A-HA: A Hybrid Approach for Hotel Recommendation

Kung-Hsiang, Huang
Rosetta.ai
steeve@rosetta.ai

Yi-Fu, Fu*
National Taiwan University
yifu.arlj@gmail.com

Yi-Ting, Lee*
National Taiwan University
amy19412000@gmail.com

Tzong-Hann, Lee
National Taiwan University
maxware0715@gmail.com

Yao-Chun, Chan
National Taiwan University
jan11781202@gmail.com

Yi-Hui, Lee
Univeristy of Texas, Dallas
yi-hui.lee@utdallas.edu

Shou-De, Lin
National Taiwan University
sdlin@csie.ntu.edu.tw

ABSTRACT

Session-based recommender system refers to a specific type of recommender system that focuses more on the transactional structure of each session rather than the user and item interactions [16]. It is stated that the users' interactions are mostly homogeneous in the same sessions, while being heterogeneous across different sessions [5]. Therefore, it is essential to extract the interest dynamics of users within each session. The 2019 ACM Recsys Challenge [10] aims to apply session-based recommender systems to the domain of travel metasearch. The goal is to predict which hotels are clicked in the search results based on the context of each session. In this paper, we propose our approach to effectively tackle the challenge. It involves an ensemble of three models, LightGBM, XGBoost, and a Neural Network based on DeepFM [6] that is capable of handling sequential features. Our team, RosettaAI, won the 4th place in this challenge, scoring 0.679933 on the final leaderboard. The source code is available online [1].

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Hotel recommendation, Gradient boosting machine, Neural networks

ACM Reference Format:
Kung-Hsiang, Huang, Yi-Fu, Fu, Yi-Ting, Lee, Tzong-Hann, Lee, Yao-Chun, Chan, Yi-Hui, Lee, and Shou-De, Lin. 2019. A-HA: A Hybrid Approach for Hotel Recommendation . In *Proceedings of the ACM Recommender Systems Challenge 2019 Workshop (RecSys Challenge '19), September 20, 2019, Copenhagen, Denmark*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3359555.3359560

*Both authors contributed equally to the paper
[1]https://github.com/rosetta-ai/rosetta_recsys2019

RecSys Challenge '19, September 20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-7667-9/19/09…$15.00
https://doi.org/10.1145/3359555.3359560

1 INTRODUCTION

With the ever-growing number of available accommodations online, successfully matching hotels with the user's interest becomes gradually important. Since the majority of users visiting these sites are not logged-in, in addition to considering the users' past behaviors, an effectual recommender system must be able to model the transition of user interests in the sessions. Therefore, to tackle the challenge, we need to build a session-based and context-aware recommender system tailored for hotel recommendation.

Several studies [14] [7] have attempted to solve this task with Collaborative Filtering (CF) [13]. Essentially, this algorithm constructs a user-item matrix based on the past interactions of all users. Predictions are made by computing the similarity between items or users. Although CF successfully extracts the latent patterns behind user and item interactions, it suffers from two main drawbacks. First, it ignores the contextual feature of each session. Such information is critical for modeling the dynamics of user interest throughout the session. In addition, the performance of CF decreases significantly when the sparsity of the user-item matrix is high. This holds in this challenge. The user-item matrix is highly sparse due to the large portion of non-logged-in users.

Our team leveraged an ensemble of three different models, Neural Network, LightGBM, and XGBoost to solve the challenge. We utilized the Neural Network's expressive ability in modeling sequential data with Bi-directional Gated Recurrent Units (Bi-GRUs) [3], while using the other two models to learn from structural data efficiently. By ensembling these three models, we can take advantage of the strengths of each of them.

In this study, we discovered that although unable to process sequential features, LightGBM and XGBoost still outperform Neural Network significantly thanks to their efficacy in extracting information from tabular data. Furthermore, session-based features, especially those related to the immediate last interaction, are found to be the most important features in model's performance.

The rest of the paper is organized as follows. First, the framework of the challenge will be described. Then, we will explain our approach to the challenge, including the loss function, the models,

the ensemble structure, and the feature engineering. Finally, the experiments and results will be illustrated.

2 CHALLENGE TASK

2.1 Problem Definition

In the 2019 ACM Recsys Challenge, trivago provided a dataset of browse logs on the Trivago website, which consists of consecutive actions of users and referenced items (e.g. hotel). There are various action types, including *interaction with item image*, *change of sort order, filter selection*, *search for destination*, *clickout item*, and etc. The goal of the competition is to predict the exact clicked item among at most 25 candidates in the rows associated with the *clickout item* action, given the information of series of actions right before the *clickout item* action occurs. The evaluation metric is Mean Reciprocal Rank (MRR), defined as follows [12]:

$$MRR = \frac{1}{|Q|}\sum_{i=0}^{|Q|}\frac{1}{rank_i} \tag{1}$$

where Q denotes the total number of samples, and $rank_i$ denotes the rank of the first correct answer for sample i.

2.2 Data Description

As stated in the Challenge Dataset webpage, the provided dataset contains training data, test data, and metadata of accommodations (items). In particular, there are 730803 users, 400277 items, and 910683 independent sessions, each of which is composed of series of actions involving one user and several items. There may exist one or more *clickout item* actions in a single session. In a *clickout item* action, all possible accommodations and their prices are listed in the same order as they were displayed to the user, also known as the *impressions list* and *price list*.

3 APPROACH

In this section, we will describe our approach to the challenge. We first define the loss function as well as the data preprocessing pipelines. Then, two different types of models we implemented, Neural Network, and Gradient Boosting Machine are illustrated. Lastly, we demonstrates the important features we engineered.

3.1 Loss Function

Inspired by this study [4], we adopted binary cross-entropy (BCE) loss as our loss function. Each of the item in the impression is broken down into individual samples. The labels associated with the clicked item are 1, while the others being 0. Mathematically, the loss function is defined as follows:

$$L = \sum_{i=0}^{N} y_i log(\hat{y}_i) + (1-y_i) log((1-\hat{y}_i)) \tag{2}$$

where N denotes the total number of broken down samples, $y_i \in [0,1]$ denotes the ground truth, and $\hat{y}_i$ denotes the output of the model, where $\{\hat{y}_i \in \mathbb{R}; 0 \leq \hat{y}_i \leq 1\}$.

3.2 Preprocessing

The preprocessing procedure involves three steps, removing invalid *clickout item* rows, breaking down impression, and encoding categorical feature. In the first step, a *clickout item* row is considered invalid if its reference value does not present in its impression. Such rows containing no positive ground truth are removed from the training and validation sets. Then, as aforementioned, the impression of each row is broken down to $\mathcal{I}_i$ samples, where $\mathcal{I}_i$ is the number of items in the i-th impression. Categorical features are encoded from 0 to $C_j - 1$, where C_j denotes the number of unique value for the j-th categorical feature. In addition to saving memory, such encoding was chosen over one-hot encoding for later feeding the encoded features into embedding layers efficiently.

3.3 Neural Network

Motivated by DeepFM [6], we propose a Neural Network that is capable of modeling the second-order feature interaction efficiently, while taking into account the temporal dynamics of user-item interactions at the same time. As depicted in Figure 1, the input features can be divided into three parts, continuous features, categorical features, and sequential categorical features. The following sub sections will illustrate the detail of each of them with the notation below.

- D: 1-dimensional continuous features. D_i denotes the i-th continuous feature.
- E: 1-dimensional encoded categorical features. E_i denotes the i-th categorical feature.
- F: 2-dimensional encoded sequential categorical features. F_i denotes the i-th sequential categorical feature.

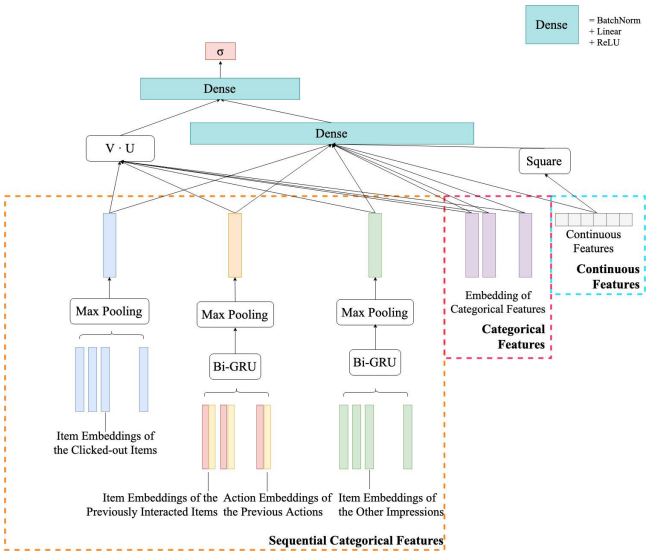

Figure 1: The model architecture of the proposed Neural Network. The inputs constitutes of three parts, continuous features, categorical features, and sequential categorical features. All layers are densely connected.

3.3.1 Continuous Features. Continuous features must be properly normalized for Neural Networks to learn well. Two different normalization techniques were applied to D, uniform normalization to [0, 1] and the Rank Gauss transformation [15]. The latter one ranks the continuous values for each D_i, uniformly maps the ranking onto [0, 1], and transforms this with the inverse error function. The error function, which is also known as the Gauss error function, is mathematically defined as [1]:

$$erf(x) = \int_0^x e^{-t^2} dt \tag{3}$$

The two methods work the best compared with standardization. Thus, both of them are used for normalization.

3.3.2 Categorical Features & Sequential Categorical Features. Both E and F are embedded to obtain a u-dimensional representation for each feature, where u $\in \mathbb{N}$. Notably, features under the same ID space share the same embedding weights. For instance, there exists a single embedding of item IDs that distinct features from E and F may contain (item IDs of the impression and the IDs of the items that the user had interacted in the current session).

To model the dynamics of user interest throughout the session, the embeddings of item IDs and actions are concatenated, and processed by a Bi-directional GRU. The outputs of GRU for each time step are then aggregated with max pooling. Formally, it can be described as:

$$G = g([F_{item}, F_{action}], \theta)$$
$$V = maxpool(G)$$

where θ denotes the parameters of Bi-GRU, g denotes the Bi-GRU transformation, G denotes the output of Bi-GRU, and V denotes the aggregated vector. Similarly, the other features in F boils down to u-dimensional vectors.

3.3.3 Feature Interaction. As stated in [6], feature interaction allows the model to discover the implicit meaning behind user's behavior. Therefore, a proper feature interaction mechanism is important for improving the performance of the model. DeepFM computes the Hadamard products of the embedding vectors of each pair of features, and aggregating them with weight-1 connections. Essentially, it is equivalent to summing the element-wise products of the embeddings. This can be achieved efficiently by computing difference between the square of sum and the sum of square of these embedding vectors. Let $V_i, V_j, V_k \in \mathbb{R}^u$ denotes the embedding vectors for feature i, j, and k, we can derive the following:

$$(V_i + V_j + V_k)^2 = V_i^2 + V_j^2 + V_k^2 + 2(V_i \circ V_j + V_j \circ V_k + V_i \circ V_k)$$
$$(V_i \circ V_j + V_j \circ V_k + V_i \circ V_k) = \frac{1}{2}((V_i + V_j + V_k)^2 - (V_i^2 + V_j^2 + V_k^2))$$

where $\circ$ denotes the Hadamard product operation.

Hence, it is not necessary to explicitly compute the Hadamard products for each pair of embedding vectors; instead, we simply compute the two terms on the right-hand-side in the above equation.

3.4 Gradient Boosting Machine

Tree-based models usually perform the best in structured data, especially for gradient boosting machines. These models have been dominant in Kaggle competitions involving tabular data. We selected LightGBM [8] and XGBoost [2] to increase model diversity for ensemble. These two models takes all the features fed to the Neural Network, except for the sequential categorical features, with additional hand-crafted features. It turns out that both of these two models outperform the Neural Network by a large margin. Details of the experiment results will be provided in the latter section.

3.5 Ensemble

Our ensemble method is an weighted average of the predictions for each of the three models discussed above. Specifically, the predictions are blended with the following ratio:

$$NeuralNetwork : LightGBM : XGBoost = 1 : 7 : 4$$

Empirically, this ratio generates the best results.

3.6 Feature Engineering

We conducted feature engineering with the aim of capturing all different aspects. The different set of features are described by the following subsections:

3.6.1 Impression-based features. There are some obvious information of the impression list including price, city, platform, rating, and star of those items on impression list. In addition, we engineered some hidden messages such as the rank of each item's price, the position of the item on the impression list, and the length of the impression list.

3.6.2 Session-based features.

- Time difference feature: The time difference between current and the last step action.
- Item time difference: The time difference between when an item was last interacted and when the clickout item action took place.
- Last interacted item impression index: The position of the last interacted item in the current impression list.
- Equal last interacted item: Whether this item in impression list equal to the last interacted item in the current session.
- Local count feature: For each section, we compute the count of different types of interaction for each item in the impression list.
- Last interact index: The impression index of the the last interacted item in the current session.
- Predicted next impression index: To model the eye movement of the user on the website, we leveraged the position of the last interacted item and the second last interacted item on the current impression list. Specifically, this feature is computed as follows: `Predicted next impression index = last impression index + ( last impression index - second last impression index )`, where the second term calculates the movement of the last interaction.

3.6.3 Aggregation features.

- Target encoding: To better approximate the priors, we aggregate the mean value for some of the categorical features, such as price rank and impression index.
- City price bin: Considering the different price indices of different cities, we categories prices into bins based on city.

4 EXPERIMENTS

4.1 Experimental Settings

To prevent the models from seeing the future clickout events, which might lead to label leakage, we adopted the *leave-one-out* evaluation as our validation strategy. In specific, 50000 sessions were randomly selected, only the last clickout events of which are used as validation data. The validation scores shown in this paper refer to the models' performance on this validation set.

4.2 Training Process

The Neural Network was implemented in PyTorch [11]. Adam [9] was used for optimizing the Neural Network with learning rate=0.001 and weight decay=0. To boost the training speed, the batch size was set to 1024. We train the Neural Network only for 1 epoch on a single Nvidia Tesla M60 GPU. In our experiments, the Neural Network usually starts to overfit when trained beyond 1 epoch. Any regularization technique other than batch normalization, such as dropout and l2-regularization, only result in poorer performance. All the weights are randomly initialized, following the default setting of PyTorch.

As for LightGBM and XGBoost, the learning rate was set to 0.01 and 0.02, respectively. The other training parameters are in general the same. Their maximum number of boosting rounds and early stopping rounds are set to 50000 and 500. BCE loss is used as a proxy for their early stopping criteria since computing MRR for every round during training is computationally expensive. Their base learners are gradient boosting decision trees, which perform the best empirically. They are all trained on 48 CPU cores.

4.3 Experimental Results

The computation time and performance of the three models are evaluated. As demonstrated in Table 1, the Neural Network takes the least time to train as it converge the fastest. LightGBM requires the longest training time since it stopped at its maximum boosting rounds, while XGBoost stopped at less than 20000 rounds. The long training time for LightGBM also explains its long inference time. The longer the model is trained, the more sophisticated the model; thus, taking more time while performing prediction.

Comparing the three single models, tree-based models outperform the Neural Network by a large margin, as shown in Table 2, even though sequential features are not fed into these models. We hypothesize that this is because tree-based models are efficient in extracting information from tabular data. In addition, such data does not exist evident hierarchical structure compared with image or text data, where Neural Networks dominates with their strong hierarchical representation abilities. Among these three, LightGBM performs the best in local validation. However, the best LightGBM prediction was not submitted since it was produced almost at the end of the challenge when we do not have many submission opportunities. That prediction result was only used for ensemble. By blending the prediction of three models with the ratio mentioned previously, the ensemble achieved the best MRR on the public Leaderboard.

	NN	LightGBM	XGBoost
Training Time	**6**	14	10
Inference Time	**0.3**	1.2	1

Table 1: Computation time comparison. (Hours)

	NN	LightGBM	XGBoost	Ensemble
Validation	0.675206	**0.685787**	0.684521	N/A
Leaderboard	0.672117	N/A	0.681128	**0.682128**

Table 2: Performance comparison. (MRR)

Feature Name	Importance Score
Equal Last Interacted Item	8.9×10^7
Item Time Difference	3.9×10^7
Impression Index	3.2×10^7
Last Interact Index	2.3×10^7
Time Difference	1.1×10^7

Table 3: Top 5 important features for LightGBM

Table 3 shows the top 5 important features generated by our best LightGBM model. It seems that session-based features related to the immediate last action before the *clickout item* actions play the most important roles in improving our modes. For instance, *Equal Last Interacted Item* indicates if an item in the impression list is the same as the immediate last interacted item, while *Item Time Difference* refers to the time difference between when an item was last interacted and when the *clickout item* action took place.

5 CONCLUSION

In this paper, we described our approach to the 2019 ACM RecSys Challenge. The challenge was formulated as a binary classification problem where BCE loss was adopted to optimize the models. An ensemble of three models were presented to tackle the challenge, including Neural Network, LightGBM and XGBoost. We discovered that tree-based models are more effective in extracting information from tabular data than Neural Networks in this challenge. Moreover, among all the features, session-based features associated with the immediate last interaction are the most critical in terms of improving the evaluation metric. In the end, our team won the fourth place on the final leaderboard of this challenge, suggesting that our proposed ensemble model is effective in solving such task.

REFERENCES

[1] Larry C Andrews and Larry C Andrews. 1992. *Special functions of mathematics for engineers*. McGraw-Hill New York.
[2] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785
[3] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling.

CoRR abs/1412.3555 (2014). arXiv:1412.3555 http://arxiv.org/abs/1412.3555

[4] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. ACM, 191–198.

[5] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep Session Interest Network for Click-Through Rate Prediction. *arXiv preprint arXiv:1905.06482* (2019).

[6] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint arXiv:1703.04247* (2017).

[7] Ya-Han Hu, Pei-Ju Lee, Kuanchin Chen, J Michael Tarn, and Duyen-Vi Dang. 2016. Hotel Recommendation System based on Review and Context Information: a Collaborative Filtering Appro.. In *PACIS*. 221.

[8] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 3146–3154. http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf

[9] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[10] Peter Knees, Yashar Deldjoo, Farshad Bakhshandegan Moghaddam, Jens Adamczak, Gerard-Paul Leyson, and Philipp Monreal. 2019. RecSys Challenge 2019: Session-based Hotel Recommendations. In *Proceedings of the Thirteenth ACM Conference on Recommender Systems (RecSys '19)*. ACM, New York, NY, USA, 2. https://doi.org/10.1145/3298689.3346974

[11] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[12] Dragomir R Radev, Hong Qi, Harris Wu, and Weiguo Fan. 2002. Evaluating Web-based Question Answering Systems.. In *LREC*.

[13] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, John Riedl, et al. 2001. Item-based collaborative filtering recommendation algorithms. *WWW* 1 (2001), 285–295.

[14] Qusai Shambour, Mouath Hourani, and Salam Fraihat. 2016. An item-based multi-criteria collaborative filtering algorithm for personalized recommender systems. *International Journal of Advanced Computer Science and Applications* 7, 8 (2016), 274–279.

[15] Tri Duc Nguyen Tang. 2019. Knowledge Distillation with NN + RankGauss. Retrieved June 5, 2019 from https://www.kaggle.com/mathormad/knowledge-distillation-with-nn-rankgauss/data

[16] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).