Using Deep Learning to Win the Booking.com WSDM WebTour21 Challenge on Sequential Recommendations

Benedikt Schifferer*
bschifferer@nvidia.com
NVIDIA
New York City, New York, United States

Chris Deotte*
cdeotte@nvidia.com
NVIDIA
San Diego, California, United States

Jean-François Puget*
jpuget@nvidia.com
NVIDIA France Developpement
Saint Raphael, France

Gabriel de Souza Pereira Moreira*
gmoreira@nvidia.com
NVIDIA
São Paulo, Brazil

Gilberto Titericz*
gtitericz@nvidia.com
NVIDIA
Curitiba, Parana, Brazil

Jiwei Liu*
jiweil@nvidia.com
NVIDIA
Pittsburgh, Pennsylvania, United States

Ronay Ak*
ronaya@nvidia.com
NVIDIA
Sarasota, Florida, United States

ABSTRACT

In this paper we present our 1st place solution of the WSDM WebTour 21 Challenge. The competition task was to predict the city of the last booked hotel in a user trip, based on the previously visited cities. For our final solution, we designed three deep learning architectures based on Multilayer Perceptron (MLP), Gated Recurrent Units (GRU) and XLNet (Transformer) building blocks. A shared component among the architectures was a Session-based Matrix Factorization head (SMF), which learns a linear mapping between the item (city) embeddings and the session (trip) embeddings to generate recommendations by a dot product operation. Our final leaderboard result was 0.5939 for precision@4 and scored 2.8% better than the second solution. We published our implementation, using RAPIDS cuDF, TensorFlow and PyTorch, on github.com[1].

CCS CONCEPTS

• **Information systems** → **Personalization**; **Recommender systems**; *Learning to rank*.

KEYWORDS

Recommender Systems, WSDM WebTour 21 Challenge, Deep Learning, Sequential recommendation, MLP, GRU, Transformer, Matrix Factorization

1 INTRODUCTION

Recommender systems (RecSys) have become a key component in many online services, such as e-commerce, social media, news service, or online video streaming. Sequence-based recommendation is a highly relevant task in real-world applications, like predicting the next item in the online shopping cart, the next online course a student will choose, or the next travel destination of a traveller.

In this paper, we present our 1st place solution for the Booking.com WSDM WebTour 21 Challenge organized by Booking.com for the WebTour workshop at ACM WSDM 2021 [1].

The task was to recommend (predict) the final city destination for a traveller trip given their previous booking history within the trip. The competition evaluation metric was Precision@4, which scored true only if the correct next city was among the top-4 recommendations (regardless the order) out of the about 40,000 cities found in the dataset.

Previous recent RecSys competitions often require identifying the positive interactions given a set of positives and negatives (e.g. click-through rate (CTR) prediction), which somehow shifts the task from generating recommendations to a binary classification problem. In that setting, the focus is often on creating features to separate positive from negative examples rather than model design [4]. Boosted trees algorithms often performed well in such binary classification tasks, as observed in the last RecSys Challenge competitions [3] [5].

Differently from previous RecSys competitions, this dataset contained only positive examples and deep learning based solutions worked best for us, challenging the status-quo highlighted in [4].

For our final solution, we developed three different neural architectures: (1) MLP with Session-based Matrix Factorization net (MLP-SMF), (2) GRU with MultiStage Session-based Matrix Factorization net (GRU-MS-SMF), and (3) XLNet with Session-based Matrix Factorization net (XLNet-SMF). The final submission was a simple ensemble (weighted average) of the predictions from models of each of those three architectures.

2 PREPROCESSING AND FEATURE ENGINEERING

The competition dataset from Booking.com challenge has a tabular data structure, consisting of 1.5M of anonymized reservations of 269k trips. Each reservation is a part of a customer's trip (identified

*Authors contributed equally to this research.
[1]https://github.com/rapidsai/deeplearning/tree/main/WSDM2021

by utrip_id) which includes at least 4 consecutive reservations. A reservation contains the metadata and city/country information but does not include information about the hotel. Thus, the sequence of trip reservations can be obtained by sorting on the check-in date.

2.1 Data splitting for Cross-Validation

Unlike most competitions, this competition allowed only two submissions per team - the intermediate one and the final one. It was crucial to design a good the validation set, because it would be key to select the best features and models for the final submission. We used k-fold cross validation: predicting the last city of the trips on the train set split into 5-folds. For each fold, the fold train data was used as validation set, and data (both train and test) from the other folds (Out-Of-Fold - OOF) was used to train the model. We used the full OOF dataset, predicting the next city given previous ones. The cross-validation (CV) score was the average precision@4 from each of the five folds.

2.2 Feature Engineering

The original dataset has 9 features: user_id, utrip_id, checkin, checkout, city_id (label), hotel_country, device_class, affiliate_id and booker_country. For the last reservation of each trip of the test set, the city_id and hotel_country were not provided.

In our initial experiments, building new features improved the accuracy of our models. The main features included in our models are listed in Table 2 in Appendix A. All features that used information from other rows (e.g. user features, city features) were computed OOF, to avoid leakage from the fold labels. Depending on the architecture, a subset of the features were used.

We implemented our preprocessing and feature engineering pipeline with RAPIDS cuDF, a library for GPU-accelerated dataframe transformations.

2.3 Data augmentation with reversed trips

The dataset with 1.5M bookings is relatively small in comparison to other recommendation datasets. We explored techniques to increase the training data by data augmentation and discovered that doubling the dataset with reversed trips improved model's accuracy. A trip is an ordered sequence of cities and although there are many permutations to visit a set of cities, there is a logical ordering implied by distances between cities and available transportation connections. These characteristics are commutative. For example, a trip of Boston->New York->Princeton->Philadelphia->Washington DC can be booked in reverse order, but not many people would book in a random order like Boston->Princeton->Washington DC->New York->Philadelphia.

2.4 Low frequency cities hashing

The distribution of the cities reservation frequency was long-tailed as one would expect - some cities are much more popular for tourism and business than others. To help the models not to focus too much on very unpopular cities (city reservation frequency < 9 in our case), we assigned their city id to a single value for training and ensured that those cities were never ranked high during inference. Such approach reduced the cardinality of the cities from more than 44,000 to about 11,000 cities (see Figure 4 in Appendix B).

3 MODEL ARCHITECTURES

This section describes the final three neural architectures designed for this competition, which were ensembled for the final submission. Each architecture can be trained with a single NVIDIA V100 GPU.

As the cardinality of the label (city) is not large, all models treated the recommendation as a multi-class classification problem, by using softmax cross-entropy loss function. A common component of all architectures was the Session-Based Matrix Factorization (SMF) head, which is described in the next section.

3.1 Session-based Matrix Factorization head (SMF)

Framing this problem under the recommender systems taxonomy, the cities are the items we want to recommend for a trip. The trip is analogous to a session in the session-based recommendation task, which is generally a sequence of user interactions – hotel reservations in this case.

For this competition, it was designed a Session-based Matrix Factorization (SMF) head to produce the scores (logits) for all items, instead of a regular Multi-Layer Perceptron (MLP) layer. This design was inspired by the MF-based Collaborative Filtering, which learns latent factors for users and items by performing a dot product of their embeddings to predict the relevance of an item for an user.

The large majority of users had only a single trip available in the dataset. So, instead of trying to model the user preference, we used the last layer of the network to represent the session (trip) embedding. Then, we compute the dot product between the session embedding s and all the set I of item embeddings i, where $i \in I$, to model items relevance probability distribution r of each item (city) being the next for that session (trip), such as $r = \text{softmax}(s \cdot I)$.

3.2 MLP with Session-based Matrix Factorization head (MLP-SMF)

The MLP with Session-based Matrix Factorization head uses feed-forward and embedding layers, and can be seen in Figure 1. Categorical input features are fed through an embedding layer and continuous input features are individually projected via a linear layer to embeddings, followed by batch normalization and Rectified Linear Unit (ReLU) non-linear activation. All embedding dimensions are made equal. The embeddings of continuous input features and categorical input features, except the lag features, are combined via summation. The output is concatenated with the embeddings of the 5 last cities and countries (lag features).

The embedding tables for the city lags are shared, and similarly for hotel country lags. The lag embeddings are concatenated, but the model should still be able to learn the sequential patterns of cities by the order of lag features, i.e., city lag1's embedding vector is always in the same position of the concatenated vector.

The concatenated vector is fed through 3 feed-forward layers with batch normalization, Parametric Rectified Linear Unit (PReLU) activation function and dropout, to form the session (trip) embedding. It is used by the Session-based Matrix Factorization head to produce the scores for all cities. More details can be found in Appendix C.

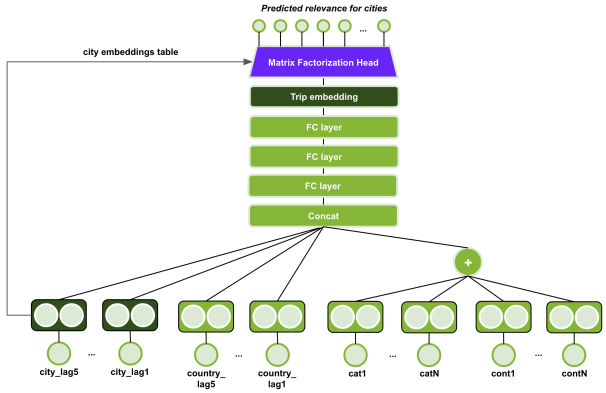

Figure 1: Visualization of MLP with Session-based Matrix Factorization head architecture.

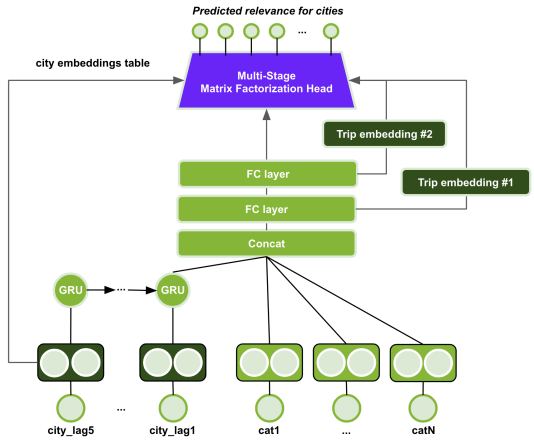

Figure 2: Visualization of GRU with MultiStage Session-based Matrix Factorization head architecture.

3.3 GRU with MultiStage Session-based Matrix Factorization head (GRU-MS-SMF)

The GRU with MultiStage Session-based Matrix Factorization head uses a GRU cell for embedding the historical user actions (previous 5 visited cities), similar to GRU4Rec [2]. Missing values (sequences with less than 5 length) are padded with 0s.

The last GRU hidden state is concatenated with the embeddings of the other categorical input features, as shown in Figure 2. The embedding tables for the previous cities are shared. The model uses only categorical input features, including some numerical features modeled as embeddings such as trip length and reservation order.

The concatenated embeddings are fed through a MultiStage Session-based Matrix Factorization head. In The first stage is a softmax head over all items, which is used to select the top-50 cities for the next stage. In the second stage, the Session-based Matrix Factorization head is applied using only the top-50 cities of the first stage and two representations of the trip embeddings (the outputs of the last and second-to-last MLP layers, after the concatenation), resulting in two additional heads. The final output is a weighted sum of all three heads, with trainable weights.

The multi-stage head works as a 2-stage ranking problem. The first head ranks all items and the other two heads can focus on the reranking of the top-50 items from the first stage. This approach can potentially scale to large item catalogs, i.e., in the order of millions. This dataset did not require such scalability, but this multi-stage design might be effective for deployment in production. Training details can be found in Appendix C.

3.4 XLNet with Session-based Matrix Factorization head (XLNet-SMF)

The XLNet with Session-based Matrix Factorization head (XLNet-SMF) uses a Transformer architecture named XLNet [7], originally proposed for the permutation-based language modeling task in Natural Language Processing (NLP). In our case, instead of modeling the sequence of word tokens, we model the sequence of items in the session (trip).

We have adapted the XLNet training task for Masked Language Modeling (also known as Cloze task), like proposed by BERT4Rec [6] for sequential recommendation. In that approach, for each training step, a proportion of all items are masked from the input sequence (i.e., replaced with a single trainable embedding), and then the original ids of the masked items are predicted using other items of the sequence, from both left and right sides. When a masked item is not the last one, this approach allows the usage of privileged information of the future reservations in the trip during training. Therefore, during inference, only the last item of the sequence is masked, to match the sequential recommendation task and do not leak future information of the trip [6].

For this network, each example is a trip, represented as a sequence of its reservations. The reservation embedding is generated by concatenating the features and projecting using MLP layers.

The sequence of reservation embeddings is fed to the XLNet Transformer stacked blocks, in which the output of each block is the input of the next block. The final transformer block output generates the trip embedding.

Finally, we use the Matrix Factorization head (dot product between the trip embedding and city embeddings) to produce a probability distribution over the cities for the masked items in the sequence. Model hyperparameters can be found in Appendix C.

4 ENSEMBLING

Ensembling is a proven approach to improve the accuracy of models, by combining their predictions and improving generalization.

We used the k-fold cross-validation and bagging (training the model many times with different random seeds in our case) techniques to ensemble models from our three architectures, as shown in Algorithm 1 in Appendix D.

In general, the higher the diversity of model's predictions, the more the ensembling technique can potentially improve the final scores. In our case, the correlation of the predicted city scores between each two combinations of the three architectures was around 80%, which resulted in a representative improvement with ensembling in our final CV score.

In addition to ensembling the three models by averaging the predictions, we also tried to add another ensembling layer: a model

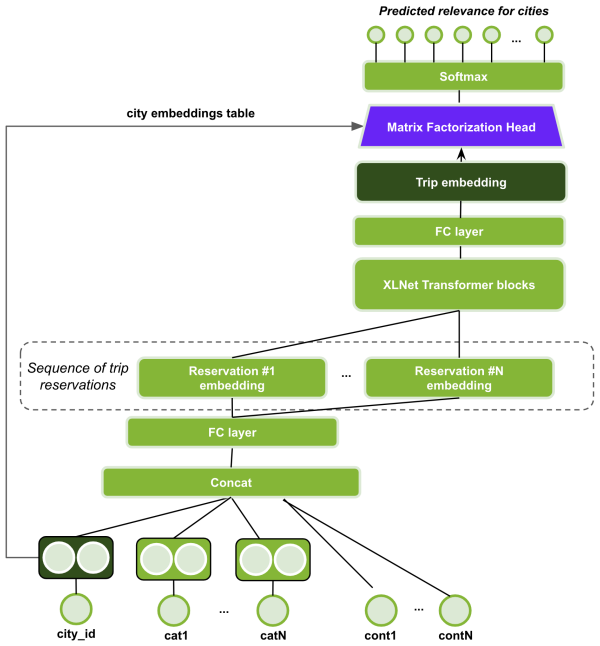

Figure 3: Visualization of XLNet with Session-based Matrix Factorization head architecture.

to rerank the top 20 cities from the averaged predictions. That insight came from the fact that the precision@20 of our averaged predictions was 0.777, a much higher score than for the competition metric precision@4 (around 0.57) (see Figure 5 in Appendix E). The reranker was a gradient boosted decision trees model for binary classification of cities (0=incorrect, 1=correct last city of the trip), having as input features the predictions from the averaged predictions and additional features like the frequency of the city.

Our initial experiments showed promising results, with improved precision@4 by 0.5%. Due to the lack of a public leaderboard and only one final submission, we decided for the conservative ensembling approach of weighted averaged predictions to rank the cities for submission, as described in Algorithm 1 in Appendix D.

5 RESULTS AND DISCUSSION

To make it clear the contribution from each architecture and from the ensembling algorithm for our final result, we present in Table 1 the cross-validation Precision@4 by architecture, individual and after bagging ensembling, and the final ensemble from the three architectures combined. We can notice that the XLNet-SMF was the most accurate single model.

Interestingly, the MLP-SMF architecture achieved a CV score comparable with the other two architectures that explicitly model the sequences using GRU and XLNet (Transformer). In MLP-SMF the embeddings of the last 5 cities are simply concatenated and combined by MLP layers, while for the GRU-MS-SMF the last 5 cities are processed by a GRU layer, and for XLNet-SMF the last 15 cities are processed by the Transformer. As the MLP-SMF model was lightweight, it was much faster to train than the other architectures, which sped up its experimentation cycle and improvements.

	Single bag CV	Ensemble CV	Final LB
MLP-SMF	0.5667	0.5756	
GRU-MS-SMF	0.5664	0.5762	
XLNet-SMF	**0.5681**	0.5751	
Final Ensemble		**0.5825**	**0.5939**

Table 1: Final results by architecture and for the final ensemble - MLP-SMF uses 8 bags, GRU-MS-SMF uses 7 bags and XLNet-SMF uses 5 bags.

As all three neural network architectures use the Session-based Matrix Factorization head and the data augmentation by reversing trips, we wanted to understand the contribution of each component by a small ablation study. Table 3 in Appendix F summarizes the results training the MLP architecture with and without the Session-based Matrix Factorization head (MLP-SMF). We found out that the SMF head improved the baseline by 2.1%.

Analogously, we trained the MLP-SMF model without and with data augmentation and noticed an improvement of 0.8% (see Table 4 in Appendix F).

6 CONCLUSION

In this paper we presented our 1st place solution of the Booking.com WSDM WebTour 21 Challenge. We designed three different deep learning architectures based on MLP, GRU and Transformer neural building blocks. Some techniques resulted in improved performance for all models, like our proposed Session-based Matrix Factorization head and the data augmentation with reversed trips. The diversity of our architectures resulted in significant accuracy improvements by ensembling model predictions.

We hope our final solution, which is provided on github.com[2], is useful to others in the RecSys field who build sequential recommendation systems.

ACKNOWLEDGMENTS

The authors wish to thank our colleagues on the NVIDIA Merlin, KGMON and RAPIDS.AI teams for their support in this competition.

REFERENCES

[1] 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)*.

[2] Balázs Hidasi, Alexandros Karatzoglou, L. Baltrunas, and D. Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2016).

[3] Pawe Jankiewicz, Liudmyla Kyrashchuk, Pawe Sienkowski, and Magdalena Wójcik. 2019. Boosting Algorithms for a Session-Based, Context-Aware Recommender System in an Online Travel Domain. In *Proceedings of the Workshop on ACM Recommender Systems Challenge*.

[4] Dietmar Jannach, Gabriel de Souza P. Moreira, and Even Oldridge. 2020. Why Are Deep Learning Models Not Consistently Winning Recommender Systems Competitions Yet? A Position Paper. In *Proceedings of the Recommender Systems Challenge 2020* (Virtual Event, Brazil) *(RecSysChallenge '20)*. Association for Computing Machinery, New York, NY, USA, 44–49. https://doi.org/10.1145/3415959.3416001

[5] Benedikt Schifferer, Gilberto Titericz, Chris Deotte, Christof Henkel, Kazuki Onodera, Jiwei Liu, Bojan Tunguz, Even Oldridge, Gabriel De Souza Pereira Moreira, and Ahmet Erdem. 2020. GPU Accelerated Feature Engineering and Training for Recommender Systems. In *Proceedings of the Recommender Systems Challenge*

[2] https://github.com/rapidsai/deeplearning/tree/main/WSDM2021

2020 (Virtual Event, Brazil) *(RecSysChallenge '20)*. Association for Computing Machinery, New York, NY, USA, 16–23. https://doi.org/10.1145/3415959.3415996

[6] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. *BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer*. Association for Computing Machinery, New York, NY, USA, 1441–1450. https://doi.org/10.1145/3357384.3357895

[7] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc Le. 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding.

A FEATURES

Feature Set	Features
original categorical features	city_id, hotel_country, booker_country device_class, affiliate_id
check-in and check-out date features	day-of-week, week-of-year, month, isweekend, season, stay length (checkout - checkin), days since last booking (checkin - previous checkout)
sequence features	first city in the trip, lagged (previous 5) cities and countries from the trip
current trip stats	trip length (# reservations), trip duration (days), reservation order (in ascending and descending orders)
past user trips stats	# user's past reservations, # user's past trips, # user's past visited cities, # user's past visited countries
geographic-seasonal city popularity	Features based on the conditional probabilities of a city c from a country co, being visited at a month m or at a week-of-year w, as follows: P(c | m), P(c | m,co), P(c | w), P(c | w,co)

Table 2: Features used by at least one of the architectures.

B DISTRIBUTION FREQUENCY CITIES

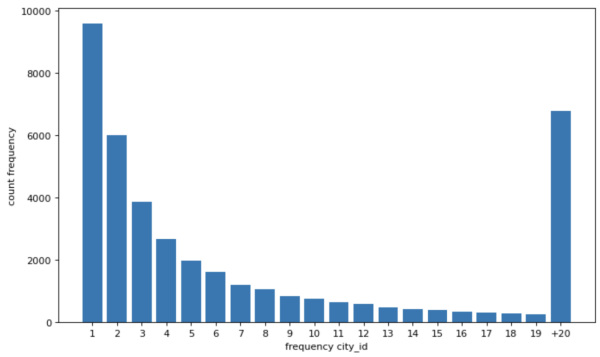

Figure 4: Distribution of frequency of city_id in train and test dataset. Around 10,000 city ids appeared only once in the dataset.

C TRAINING DETAILS

MLP-MF: In this architecture, embedding tables were initialized with $N \sim (0, 0.01)$. Low frequent cities are ignored by Categorical Cross Entropy loss. The model is trained with Adam optimizer for 12 epochs with a batch size of 1024, with a 1 cycle learning rate scheduler.

GRU-MS-SMF: The model is trained with the Adam optimizer for 5 epochs, with a batch size of 512, and a learning rate step decay schedule (1e-3, 1e-3, 1e-4, 1e-5, 1e-6). A variation of bagging is applied by training the architecture 7 times with different seeds and a slight variation of used input features. The final predictions are generated by averaging the probabilities over all 7 models.

XLNet-SMF: It was stacked 4 XLNet Transformer blocks, with 4 heads each, with dimension 512. The model was trained for 30 epochs, with initial learning rate 2e-4 and linear decay. The batch size was 128, where each training instance is a session (trip) truncated to maximum length 15. For regularization, it was used dropout 0.1 and AdamW optimizer with weight decay of 1e-5. The masking probability during training was 0.1, but ensuring that we always have at least one masked item for each session (trip).

D ENSEMBLING ALGORITHM

Algorithm 1: Ensembling algorithm

```
split data in k-folds;
for each architecture do
    j = number of bags for this architecture;
    for each fold in folds do
        for i in [1, ..., j] do
            concatenate train and test out-of-fold data
              (OOF);
            train model on OOF;
            evaluate model based on last city of in-fold train
              data;
            predict last city of all folds of test data;
        end
    end
    ensemble by averaging over each fold and each bag;
end
ensemble by averaging over architectures;
```

E PRECISION@K

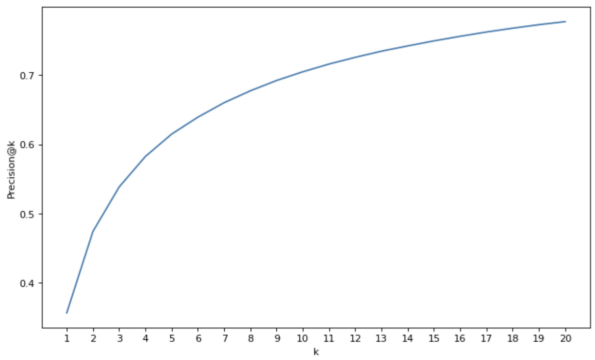

Figure 5: Precision@k for top20 recommendations by ensemble model.

F ABLATION STUDY

	1 model CV
MLP	0.5550
MLP-SMF	0.5667 (+2.1%)

Table 3: Cross validation score for training MLP architecture with and without the Session-based Matrix Factorization head (MLP-SMF).

	1 model CV
MLP-SMF without data augmentation	0.5620
MLP-SMF	0.5667 (+0.8%)

Table 4: Cross validation score for training MLP-SMF with and without data augmentation.