Explore next destination prediction

Yuanzhe Zhou
zhouyuanzhe@whu.edu.cn
WUHAN, CHINA

Shikang Wu
danny199607@gmail.com
BEIJING, CHINA

Chenyang Zheng
zhengchenyang01@baidu.com
BEIJING, CHINA

ABSTRACT

Next destination prediction problem by users' traveling sequence has provoked many researchers' attention recently. It is a perfect benchmark for a graph neural network or graph embedding algorithm, which have achieved state of the art for many kinds of tasks involving graph information. In this paper, we explore this problem by recurrent neural network and we have achieved 0.5741 top-4 accuracy.

CCS CONCEPTS

• **Information systems** → *Recommender systems.*

KEYWORDS

Booking.com Data Challenge, neural networks, deep learning, network embeddings, recommendation systems

1 INTRODUCTION

In this paper, we will introduce our model and method employed in the WebTour 2021 Challenge [2] holding by Booking.com. The source code for training and prediction can be accessed here:
https://github.com/zhouyuanzhe/booking_challenge-.

2 DETAILS OF MODEL

As the preprocessing, We pad/crop users' destination sequences to the same length 20 in order to use recurrent neural network. User sequences with length less than 2 are removed since they will not appear in our testing data. We use the data from both training data and test data for Word2Vec[5][6] pretraining. We choose window size 1 and embedding size ranging from 64 to 256 under mode sg (skip-gram).

The structure [Figure:1] of our model is shown blow. The input of our model are destination sequences and manual features. We use LSTM [3] (Long Short-Term Memory) and GRU [1] (Gated recurrent unit) to extract features from destination sequences. We use one layer of bi-directional GRU after one layer of bi-directional LSTM. The last output state of GRU is used as the feature of the destination sequence. Then we introduce the gate structure proposed in [4] to model the relationship between the last destination and the next destination. It is conducted in the following way,

$$d_{\text{output}} = \alpha \cdot d_n + (1-\alpha) \cdot RNN([d_1,\ d_2,\ ...,\ d_n])$$

$$\alpha = MLP([d_n,\ RNN([d_1, d_2, ..., d_n])])$$

where d_i are embedding vector of different cities.

We have 2 outputs, next destination and next country. The extra country information introduced here help us to decrease the loss by a lot. We apply softmax activation function to the output and use cross-entropy to optimize our model.

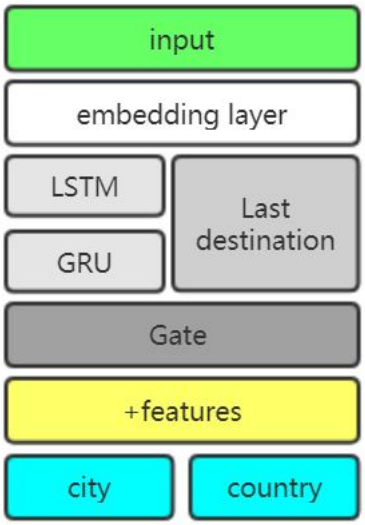

Figure 1: Model structure

2.1 Manual features

Since Booking.com allows us to use part of the information about the next destination, we made some manual features concerning the check-out date of the last destination, check-in date of the next destination and the duration the traveler will stay. The extra features improve our top-4 accuracy by 1.5%.

Our features include the features concerning time,

- Duration of last trip.
- Day of the month for check in.
- Day of the week for check in.
- Day of the month for check out.
- Day of the week for check out.

The sinus and cosinus value of the time are used. For example,

$$\text{day of the week sin} = \sin(\text{day of the week}/7)$$

$$\text{day of the week cos} = \cos(\text{day of the week}/7)$$

Other features like affiliate_id, device_class and booker_country are also included. The source is processed by a trainable embedding layer. The additional information improved the final result by 1.5%.

We use a 2 layers MLP to process the input of features. It is then concatenated to the output from RNN. We make the final prediction by a densely connected layer, with the activation function softmax. All the cities are treated as the promising destination (this part can be improved by carefully select positive and negative samples).

3 TRAINING STRATEGY

Our training strategy is 20 k-fold cross-validation with learning rate decay strategy, 'reduce lr on plateau'. We trained our model with Adam optimizer with learning rate 0.001 and batch size 8096. What contributes most to our final result is the two phase training strategy.

3.1 Pretraining for sequential recommendation

Pretraining has been proved to be efficient and effective for many tasks. Although RNN is not as capable as transformer, we can still improve its performance by using

We pretrain our RNN model by using the sub-sequences from the complete sequence. Assume that the complete sequence is,

$$(d_0,\ d_1,\ d_2, ...,\ d_n)$$

The sub-sequences are,

$$(d_0),\ (d_0,\ d_1),\ (d_0,\ d_1,\ d_2),\ ...,\ (d_0,\ d_1,\ d_2, ...,\ d_{n-1})$$

where d_i denotes the destinations.

3.2 Fine-tune

Pretraining can also be problematic because some intermediate destinations are not directly connected with user's origin. Thus some sub-sequences are noisy and the data distribution of sub-sequences is different from that of the original data. To handle this difference of data distribution, we only need to fine-tune our model with the original training data.

The following [Table 1] is the comparison of top-4 accuracy for local validation data between different settings.

	top-4 accuracy
LSTM	52.51%
LSTM+features	54.02%
LSTM+features+pretrain	54.95%
LSTM+features+pretrain+finetune	55.53%
LSTM+features+pretrain+finetune+test data	57.49%

Table 1: Top-4 accuracy on local validation

4 CONCLUSION

In this paper, we start from a very simple RNN model for next destination prediction. The main improvements are made through understanding the data and making the most of the data through data exploration. We have shown that pretraining for sequential recommendation system is beneficial and fine-tuning can improve the performance further. Our final result is competitive to SOTA.

REFERENCES

[1] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *CoRR* abs/1406.1078 (2014). arXiv:1406.1078 http://arxiv.org/abs/1406.1078

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

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

[4] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-Based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1831–1839. https://doi.org/10.1145/3219819.3219950

[5] 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.

[6] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50.