodity information (item ID, category, etc.), event (click, buy, etc.). The user behaviors have the following characteristics: Sequentiality, Uncertainty, and Multidimensionality, that lead to challenges for the user behavior prediction task and corresponding recommendations.

Sequentiality The events given by the users are strictly sequential so that the prediction models should pay more attention to the time information.

Uncertainty The user behaviors are complex and changeable. Short and simple click may lead to a buy action; however, it is also possible for a user buying nothing after a long time session with many clicks.

Multidimensionality There are a huge number of commodities in the world with diverse categories. Also, users may have different behaviors at different times such as promotion time.

There are many models using classic machine learning methods such as Random Forest [1] and Gradient Boosting Decision Tree (GBDT) [2] to predict the user behavior. Recently, some simple prediction models such as Factorization Machine (FM) [3] [4] series are also widely used. However, because of the complexity of the user behavior data mentioned, all of these methods need careful data analysis and feature engineering before the model training.

Neural Network (NN) models have developed rapidly in recent years. They can discriminate and select the features automatically to prevent from complicated feature engineering, and finally provide good prediction and recommendation results [5]. Especially, the Recurrent Neural Networks (RNN) have great adaptability for sequential data. We believe that the RNN will also perform well in this task for user behaviors that are strictly sequential.

The present study is based on RNN to predict the buy action intention of the users and explores the prediction performance in the case where the models are trained by the aggregated data.

2 RELATED WORK

In the RecSys 2015 challenge [6], Wu, et al. [5] proposed an RNN model to predict the buy session by simply using the time-stamp of the clickstream data to analyze the item and session features. Also, Sheil, et al. [7] predicted purchasing intent by considering the item price variance and doing skip connections to combine the original input with every RNN layers. Besides, Gligorijevic, et al. proposed a time-aware approach [8] to model user behavior which can capture implicit signals of users' conversion intents.

3 METHODOLOGY

3.1 Data Aggregation

The proposed approach focuses on data preprocessing. Our starting point is analogous to the way people think and remember when browsing products in online shopping. Actually, people do not remember exactly when and what items they browsed or bought but only have a rough impression. For example, a user may only remember that he browed spoons, chopsticks, and bows yesterday morning, and especially a stainless steel spoon impressed him a lot. In this example, the time information is no longer as accurate as the data record we have, and the correspondence relationship between time and items has been blurred. Also, the information about the products the users browsed is not completely remembered. The farther the behavior is, the less information the users remember. Based on this view, we propose a data aggregation process that aggregating ascending numbers of activities to one activity in the data preprocessing. Firstly, the user behavior records are too huge to handle. The aggregation of the original data can summarize and compress the data. Secondly, predicting behaviors for users who have short activity records is a challenge. As user records have quite different lengths, after the aggregation, the overall record length will be significantly reduced. Hence, not only the users with a small number of activities we can focus, but also can consider the long-activity users.

A session is full of click records by a user, and each record including this session's ID (SSID), clicked timestamp, item and its category. We apply our aggregating methods to every session. The aggregating methods are detailed as follows. We propose three aggregating methods: Natural Aggregation, Fibonacci Aggregation, and Exponent Aggregation.

3.1.1 Aggregation definitions

Firstly, we define the number of activities we aggregated within one session in every step P_i as:

$$P_i, \quad i = 1, 2, 3, \dots$$

Then, the total numbers of activities aggregated $N_{aggregated}$ can be represented as:

$$N_{aggregated} = \sum_{i=1}^{max(i)} P_i, \quad i = 1, 2, 3, \dots$$

Obviously, $N_{aggregated}$ should be smaller than the longest session in the dataset:

$$N_{aggregated} \leqslant maxLength(session)$$

For Natural Aggregation showed in Figure 1, we aggregate numbers of activities in natural number sequence, so P_i will be:

$$P_i = i, \quad i = 1, 2, 3, \dots$$

In Fibonacci Aggregation, the Fibonacci number sequence can be represented as:

$$F(i) = F(i-1) + F(i-2), \quad i = 1, 2, 3, \dots$$

So, the number of processed activities in every step will be:

$$P_i = \begin{cases} 1, & i = 1 \text{ and } 2 \\ F(i), & i = 3, 4, 5, \dots \end{cases}$$

Similarly, in the Exponent Aggregation, we choose the base as 2, therefore the P_i will be:

$$P_i = 2^{i-1}, \quad i = 1, 2, 3, \dots$$

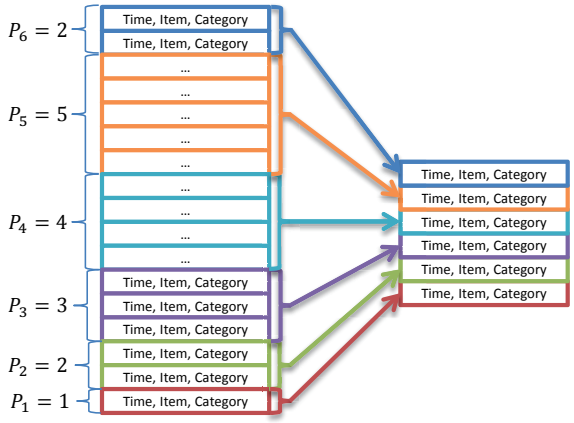

Figure 1: Natural Aggregation

3.1.2 Aggregation method in every step

The features in the records may include timestamp, item ID and item category. In every aggregating step, for those numerical features, like timestamp, we get the average of the features:

$$feature_{new} = avg(\sum feature_{original})$$

For those non-numerical features, like item ID and category, we choose the most clicked feature in this aggregating step. Note that if some features are correlative, the new features should also be correlative. For example, a most clicked item and its category should be:

$$feature_{newItem} = mostClicked(feature_{clickedItemList})$$

$$feature_{newCategory} = Category(feature_{newItem})$$

3.2 Embedding

Since the advent of Word2Vec [9], embedding has been a common technology in Machine Learning. It can map the high dimensional sparse vectors, like one-hot representation, to low-dimensional dense vectors. Embedding plays a vital role not only in Natural Language Processing (NLP) tasks, but also entity representation [10] in Knowledge Base tasks.

We embed time information, item IDs, and tem categories in the user behavior records to solve a Recommendation System prediction task.

3.3 Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNN) have a node structure that can be cycled so that it can handle the sequential input. And there are inverted sentences in some situations. The Bi-directional RNN (Bi-RNN) models can consider the context in order and reverse to catch more accurate meanings of the sentences.

However, original RNN is susceptible to the problem of gradient disappearance or explosion, which will stop the training or let the network unable to convergence, due to the simple cell structure. The emergence of Long Short-Term Memory (LSTM) attempts to alleviate this problem.

We utilize an LSTM in our models. As for the Bi-LSTM, we believe that the bi-directional model can get more user action information from E-Commerce clickstream data to perform better.

4 EXPERIMENTS

4.1 Dataset

The RecSys 2015 Challenge dataset [6] published by YOOCHOOSE provides user clickstream data for training and testing our prediction models. It consists of 9.2 million user sessions. The sessions are anonymous and consist of a chronological sequence of events describing user behaviors. There are two kinds of user actions in the RecSys 2015 dataset: click and purchase. The click data including session ID, timestamp, the item user clicked and the item's category, while the buy data including the session ID and timestamp, the purchased item ID and its price and quantity.

Table 1: RecSys 2015 dataset overview

Total clicks	Click sessions	Buy sessions	Unique Item	Unique Category
33,003,944	9,249,729	509,696	52,739	339

Table 2: RecSys 2015 dataset click action sample

SSID	Timestamp	Item ID	Category
1	2014-04-07T10:51:09.277Z	214536502	0
1	2014-04-07T10:54:09.868Z	214536500	0

Table 3: RecSys 2015 dataset buy action sample (Timestamp column omitted)

SSID	Item ID	Price	Quantity
420374	214537888	0	1
420374	214537850	0	1

4.2 Data Preparation

We use the whole RecSys 2015 dataset. As most sessions consist of less than 50 activities, we choose the activities which length are shorter or equal to a suitable value in each aggregating method (Table 4). We applied padding to sessions that are shorter than the value. However, the imbalance between buy and the not-buy session is still an unavoidable problem: only 5% of sessions ending with one or more buy actions. This makes the purchase prediction, which is a positive sample, quite hard to implement directly. Thus, we sampled nearly the same number of buy data from the click data to get a balanced training dataset.

Table 4: Activity length limit and shrunken percentage in each aggregating method

Aggregation	Activity Length Limit	Shrunken Percentage
Natural	36	99.60%
Fibonacci	33	99.46%
Exponent	31	99.35%

Our task is to predict whether the user will give a purchase action or not. We process each property of click and buy data as follows:

- Click data: All properties are unchanged.
- Buy data: Only session ID, timestamp and item ID are retained.
- Session ID is discarded in the training and prediction (of course we retain this during session grouping by the ID).
- Timestamps are split into month, day, day of week, hour, minute, and second. All sessions happened in 2014, so we discard the year property. And we believe that the minute and second features are not critical to the prediction so that we discard them when we are using the data.
- Category IDs are unchanged.
- Target data: For every session in click data, if the session ID exists in the buy data, this means the session is a buy session. We assign number 1 to show that this is a buy session. Otherwise, set number 0 to show that the session not ended with a buy action.

Finally, we group the session into one row, including session ID, session features, and session buy. There are month, day, day of week, hour, minute, second, item ID, and item category in the session feature column. The item ID and item category field are recorded as two lookup tables and converted to indexes. Table 5 shows the sample after the basic feature engineering. Then 6 features (except minute and second features) are embedded into dense vectors.

Table 5: Feature data sample

SSID	Feature	is_buy
7	4:2:3:6:38:53:389:0|4:2:3:6:39:5:847:0	0
367	4:7:1:9:42:12:566:0|4:7:1:9:43:14:566:0	1

4.3 Activity Aggregation

For the time data including month, day, day of week, hour, minute, and second, firstly we convert to seconds from the year 1970, then we calculate the average time in an aggregation step as the new timestamp, then restore to the time features. As for the item ID and category, we choose the most clicked item and its category in this aggregating step. There may remain activities after the aggregation. These remaining activities will be processed in the same way mentioned above. Table 4 shows the change of the dataset after activity aggregation.

4.4 Model

In this work, we construct an LSTM model to handle the sequential data. We prepared two LSTM models which are simple and complex. Then every model applied with Bidirectional feature and set dropout rate as 0.5 during training. All structures of the LSTM models as follows:

Table 6: LSTM Model Structures

	Hidden size	Hidden layer
Simple	32	1
Complex	128	3

The last output of the LSTM will be full connected [9] to the final output layer, which is a neuron that provides the probability of a buy session. We use a sigmoid cell to map the output to [0, 1] space and set a threshold as 0.5, which means the outputs bigger than 0.5 will be predicted as a buy session. Figure 2 shows the overview of the model structure.

4.5 Experiment Results

For the buy prediction task in E-Commerce, we evaluate the predictions by measuring buy and not-buy session's precision, recall, and the ROC_AUC score.

Table 7 shows the approximate value of data compress rate and training time reduction rate. Firstly, we can greatly

Figure 2: Model structure with feature embedding widths

compress the original data to aggregated data to save space usage. Then, using aggregated data can reduce the training time, especially for complex neural networks.

Table 8 and Table 9, which are the results from the balanced and original dataset, have a similar shape. Except the results of Fibonacci Aggregation, in each table, the simple LSTM models trained by aggregated data have better performance in not-buy precision and buy recall, but worse performance in buy precision and not-buy recall than the model trained by the unaggregated data. We consider that the simple models trained by the natural and exponent aggregated data have more tendency to predict user buy action so that causing the better buy recall result. In the meantime, they lose not-buy recall performance and get a little better result in not-buy precision.

However, for the complex models, the results are completely opposite to the simple models' results. We believe that the models gain more ability to predict not-buy action, which causes the results: better performance in not-buy recall and buy precision, worse performance in not-buy precision and buy recall.

Finally, let's concentrate on the results of the models trained by the Fibonacci aggregated data. Compared with the other two aggregation methods, the changes are more stable, or, the shape of the results are similar with the models trained by the unaggregated data: in every evaluate target, the result is reduced or similar (except buy recall in the simple models), with the best ROC_AUC values. We believe that the models trained by the Fibonacci Aggregation give the best comprehensive performance of the three aggregating methods. With 97.5% - 98.5% of the best performance (the results trained by unaggregated data), it provides data compression and huge training acceleration.

Table 7: Data compress rate and training time reduction rate (Approximation)

	Dataset compressed rate	Training time reduction rate
Simple	20%	15%
Complex		50%

Table 8: Results of balanced test dataset (buy: not-buy = 1:1)

	Buy Precision	Not-buy Precision	Buy Recall	Not-buy Recall	ROC_AUC
Simple	**0.7330**	0.7387	0.7363	**0.7354**	**0.7358**
Simple with Natural Aggregation	0.6645	0.7731	0.8247	0.5892	0.7070
Simple with Fibonacci Aggregation	0.7086	0.7384	0.7528	0.6927	**0.7228**
Simple with Exponent Aggregation	0.6419	**0.7816**	**0.8492**	0.5326	0.6909
Complex	0.7178	**0.7476**	**0.7585**	0.7058	**0.7322**
Complex with Natural Aggregation	**0.7414**	0.6676	0.5994	**0.7937**	0.6965
Complex with Fibonacci Aggregation	0.7029	0.7270	0.7390	0.6899	**0.7145**
Complex with Exponent Aggregation	0.7324	0.6817	0.6352	0.7709	0.7031

Table 9: Results of original test dataset (buy: not-buy = 1:18)

	Buy Precision	Not-buy Precision	Buy Recall	Not-buy Recall	ROC_AUC
Simple	**0.1375**	0.9795	0.7339	**0.7343**	**0.7341**
Simple with Natural Aggregation	0.1039	0.9829	0.8221	0.5906	0.7064
Simple with Fibonacci Aggregation	0.1249	0.9797	0.7503	0.6959	**0.7231**
Simple with Exponent Aggregation	0.0957	**0.9842**	**0.8528**	0.5328	0.6928
Complex	0.1295	**0.9806**	**0.7582**	0.7058	**0.7320**
Complex with Natural Aggregation	**0.1429**	0.9714	0.5948	**0.7941**	0.6945
Complex with Fibonacci Aggregation	0.1216	0.9786	0.7385	0.6915	**0.7150**
Complex with Exponent Aggregation	0.1390	0.9734	0.6368	0.7713	0.7040

5 CONCLUSIONS

In this work, we proposed three kinds of data aggregation methods: Natural, Fibonacci, and Exponent Aggregation. Then we investigated and discussed the effects made by the aggregating of the original data. Firstly, the result shows that the models trained by balanced and unbalanced original dataset have similar result shape. Then, except Fibonacci Aggregation, for simple LSTM model, the models perform better in not-buy precision and buy recall, but worse performance in buy precision and not-buy recall. However, for complex LSTM model, the results are completely opposite. On the other hand, the Fibonacci Aggregation performs similar with the results trained by unaggregated data. It gives the best comprehensive performance of the three aggregating methods. The method provides data compressing and huge training acceleration with 97.5% - 98.5% of the best performance.

During the experiments, the biggest problem is data overfitting. We had to early stop the training process to get the best training results. However, all of the ROC_AUC scores are not satisfied, which means that the ability of models is unsatisfactory. The problems may be as follows:

- The models were not trained enough. Probably there are some problems with the model and hyperparameters.
- We did not apply dynamic RNN to deal with different real session length. The padding blocks may interfere with the training process.
- Some of the features, such as item IDs, may need more analysis to convert the original features to some more meaningful features.

We consider that we should pay more attention to data analysis and feature crafting. Besides, advanced and complex neural networks is also a research point. Moreover, we should propose better solutions to handle more general and imbalanced E-Commerce data in the future.

ACKNOWLEDGMENTS

Special thanks to all of the members in Real Sakai Lab. Thanks for their important suggestions and guidance. Also special thanks to Nao Toyama, who always gives me emotional support.

References

[1] Breiman, Leo. “Random Forests.” Machine Learning 45 (2001): 5-32.

[2] Friedman, Jerome H.. “Greedy function approximation: A gradient boosting machine.” (2001).

[3] Rendle, Steffen. “Factorization Machines.” 2010 IEEE International Conference on Data Mining (2010): 995-1000.

[4] Rendle, Steffen. “Factorization Machines with libFM.” ACM TIST 3 (2012): 57:1-57:22.

[5] Wu, Zhenzhou et al. “Neural Modeling of Buying Behaviour for E-Commerce from Clicking Patterns.” RecSys ’15 Challenge (2015).

[6] Ben-Shimon, David et al. “RecSys Challenge 2015 and the YOOCHOOSE Dataset.” RecSys ’15 (2015).

[7] Sheil, Humphrey et al. “Predicting Purchasing Intent: Automatic Feature Learning using Recurrent Neural Networks.” ArXiv abs/1807.08207 (2018): n. pag.

[8] Gligorijevic, Djordje et al. “Time-Aware Prospective Modeling of Users for Online Display Advertising.” ArXiv abs/1911.05100 (2019): n. pag.

[9] Mikolov, Tomas et al. “Efficient Estimation of Word Representations in Vector Space.” CoRR abs/1301.3781 (2013): n. pag.

[10] Liu Zhiyuan,Sun Maosong,Lin Yankai, et al. “Knowledge Representation Learning: A Review[J].” Journal of Computer Research and Development, 2016, 53(2): 247-261.