Weighted Averaging of Various LSTM Models for Next Destination Recommendation

Shotaro Ishihara*
Nikkei, Inc.
Tokyo, Japan
shotaro.ishihara@nex.nikkei.com

Shuhei Goda*
Wantedly, Inc.
Tokyo, Japan
shu@wantedly.com

Yuya Matsumura*
Wantedly, Inc.
Tokyo, Japan
yuya@wantedly.com

ABSTRACT

This paper describes the 6th place approach to Booking.com WSDM WebTour 2021 Challenge, which is a challenge with a task of predicting travellers' next destination. We, in the team "hakubishin3 & u++ & yu-y4", trained four types of Long short-term memory (LSTM) models, and achieved the final score: 0.5399 by weighted averaging of these predictions. There are some differences in these models in feature engineering, multi-task learning, and data augmentation. Our experiments showed that the diversity of the models boosted the final result. Our codes are available at https://github.com/hakubishin3/booking-challenge-2021 and https://github.com/upura/booking-challenge-2021.

CCS CONCEPTS

• **Information systems** → **Information systems applications**; **Recommender systems**;

KEYWORDS

Booking.com WSDM WebTour 2021 Challenge, Recommender systems, Long short-term memory

1 INTRODUCTION

Booking.com is one of the world largest online travel agencies. Its mission is to make it easier for everyone to experience the world, and it seeks the way of using information technology that helps reduce the time and effort regarding travel [4]. One of the implementations is the recommendation of the destination. According to Booking.com, many of the travellers go on trips which include more than one destination. Suggesting travel destinations based on the past history would be helpful for travellers.

In 2021, Booking.com published dataset and organized a challenge with a task of predicting travellers' next destination. In this challenge, participants consider a scenario in which the users of Booking.com make a reservation and immediately suggest options for extending their trips.

The rest of this paper is organized as follows. In section 2, the overview of the challenge is described. Before our own approach, section 3 describes the previous works by the organizer that we followed. From section 4 to 6, we present our solution step by step. Each section shows the architecture of the neural network models, the input features, and the training methods. In section 7, we report the experimental results of our proposed method. The final section provides a conclusion of this paper.

*These authors contributed equally to this work.

2 CHALLENGE TASK

2.1 Metrics

The goal of the challenge is to develop a strategy for making the best recommendation of cities as a travellers' next destination. The quality of the recommendations are evaluated by using *Precision@4* metric. In other words, it is considered correct when the true city is one of the top four suggestions for each trip.

2.2 Dataset Description

The training dataset consists of over a million of anonymized hotel reservations, with the following columns:

- *user_id*: User ID
- *check-in*: Reservation check-in date
- *checkout*: Reservation check-out date
- *affiliate_id*: An anonymized ID of affiliate channels where the booker came from (e.g. direct, some third party referrals, paid search engine, etc.)
- *device_class*: desktop/mobile
- *booker_country*: Country from which the reservation was made (anonymized)
- *hotel_country*: Country of the hotel (anonymized)
- *city_id*: city_id of the hotel's city (anonymized)
- *utrip_id*: Unique identification of user's trip (a group of multi-destinations bookings within the same trip)

The evaluation dataset is constructed similarly, however the city_id and hotel_country of the final reservation of each trip are concealed. We are required to predict the city_id.

The distribution of the city_id appearing in the dataset is long-tailed. Though there are about 40,000 candidates, we can achieve the score of 0.036 by just suggesting the top four most frequent cities in the dataset.

3 BASELINE APPROACH

The organizer's previous works [6, 9] were a good starting point. We implemented a neural network model based on these references as a baseline for our solution. This model consists of recurrent neural networks (RNNs) model that handles a series of features. As features, affiliate_id, device_class, booker_country, and city_id are used. These sequential categorical features are encoded by GRU cell [5], and the output is converted to probability values for each city_id through a Softmax layer. The top four city_ids with the highest probability values can be regarded as the model's recommendation.

Some beneficial ideas are also explained in the previous works. First, the experiment showed that adding features other than the series of city_id greatly improved the performance[9]. This made us

realize the importance of feature engineering in this task. The second is how the features should be used. Two basic merge functions, concatenation and element-wise multiplication, were compared in the experiment[9]. And the result showed that the former was better. Finally, some tips for training were introduced. For example, the following techniques were shown in the previous work of [6].

- The data should be sorted by series length to reduce the number of padding in making batches.
- The duplicates should be eliminated when the same city_id is consecutive.

4 MODEL ARCHITECTURE

We prepared two types of model architecture. Both architectures are described in this section.

4.1 Long short-term memory

As RNNs, we adopted Long short-term memory (LSTM) [2] instead of GRU. Figure 1 shows an architecture of a LSTM model. There are no major differences from the baseline presented in section 3, except for the replacement of the RNNs unit. The differences in feature engineering and training process are described in section 5 and 6.

4.2 LSTM with Multi-task Learning

We also used a LSTM model with another type of structure, shown in Figure 2. This is an extension of the model with the concept of multi-task learning (MTL). MTL is a training paradigm in which machine learning models are trained with the dataset from multiple tasks simultaneously [3]. It is known that there are some advantages like improved data efficiency, reduced overfitting through shared representations, and so on.

In this challenge, we built an architecture that predicts not only the city_id but also the hotel_country at the same time. Since there is an inclusion relationship between hotel_country and city_id, we thought that the prediction of hotel_country would also contribute to the quality of the prediction of city_id. To predict hotel_country is an easier task than city_id. When a hotel_country is given, the candidates of city_id are limited. This can help our model to predict a correct city_id.

5 FEATURES

We generated some features from the original dataset, and removed some of the original dataset. Features can be divided into three types, categorical, numerical, and graphical variables. Each type of feature is briefly summarized in this section.

5.1 Categorical Features

There are seven different categorical features in the dataset, and six of them are used in the baseline as described in section 3. The only removed one is utrip_id, a unique identificator.

We added some categorical features. The followings are categorical features we used.

- *month_checkin*: Month of check-in.
- *past_city_id*: Previous city_id.
- *past_hotel_country*: Previous hotel_country.

5.2 Numerical Features

We extracted some numerical features. The days_stay is a feature that uses future information, we should be careful. In this challenge, check-in and checkout date of the targets are given. Therefore, days_stay which should not be available practically can be calculated, which is a useful feature since it indicates the characteristics of the targets.

- *days_stay*: The number of days staying in a current hotel.
- *days_move*: The number of days from a previous hotel checkout date to a current hotel check-in date.
- *num_checkin*: The number of check-in within utrip_id.
- *num_visit_drop_duplicates*: The number of unique city_id within utrip_id.
- *num_visit*: The number of city_id within utrip_id.
- *num_visit_same_city*: The number of duplicated city_id within utrip_id.
- *num_stay_consecutively*: The number of consecutive stay in the same city_id.

Figure 1: Model Architecture of LSTM

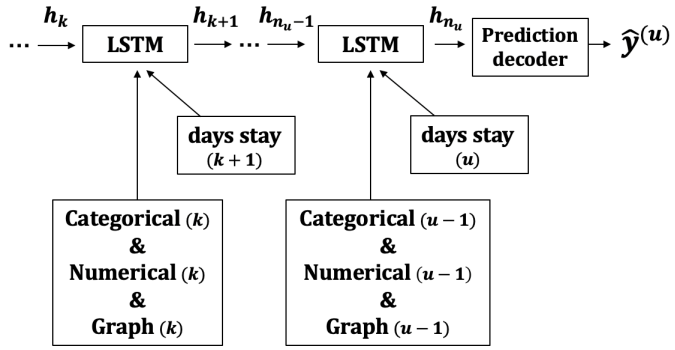

Figure 2: Model Architecture of LSTM with Multi-task Learning

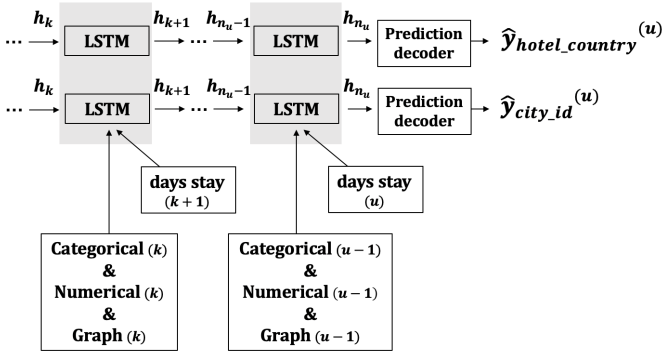

Figure 3: The embeddings of city_id calculated by Word2Vec compressed by UMAP

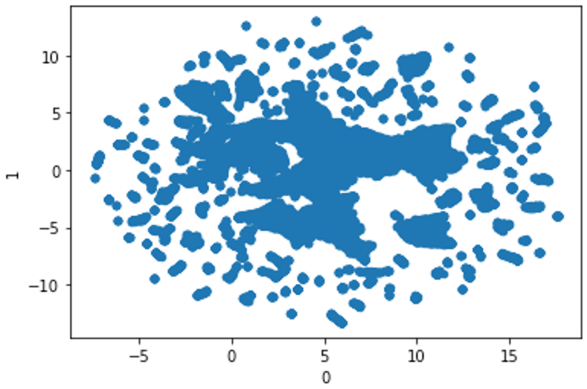

Figure 4: The distribution of the length of trips

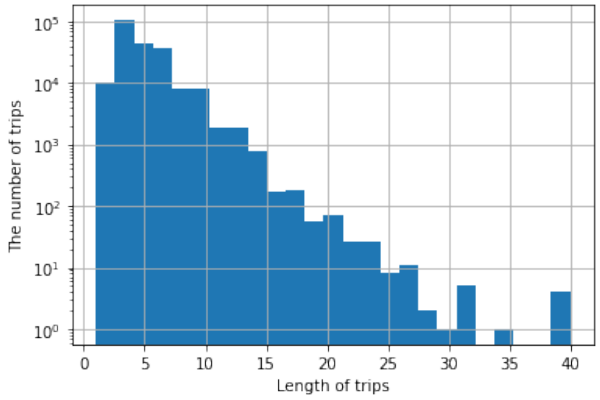

Table 1: Settings of Four LSTM Models

Model	MTL	Graph	Loss	Augmentation
LSTM 1	false	Word2Vec	CrossEntropyLoss	false
LSTM 2	true	Word2Vec	CrossEntropyLoss	false
LSTM 3	true	Word2Vec	CrossEntropyLoss	flip dataset
LSTM 4	false	PyTorch-BigGraph	FocalLoss	test dataset

In LSTM 3, booker_country was removed from categorical features as described in subsection 6.3.

5.3 Graph Features

Each sequence of trips are just a fragment of it. We believe that graph related features are important because it can lead to reconstruct geographical information. We believe that the graph related features are important because they can lead to reconstruct geographical information. We used Word2Vec [8] and PyTorch-BigGraph [1] to create graph features using the sequences of each trip.

Figure 3 is a scatter plot of city_id vectors calculated by Word2Vec compressed by UMAP [7]. When we used MTL architecture, the same embeddings were calculated regarding hotel_country.

6 TRAINING PROCESS

6.1 Loss Functions

CrossEntropyLoss is commonly used in multi-classification tasks. However, the classes of city_id that are required to be predicted in this challenge are imbalanced with long-tailed distribution. In order to suppress the bias of the loss caused by the class imbalance, we adopted FocalLoss [10] as one of the options of a loss function.

6.2 Validation Strategy

We chose Stratified K-Folds so that the distribution of the length of trips in each fold is equal. This is because there is a difference in the length of trips shown in Figure 4, which may affect the quality of the prediction. The number of folds was set to five. We trained 15 epochs in each fold, and used a model with the best validation score for predictions.

6.3 Data Augmentation

We believe that each sequence of trips can be flipped. In applying this data augmentation, booker_country must be removed in order to keep consistency of the dataset.

In this challenge, we can use the information of the evaluation dataset. The final city of each trip in the evaluation dataset is masked to be used for evaluation. Therefore, we created an additional dataset from the evaluation dataset by regarding the second city_id from the final reservation as the target of each trip and added it to the training data.

7 EXPERIMENTS

7.1 Settings

Table 1 shows the settings of each model. Each model was composed of the combination of the proposed architecture, features, and the training methods.

MTL architecture was used in LSTM 2 and 3. Graph features and loss function were different in LSTM 4. Data augmentation techniques were applied in LSTM 3 and 4.

7.2 Results for Each Model

Table 2 shows the results of our experiments for each model. The validation scores were calculated by averaging the predictions of all folds.

We can see that our proposed models outperformed the baseline. From the comparison between LSTM 1 to 3, it was observed that MTL worked in our experiment, and data expansion by flipping didn't work. LSTM 4 gave us the best result.

7.3 Weighted Averaging

A weighted averaging was used for the ensembling. In this challenge, we decided to use it because of the small opportunity of submissions and the high computational cost. Table 2 shows that the diversity of the models leads to higher scores. It is interesting

Table 2: Validation Scores of Four LSTM Models and Weighted Averaging

Model	Validation score	Weights
Baseline	0.4629	-
LSTM 1	0.4927	-
LSTM 2	0.4937	-
LSTM 3	0.4862	-
LSTM 4	0.5043	-
Weighted averaging 1	0.5156	(0.2, 0.15, 0.05, 0.7)
Weighted averaging 2	0.5149	(0.15, 0.10, 0.05, 0.7)
Weighted averaging 3	0.5160	(0.25, 0.20, 0.15, 0.4)

The four numbers in Weights mean the ratio of LSTM 1 to 4 respectively.

to note that even though LSTM 4 performed well on its own, the best result was obtained when the ratio was reduced to 0.4.

8 CONCLUSION

This paper described our approach to Booking.com WSDM WebTour 2021 Challenge. We trained four types of LSTM model, and weighted averaging of these predictions increased the accuracy. There is a diversity regarding feature engineering, multi-task learning, and data augmentation. In the end, we won the 6th place on the final leaderboard of this challenge.

ACKNOWLEDGMENTS

We thank the organizers of the Booking.com WSDM WebTour 2021 Challenge for the opportunity to participate in this interesting challenge.

REFERENCES

[1] Lerer Adam, Wu Ledell, Shen Jiajun, Lacroix Timothee, Wehrstedt Luca, Bose Abhijit, and Peysakhovich Alex. 2019. PyTorch-BigGraph: A Large-scale Graph Embedding System. In *Proceedings of the 2nd SysML Conference*. Palo Alto, CA, USA.

[2] Sarah Cohen, Werner Nutt, and Yehoshua Sagic. 1997. Long Short-Term Memory. *Neural Comput* 9, 8 (1997), 1735–1780.

[3] Michael Crawshaw. 2020. Multi-Task Learning with Deep Neural Networks: A Survey. *arXiv preprint arXiv:2009.09796v1* (2020).

[4] Goldenberg Dmitri, Kofman Kostia, Levin Pavel, Mizrachi Sarai, Kafry Maayan, and Nadav Guy. 2021. Booking.com WSDM WebTour 2021 Challenge. https://www.bookingchallenge.com/. In *ACM WSDM Workshop on Web Tourism (WSDM WebTour'21)*.

[5] Matthew Van Gundy, Davide Balzarotti, and Giovanni Vigna. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. https://doi.org/10.3115/v1/D14-1179. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, Doha, Qatar.* 1724––1734.

[6] Pavel Levin. 2018. *Modeling Multi-Destination Trips with RNNs.* Technical Report. DataConf 2018, October 4th, Jerusalem, Israel.

[7] Leland McInnes, John Healy, and James Melville. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv preprint arXiv:1802.03426v3* (2020).

[8] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781v3* (2013).

[9] Sarai Mizrachi and Pavel Levin. 2019. Combining Context Features in Sequence-Aware Recommender Systems. *ACM RecSys 2019 Late-breaking Results, 16th-20th September 2019, Copenhagen, Denmark* (Sept. 2019).

[10] Lin Tsung-Yi, Goyal Priya, Girshick Ross, He Kaiming, and Dollar Piotr. 2017. Focal Loss for Dense Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.