Hybrid Model with Time Modeling for Sequential Recommender Systems

Marlesson R. O. Santana
marlessonsa@gmail.com
Deep Learning Brazil – Federal University of Goiás
Goiânia, Goiás, Brazil

Anderson Soares
anderson@inf.ufg.br
Deep Learning Brazil – Federal University of Goiás
Goiânia, Goiás, Brazil

ABSTRACT

Deep learning based methods have been used successfully in recommender system problems. Approaches using recurrent neural networks, transformers, and attention mechanisms are useful to model users' long- and short-term preferences in sequential interactions. To explore different session-based recommendation solutions, Booking.com recently organized the *WSDM WebTour 2021 Challenge*, which aims to benchmark models to recommend the final city in a trip. This study presents our approach to this challenge. We conducted several experiments to test different state-of-the-art deep learning architectures for recommender systems. Further, we proposed some changes to Neural Attentive Recommendation Machine (NARM), adapted its architecture for the challenge objective, and implemented training approaches that can be used in any session-based model to improve accuracy. Our experimental result shows that the improved NARM outperforms all other state-of-the-art benchmark methods.

CCS CONCEPTS

• Computer systems organization → Embedded systems; **• Information systems → Recommender systems**; **• Computing methodologies → Sequential decision making**.

KEYWORDS

Recommender systems, Session-based recommendations, Recurrent neural networks, Hybrid model, Time modeling

1 INTRODUCTION

Deep learning approaches for recommender systems have garnered significant attention owing to their potential for modeling long- and short-term user preferences. Models using recurrent neural networks (RNNs), transformers, and attention mechanisms are handy for text, presenting encouraging results when used in session-based recommendation problems [2, 6, 8, 15–17], where sequential information and short-term context are fundamental in recommending the next item of the session.

In general, session-based approaches use only the information from the item's interaction sequence in the session as a predictor of the next item. However, in some domains, contextual information within and between sessions and global information are essential to modeling user behavior. Recommending travel destinations has several particularities, such as the sequence of cities on a trip can contain noise, financial constraints may lead to diversions, and destinations are highly correlated to the moment in time and duration of the trip [1, 7, 11] .

The *WSDM WebTour 2021 Challenge* [5] organized by Booking.com, focuses on recommending travel destinations in a session. This study describes our approach for the *WSDM WebTour 2021 Challenge* and proposes some approaches that can improve session-based models for recommender systems, especially with the highly time-dependent recommendation domains, noise information, and imbalanced classes.

2 THE CHALLENGE AND DATASET

Booking.com is the world's largest online travel agency. It is a platform where millions of travelers find accommodations for their trips, and millions of accommodation providers list their hotels, apartments, guest houses, and other lodgings. [1, 7].

Booking.com recently organized the *WSDM WebTour 2021 Challenge* [5]. The training dataset consists of over a million anonymized hotel reservations based on real data. Each reservation is a part of a customer's trip, which includes at least four consecutive reservations. The challenge's goal is to recommend the final city of each trip, and we evaluated models using an accuracy metric for the first four items suggested. For Accuracy@4, the metric value is one when the real city is one of the four main suggestions and zero otherwise.

3 PROPOSED APPROACHES

This section describes each approach used in the present study to improve our final model. First, we chose a state-of-the-art session-based model to improve (described in 3.1). Further, we created new features from statistical information about users and cities and added time modeling to focus on the travel problem particularity (described in 3.2 and 3.3). Also, we have reduced the effect of the imbalance dataset using specific loss functions and multitask modeling (described in 3.5 and 3.4). Finally, we improve the generalization of noise information through data augmentation (described in 3.6).

Each approach can be applied separately in other session-based models.

3.1 Model Architecture

We use the architecture of the neural attentive recommendation machine (NARM) [8] and adapt it for the traveling problem presented in this study. NARM uses an encoder-decoder architecture to address the session-based recommendations problems. According to the authors of the paper, the idea of NARM is to build a hidden representation of the current session using an RNN module. It converts the input click sequence $X = [x_1, x_2, ..., x_n]$ into a set of high-dimensional hidden representation with an attention signal that can be used to produce a ranking list of all items that can occur in the next step of the current session.

Our approach uses categorical and dense features of the user, city, and trip combined with the trip history. The core of the NARM module is the same as the original paper, and we just changed the size of the inputs, the bottleneck with hidden representation, and the outputs.

The features are concatenated and follow two paths. The first group of features passes through an attention layer before being fed into the NARM module. It is an important step for relating different positions of the same input sequence. In the second path, the features bypass the feature bottleneck generated by the NARM module, which improves the decoder by providing it with more contextual information on the session.

Figure 1 shows the final architecture proposed with modifications described in this study.

3.2 Feature Engineer

Our approach uses statistical features from users, cities, and trips combined with the trip's history information. For every trip, we extract more than 30 statistical features. Below we provide examples of the important features we use in the present study:

User Statistics

- Number of trips
- Number of cities visited
- Average number of unique cities visited per trip
- Average trip size
- Average trip duration
- Most frequently traveled month
- Number of unique cities the user visited
- Number of unique countries the user visited
- Device used for booking
- Booker country

City Statistics

- City interactions that have preceded the current trip
- City interactions that have preceded the current trip by user
- City interactions in the current trip
- Country interactions that have preceded the current trip
- Country interactions that have preceded the current trip by user

Furthermore, we employ user statistical features to create user embedding from an autoencoder trained on the same training split data. This approach proved helpful because only 0.7% of the users have made more than one trip; the vast majority are new users. In addition to providing a dense representation of the user features, the autoencoder ensures that similar users are close to each other in the created vector space. Moreover, the autoencoder was trained separately from the recommendation model. Figure 2 shows user embeddings colored by the most frequently traveled month.

3.3 Time Modeling

In terms of tourism, a trip's location is merely a piece of information that can tell how good the place is for a trip. The date (moment in time) and duration of the trip are as crucial as, for example, sometimes, the traveler can only visit certain places only in specific seasons or specific time windows. Sometimes, it is not feasible to stay for a few days more.

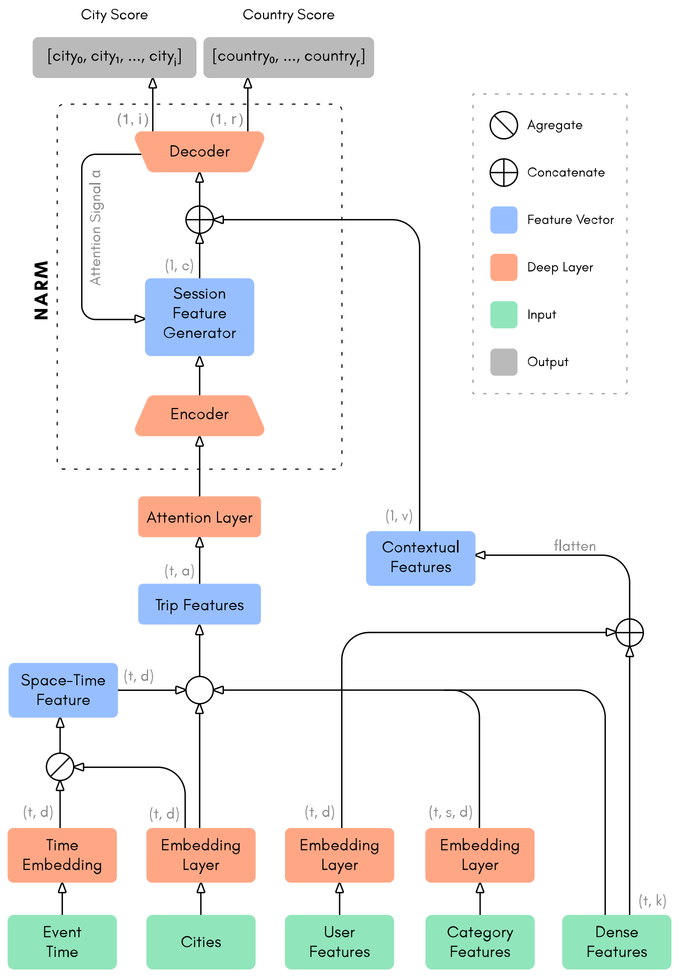

Figure 1: The general architecture of the proposed model. Our approach uses categorical and dense features of the user, city combined with the trip history. The features follow two paths. First, the features pass through an attention layer before being fed into the NARM module and in the second path, the features bypass the feature bottleneck generated by the NARM module, which improves the decoder by providing it with more contextual information on the session.

Embeddings are highly useful for modeling categorical features such as the location, and RNNs are remarkably good for capturing sequential data's time-dependence. However, they are limited when the events have irregular time intervals, such as duration of stay on a specific city trip. Therefore, we need to incorporate time interval explicitly into the model as a time-space embedding. In this case, we use the approach presented in [9] concatenated with a start trip month embedding for time-space embedding.

Therefore, each city embedding presented in a trip follows through a time embedding layer. We incorporate time (start trip month) and duration (how long users stayed in a city) information. In the end, we obtain a space-time embedding.

3.4 Multi-Task Learning

The data about the cities in our dataset are imbalanced and have high dimensionality. In contrast, the data about countries are less imbalanced and have lesser dimensionality. Thus, we adapt our model to predict both to try to improve models' gradient signal. According to [12], multi-task learning acts as a regularizer by introducing an inductive bias into the model. As such, it reduces the risk of overfitting and the Rademacher complexity of the model, and thus, it has the ability to fit random noise.

We used two targets in the present study. The city of the next step interaction was the main target that we used to evaluate the model, and the city's country was used as the second target. The second target was used only for regularization and as inductive bias in the primary target. Both targets use the same loss function, and the final loss is a combination of the two.

3.5 Loss Function

Focal loss [10] is very useful for training imbalanced datasets. It adds a weighted term in front of the cross-entropy loss to balance the gradient from positive and negative samples. Easily classified negatives comprise most of the loss and dominate the gradient; this effect can be reduced by using a focal loss strategy. We use focal loss for both targets, and the final loss is the average of each loss.

Formally, the focal loss is expressed as follows:

$$FL(p) = -\alpha_t(1-p)^{\gamma} log(p) \tag{1}$$

where γ is adjusts the rate at which easy examples are down-weighted and α is a prefixed value between 0 and 1 to balance the positive-labeled samples and negative-labeled samples. We use Equation 2 for calculation loss.

$$L(p_c, p_o) = \beta FL(p_c) + (1-\beta)FL(p_o) \tag{2}$$

where β is a balance parameter, and p_c and p_o are obtained from the target prediction.

3.6 Data Augmentation

Data augmentation techniques have been widely used to enhance image [14], text [18], or recommendations based models [4, 15]. The principal inputs of the model in the challenge are cities in the current session's history. These inputs can have noise, some users may take a long or short trip, and users can jump one or more cities on a popular route. We applied three different data augmentations steps to improve the accuracy and generalize the model.

The first is a sequence preprocessing step proposed in [4, 15], where we generate new samples using each trip's time-step. Given an input training trip $[c_1, c_2, c_3, ..., c_n]$, we generate the sequences and corresponding labels $([c_1], c_2), ([c_1, c_2], c_3), ([c_1, c_2, ..., c_{n-1}], c_n)$ for training. We filter only trips with more than four cities.

For each sample in training, we randomly choose a step in the trip to change. Given an input training trip $[c_1, c_2, c_3, c_4]$, we change it in three ways: remove a step as a dropout layer and generate sequences such as $[c_1, c_2, c_4]$; replace with a mask token (unknown) and generate sequences with the same size but with mask $[c_1, c_2, c_3, c_{UKN}]$; and replace with a similar city, such as $[c_1, s_2, c_3, c_4]$ where s_2 and c_2 are similar cities. The mask token will be the same as that used in production/inference mode when the model does not embed a specific city, and the similarity definition is the same as that we used for the Item-KNN model.

We apply both methods to make the model less susceptible to overfitting or noise information.

4 EXPERIMENTS

In this section, we present the experimental settings and results.

4.1 Dataset

We randomly partition the training dataset by trip, using 90% of data for training and 10% for model validation. We filtered trips with less than four cities or more than 10 and trips with a duration of more than 22 days in the training split. These values came from the initial analysis and are outliers. Table 1 shows the statistics of the dataset used in our experiments.

Table 1: Statistics of the dataset used in our experiments

Split	Trips	Users	Cities
train	195917	181480	38542
test	21769	21524	15488

4.2 Baselines

To show the effectiveness of our approach, we choose some popular baseline models used for session-based recommendation problems.

- **Popularity**: Popular predictor recommends the most popular city in last city's country on the current trip.
- **Item-KNN**: This baseline recommends the most similar city to the last city on current trip. The similarity is defined as the cosine distance between the trip vector of the city. It is similar to the co-occurrence approach. [3]
- **Caser**: This baseline is a convolutional neural network (CNN) approach for a sequential recommendation. The Convolutional Sequence Embedding Recommendation Model (Caser) embeds a sequence of recent items into an "image" in time and learns sequential patterns as local features. [16]
- **SASRec**: SASRec is a self-attentive approach that captures users' sequential behaviors and achieves state-of-the-art performance on sequential recommendation. [6]
- **NARM**: Neural Attentive Session-based Recommendation is an encoder-decoder architecture with an attention mechanism to model the user's sequential behavior, which is then combined as a unified session representation later. [8]

We choose the NARM model, which shows the best performance among the models listed above, such as our baseline model to improve during the competition. Therefore, our approach is an adaptation of the NARM model. Some approaches were modeled for the click prediction problem. For our experiments, we adapted the last layer of all deep learning models to produce a ranking list of all items, $y = [y_1, y_2, y_3, ..., y_n]$, that can occur on the current trip. Furthermore, we trained our model using the same loss function.

4.3 Experimental Settings

All our models were trained using similar parameters. We used 50-dimensional embeddings for all category features, and we normalized dense features using standard deviation. Optimization was carried out using Rectified Adam (RAdam), with a 0.001 learning rating and 0.01 weight decay, with mini-batch size fixed at 64. We truncated the trip history size using a fixed window of 10 time-steps with padding. The number of epochs was defined by early stopping using 10 steps, and we used cross-entropy as a loss function for most models or focal loss with 1 α and 3 γ.

Finally, we use the MARS-Gym framework [13] to model, train, and evaluate all experiments described in this paper. All experiments are present and can be reviewed at https://github.com/marlesson/booking_challenge.

4.4 Experimental Results

Each user's embeddings are presented in the 2-D plane in Figure 2. We can see that the month is a good predictor of user behavior, and, in general, users who travel in the same month also share similarities. We can also notice that there is a mixed cluster in the center, maybe belonging to users who are likely to travel on different dates with greater frequency. Furthermore, we can use this representation for new users to improve a cold-start recommendation.

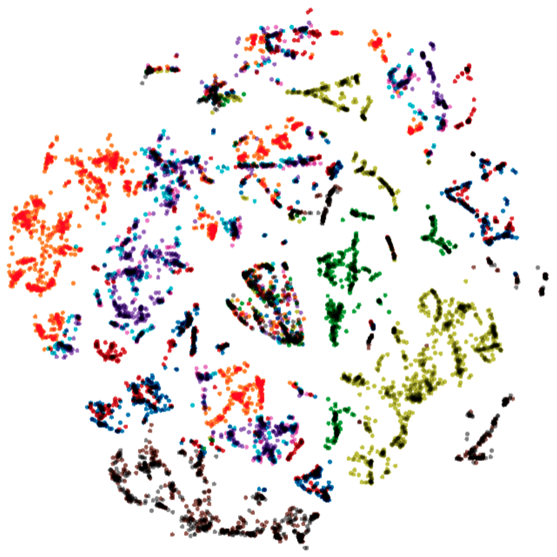

Figure 2: We used t-SNE to project 100-D into 2-D and produce this Figure. Each point represents a user, and the color shows the most frequent month that users traveling.

Each model's performance on the test dataset is summarized in Table 2. We can see that the Item-KNN method performs similarly to the Popularity method, which is a cold start method. Therefore, there is no benefit in adopting the Item-KNN method given the complexity posed by cold recommendations.

SASRec and Caser are very different approaches, the former based on attention mechanism and later on horizontal and vertical convolutions. Both showed similar results, with Caser performing slightly better. However, the best base baseline model was NARM. We can see that the original NARM outperformed state-of-the-art baselines using the same input information. Therefore, we chose NARM in this study to improve the architecture and training phase.

Table 2: Performance comparison of proposed method with baseline methods

Methods	ACC@4 (test)
Popularity	0.364
Item-KNN	0.371
SASRec	0.478
Caser	0.484
NARM	0.497
NARM V1	0.520
NARM V2	0.545

We evaluate two versions of NARM. In NARM V1, we use only improvements that were applied in the training and regularization steps, with the same input data were used for other baseline models, but applying approaches 3.3, 3.4, 3.5, 3.6. NARM V1 outperforms the original NARM with approximately +4.62% accuracy. This is a promising finding, as this indicates that these techniques can now be used for any other session-based models that are in need of improvement.

Finally, we apply all approaches present in this study to NARM V2, and it represents our final approach for the challenge. Figure 1 shows our final architecture. We found that NARM V2 outperforms the original NARM with approximately +9.66% accuracy. This supports the idea that when it is not possible to obtain an item or user metadata, session statistics can also be used as features.

5 CONCLUSION

In this study, we present our approach for the *WSDM WebTour 2021 Challenge*. We conducted several experiments using different session-based models to recommend the next destination on a trip. We modified the existing NARM model to add contextual information to the session, space-time modeling, and approaches to reduce the negative effect of class imbalance in the training phase. Our results show that it is possible to enhance the performance of state-of-the-art models through simple changes. The improved NARM outperforms all baseline models, and that the implemented training approaches can be used in any session-based recommendations model.

REFERENCES

[1] Lucas Bernardi, Themistoklis Mavridis, and Pablo Estevez. 2019. 150 successful machine learning models: 6 lessons learned at booking. com. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1743–1751.

[2] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*. 1–4.

[3] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, et al. 2010. The YouTube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*. 293–296.

[4] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. 2015. Artificial neural networks applied to taxi destination prediction. *arXiv preprint arXiv:1508.00021* (2015).

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

[6] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[7] Julia Kiseleva, Melanie JI Mueller, Lucas Bernardi, Chad Davis, Ivan Kovacek, Mats Stafseng Einarsen, Jaap Kamps, Alexander Tuzhilin, and Djoerd Hiemstra. 2015. Where to go on your next trip? Optimizing travel destinations based on user preferences. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1097–1100.

[8] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1419–1428.

[9] Yang Li, Nan Du, and Samy Bengio. 2017. Time-dependent representation for neural event sequence prediction. *arXiv preprint arXiv:1708.00065* (2017).

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[11] Sarai Mizrachi and Pavel Levin. 2019. Combining Context Features in Sequence-Aware Recommender Systems.. In *RecSys (Late-Breaking Results)*. 11–15.

[12] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098* (2017).

[13] Marlesson R. O. Santana, Luckeciano C. Melo, Fernando H. F. Camargo, Bruno Brandão, Anderson Soares, Renan M. Oliveira, and Sandor Caetano. 2020. MARS-Gym: A Gym framework to model, train, and evaluate Recommender Systems for Marketplaces. arXiv:2010.07035 [cs.IR]

[14] Connor Shorten and Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of Big Data* 6, 1 (2019), 1–48.

[15] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 17–22.

[16] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 565–573.

[17] Maksims Volkovs, Anson Wong, Zhaoyue Cheng, Felipe Pérez, Ilya Stanevich, and Yichao Lu. 2019. Robust contextual models for in-session personalization. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*. 1–5.

[18] Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196* (2019).