Session-aware Item-combination Recommendation with Transformer Network

Tzu-Heng Lin
Xiaomi AI Lab
Xiaomi Inc.
lzhbrian@gmail.com

Chen Gao
Department of Electronic Engineering
Tsinghua University
chgao96@gmail.com

***Abstract*—In this paper, we detailedly describe our solution for the IEEE BigData Cup 2021: RL-based RecSys (Track 1: Item Combination Prediction)[1]. We first conduct an exploratory data analysis on the dataset and then utilize the findings to design our framework. Specifically, we use a two-headed transformer-based network to predict user feedback and unlocked sessions, along with the proposed session-aware reweighted loss, multi-tasking with click behavior prediction, and randomness-in-session augmentation. In the final private leaderboard on Kaggle, our method ranked 2nd with a categorization accuracy of 0.39224.[2]**

***Index Terms*—recommender system, item combination prediction, transformer, loss reweighting**

I. Introduction

The task of the IEEE BigData Cup 2021: RL-based RecSys (Track 1: Item Combination Prediction) [1], [2] is to predict each user's purchasing feedback to nine exposed items, given this user's click history, portrait features, and items' features, which is similar to *bundle recommendation* [3]. The special setting in this task is that the nine items are grouped into three sessions. The user can only unlock the subsequent session after he/she buys all three items in the current session.

More formally, given a user u (along with his/her clicking history $c_{u,1}, c_{u,2}, ...,$ and some portrait features $f_{u,1}, f_{u,2}, ..., f_{u,10}$), and his/her nine exposed items $i_{u,1}, i_{u,2}, ..., i_{u,9}$ (along with some item features $f_{i,1}, f_{i,2}, ..., f_{i,6}$ for each item i), the objective is to predict nine interactions $y_{u,1}, y_{u,2}, ..., y_{u,9} \in \{0, 1\}$. Each one of the interactions indicates whether this user would buy the corresponding item or not. In addition, in this scenario, the middle three items $i_{u,4}, i_{u,5}, i_{u,6}$ are not unlocked until the user has bought all of the first three items $i_{u,1}, i_{u,2}, i_{u,3}$, and similarly, the last three items $i_{u,7}, i_{u,8}, i_{u,9}$ are not unlocked until the user has bought all of the first six items $i_{u,1}, i_{u,2}, ..., i_{u,6}$ (*c.f.* Figure 1). The evaluation metric for this task is the Categorization Accuracy measure, which is defined as follows,

$$\textbf{accuracy} = \frac{1}{M} \sum_{u=1}^{M} \prod_{j=1}^{9} [y_{u,j} = \hat{y}_{u,j}], \tag{1}$$

[1]https://www.kaggle.com/c/bigdata2021-rl-recsys/
[2]Our code is available at https://github.com/lzhbrian/bigdatacup2021

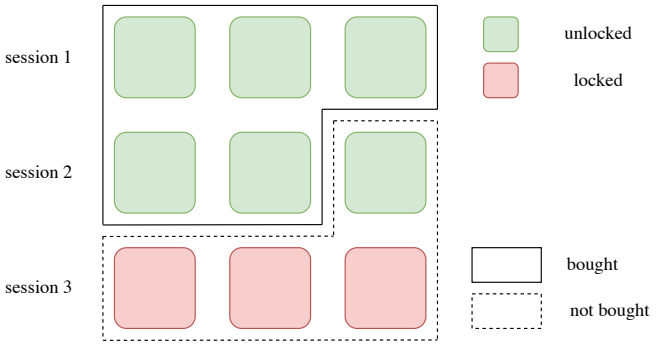


Fig. 1: Problem Setup. Each user is exposed to nine items simultaneously. However, the items are divided into three 3-length sessions. The user can only unlock the subsequent three items after he/she buys all three items in the current session. We want to predict whether a user would buy the nine exposed items or not.

where M denotes the number of users, $y_{u,j}$ and $\hat{y}_{u,j}$ are the predicted and ground-truth interactions, and $[y_{u,j} = \hat{y}_{u,j}]$ is the *Iverson bracket*.

Overall speaking, this task is challenging in two aspects.

- Firstly, the nine exposed items are correlated and treated differently by the users. We cannot simply apply a single traditional recommendation method to predict each interaction independently.
- Secondly, with the given evaluation metric, it is required to correctly predict all of the nine interactions of a user, while partially correct predictions contribute nothing to the final score.

To overcome the above challenges, we propose a delicate two-headed transformer-based framework to predict both users' buying behavior and unlocked sessions. The unlocked session prediction can be used to refine unreasonable buy predictions. We further propose a randomness-in-session augmentation technique and a novel session-aware reweighted loss to address the unique characteristics in this scenario. Finally, a multi-tasking training procedure with click prediction is utilized to assist the learning of embedding layers. Extensive experiments and ablation studies have demonstrated the effec-

978-1-6654-3902-2/21/$31.00 © 2021 IEEE

TABLE I: Overall data statistics

# buying entries (users)		# clicks		# items
# train	# test	# train	# test	
260,087	206,254	10,435,798	8,357,719	381

tiveness of our method.

In what follows, we will discuss related works in Section II, conduct an exploratory data analysis in Section III, describe our proposed method in Section IV, and finally conclude the paper with discussion and future works in Section V.

II. Related work

Recommender systems aim to filter information for users, which has become one kind of fundamental service in today's information platforms [4]. Generally, from the perspective of real-world application, the recommender systems contain two stages, matching and ranking. Recently, deep learning has become the state-of-the-art solution of recommender systems in both two stages [5]–[7]. As for the matching stage, of which the mainstream methods are collaborative filtering [8], which learns user interests from historical behaviors, deep neural networks methods [9], or even graph neural networks [10], [11], achieve promising performance. As for the ranking stage, which is also known as click-through rate (CTR) prediction, deep learning-based models such as DeepFM with multi-layer perceptron [12], xDeepFM with compressed interaction network [13], DIN [14] with attention mechanisms, etc., are demonstrated effective in learning from complex features of users and items.

In this work, we develop a method based on transformer network, a recent advance of neural network with extraordinary achievements in many areas, for capturing the complex behavior of users in the task of item combination recommendation.

III. Exploratory Data Analysis

Before diving into the model design, we conduct exploratory data analysis firsthand to master the whole picture of the dataset.

A. Data statistics

Table I shows the overall statistics of this dataset. In total, there are 381 items. There are 260,087 buying entries for training and 206,254 buying entries for testing. These entries are also accompanied by 10,435,798 and 8,357,719 clicking logs, respectively. We will then analyze more details about the clicking and buying behavior of users in the following.

Table II shows how many clicks and buys do items in each session possess. It's worth noticing that an item would only appear in its specific session. We can see that items in later sessions are with more types, and items with earlier sessions possess more clicks and buys. This is reasonable since users need to buy early items in order to unlock items (with higher prices) in the later sessions.

TABLE II: Click and buy statistics in different sessions.

session	item IDs	# items	# clicks	# buys
1	1∼39	39	4,606,977	616,952
2	40∼147	108	3,608,173	485,449
3	148∼381	234	2,220,648	287,482

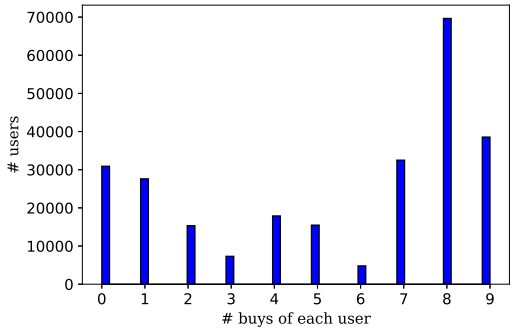


Fig. 2: Histogram of the number of buys of each user.

B. Buying behavior analysis

Due to the dataset characteristics (*c.f.* Section I), we plot the histogram of the number of items each user bought in Fig. 2, and classify users into four groups according to the number of items they have bought as follows,

- Group-0: 30,912 users who have bought 0 item.
- Group-1: 50,267 users who have bought 1∼3 items.
- Group-2: 38,191 users who have bought 4∼6 items.
- Group-3: 140,717 users who have bought 7∼9 items.

We can see that a decent population (Group-0) didn't buy anything, the number of users who bought 4∼6 items (Group-2) are the fewest, and a large portion of users (Group-3) chose to buy no less than seven items. This indicates an **hourglass shape of user distribution**. It's also worth noticing that very few people buy three or six items (*c.f.* Fig. 2). We hypothesize that this is because the main reason why a user buys three or six items is to unlock and buy items in the next session.

C. Clicking behavior analysis

We plot the histogram of the number of clicks of each user in Fig. 3. There are 28,184 users who did not click anything. However, we do see that the majority of users are with a decent number of clicks, which motivates us to utilize the clicking logs to assist the training.

D. User portrait features and item features

We further present user portrait features and item features in Table III and Table IV. We can see that all user portraits are discrete features, while two of the item features are continuous features.

IV. Method & Experiments

The overall structure of our method is shown in Fig. 4. In what follows, we will introduce each part of our framework. The ablation study results are shown in Table V.

TABLE III: User portrait features.

user features	$f_{u,1}$	$f_{u,2}$	$f_{u,3}$	$f_{u,4}$	$f_{u,5}$	$f_{u,6}$	$f_{u,7}$	$f_{u,8}$	$f_{u,9}$	$f_{u,10}$
# unique values in train set	3	1363	20	10	195	49	3	11	2	2164
# unique values in test set	3	1319	19	10	191	47	3	13	2	2054
discrete or continuous (disc./cont.)	disc.	disc.	disc.	disc.	disc.	disc.	disc.	disc.	disc.	disc.

TABLE IV: Item features.

item features	$f_{i,1}$	$f_{i,2}$	$f_{i,3}$	$f_{i,4}$	$f_{i,5}$	$f_{i,6}$ (price)
# unique values	4	10	2	n/a	n/a	248
values	1,2,3,4	0,1,2,3,4,5,6,7,8,9	1,2	0∼1, float	0∼1, float	150∼16621, int
discrete or continuous (disc./cont.)	disc.	disc.	disc.	cont.	cont.	cont.

TABLE V: Experimental results of different models (take G as an example, it is built on F with an additional design of augmentation). The numbers in this table are ablation studies after the competition. * means the settings of the best submission during competition. ⋄ means the settings are providing unstable yet higher scores.

Model	validation	test
A MLP basic model	0.29169	0.33817
B + randomness-in-session augmentation (train)	0.29965	0.35007
C + transformer backbone	0.31140	0.36210
D + two-headed (buy and group) prediction	0.31475	0.36258
E + session-aware loss reweighting	0.33090	0.38355
F + multi-tasking with click prediction *	0.33323	0.38805
G + randomness-in-session augmentation (inference) ⋄	0.33335	0.39161

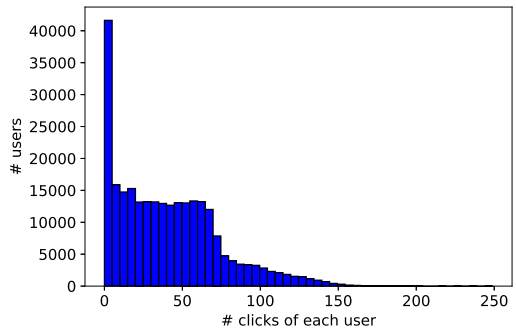


Fig. 3: Histogram of the number of clicks of each user.

A. Network design

MLP basic model (Config-A) We start with a very simple basic network. The network takes the following inputs: user profile features, user clicked items' id and features, nine exposed target items' id and features. These inputs are processed by their corresponding embedding layers, and further fed to an MLP module. Then the network predicts whether the user will buy the nine exposed target items. We propose this framework since the nine items' labels are correlated. For example, users might buy all of the first six items, only to unlock and buy subsequent items. Therefore, it is not suitable to predict the nine feedback independently, and we need to ensure the network is able to predict nine feedback simultaneously. The training of the model is supervised by a vanilla binary cross entropy (BCE) loss on each item respectively as follows,

$$\mathcal{L}_{\text{buy}} = \frac{1}{M} \sum_{u=1}^{M} \sum_{j=1}^{9} \text{BCE}(\hat{y}_{u,j}, y_{u,j}), \tag{2}$$

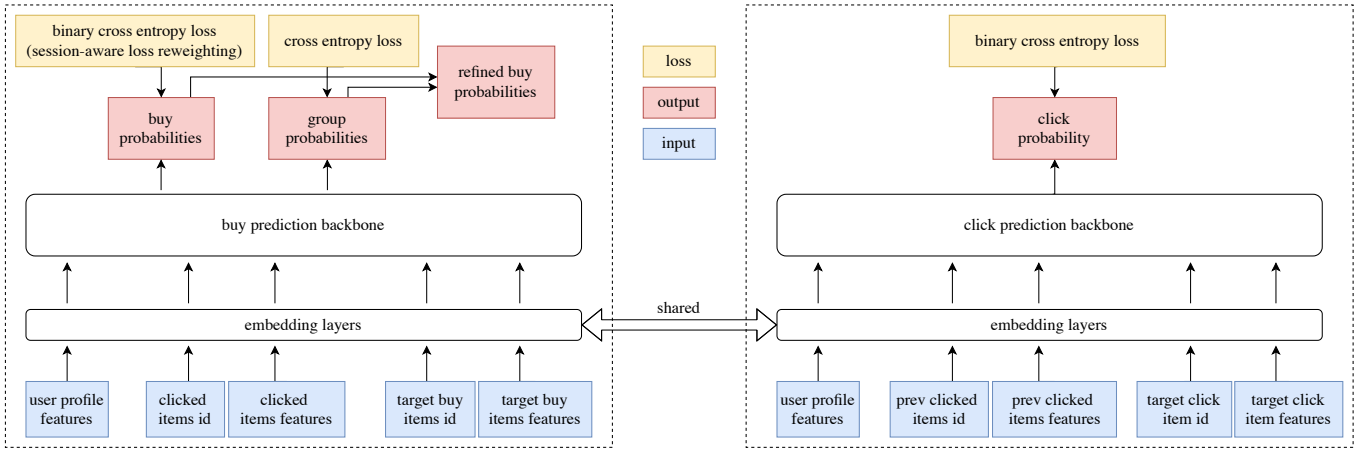
where $\hat{y}_{u,j}$ and $y_{u,j}$ denote the ground-truth and predicted feedback between user u and the exposed j-th item, respectively, and

$$\begin{aligned} \text{BCE}(\hat{y}_{u,j}, y_{u,j}) = &- \hat{y}_{u,j} \log y_{u,j} \\ &- (1 - \hat{y}_{u,j}) \log(1 - y_{u,j}) \end{aligned} \tag{3}$$

is the binary cross entropy term for each one of the nine items. We set the embedding size to 16 here, and the MLP-structure is set to $\{1440, 256, 64, 9\}$. This very simple basic model can achieve 0.29169 on validation set, and 0.33817 on test set.

Randomness-in-session augmentation (Config-B, G) To prevent over-fitting and make training more robust, we randomly shuffle items' orders within the same sessions during the training. Note that in this scenario, users are not sensitive to the items' order within the same session. However, our network treats them with different parameters. So we propose to use this augmentation technique to alleviate this shortcoming. This strategy is also used for test time augmentation, where original prediction and predictions produced by shuffled inputs are averaged to produce the final results. In our experiments, augmentation in training (**Config-B**) increases the score from 0.29169 to 0.29965 on validation set and from 0.33817 to 0.35007 on test set. However, this proposed augmentation method in inference is not so stable and sometimes might do some harm to the score. In Table V, we are just reporting the result of one experiment (**Config-G**), which improves the score.

Transformer backbone (Config-C) Instead of simple MLPs [14], [15], we switch the backbone part into a transformer [16], as their self-attention mechanism is proved to be effective on capturing inter-relations between different



(a) buy prediction (b) click prediction

Fig. 4: Overall structure of our method.

features.

$$\begin{aligned}
\mathbf{z}_0 &= \left[x^1\mathbf{E}; x^2\mathbf{E}; \cdots; x^N\mathbf{E}\right] + \mathbf{E}_{\text{pos}}, \\
\mathbf{z}_\ell &= \text{MSA}\left(\text{LayerNorm}\left(\mathbf{z}_{\ell-1}\right)\right) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \ldots L \\
\mathbf{z}_\ell &= \text{MLP}\left(\text{LayerNorm}\left(\mathbf{z}_\ell\right)\right) + \mathbf{z}_\ell, \quad \ell = 1 \ldots L \\
\mathbf{y} &= \text{LayerNorm}\left(\mathbf{z}_L\right),
\end{aligned} \tag{4}$$

where x^n is the corresponding one-hot vector of features, $\mathbf{E} \in \mathbb{R}^{N\times D}, \mathbf{E}_{pos} \in \mathbb{R}^{N\times D}$, N is the number of features, D is the embedding size, L is the number of transformer layers, and $\text{MSA}(\cdot)$ is the multi-head self attention. We set the embedding size to 128, number of layers to 3, number of self-attention head to 4, and the sizes of q,k,v in the self-attention module to 32, and MLP size to 64. Using the transformer backbone (**Config-C**) can improve our score from 0.29965 to 0.31140, and from 0.35007 to 0.36210 on validation and test set, respectively. However, we do note that this improvement compared to **Config-B** might in part come from a larger embedding and network size. We didn't do that ablation study due to limited time.

Two-headed (buy and group) prediction (Config-D) One should notice that the above simple framework might introduce some invalid buy predictions that are impossible to happen in the real world. For example, the network might predict that the user buys two items, the 1st one and the 9th one. However, this is impossible since the user has to buy all of the first 6 items in order to buy the 9th item.

Thus, in addition to the buy prediction, we propose to also predict the group (as defined in Section III-B) of each user, which forms a two-headed prediction network, as shown in Fig. 4(a). This group prediction part is supervised by a cross entropy loss as follows,

$$\mathcal{L}_{\text{group}} = \frac{1}{M}\sum_{u=1}^{M}\sum_{j=1}^{4} -\hat{g}_{u,j}\log g_{u,j}, \tag{5}$$

where $\hat{\mathbf{g}}_u \in \mathbb{R}^4$ is a one-hot ground-truth vector indicating which group user u belongs to. Here $\mathbf{g}_u \in \mathbb{R}^4$ is the predicted group vector (after a *softmax* layer). The loss is added with previous ones and back-propagated together as follows,

$$\mathcal{L} = \lambda_{\text{buy}}\mathcal{L}_{\text{buy}} + \lambda_{\text{group}}\mathcal{L}_{\text{group}}, \tag{6}$$

where we set $\lambda_{\text{buy}} = 0.8, \lambda_{\text{group}} = 0.1$. After training, the predicted group vector $\mathbf{g}_u$ will be used to refine and fix the unreasonable predicted buying behavior of the nine exposed items $\mathbf{y}_u \in \mathbb{R}^9$ as follows,

$$\mathbf{y}_u = \begin{cases}
[0,0,0,0,0,0,0,0,0] & \arg\max_j \mathbf{g}_u = 0 \\
[y_{u,1}, y_{u,2}, y_{u,3}, 0,0,0,0,0,0] & \arg\max_j \mathbf{g}_u = 1 \\
[1,1,1,y_{u,4}, y_{u,5}, y_{u,6}, y_{u,7}, y_{u,8}, y_{u,9}] & \arg\max_j \mathbf{g}_u = 2 \\
[1,1,1,1,1,1,y_{u,7}, y_{u,8}, y_{u,9}] & \arg\max_j \mathbf{g}_u = 3.
\end{cases} \tag{7}$$

After refined using the group predictions, our score improves from 0.31140 to 0.31475 on validation set and from 0.36210 to 0.36258 and test set.

Session-aware loss reweighting (Config-E) To better model users' buying behaviors, we classify the nine exposed items into four types (weak positive, strong positive, strong negative, weak negative) as shown in Fig. 5.

- For sessions before the last session user has unlocked, items should be treated as **weak positives**, as the user might buy these items only to unlock the later sessions.
- For the last session user has unlocked, items should be treated as **strong positives** and **strong negatives**. As the user unlocked and stopped in this session, items bought or not bought should be classified as strong signals.
- For later locked sessions, items should be treated as **weak negatives**, as users haven't unlocked these sessions, we should not assume too strong preferences on these items.

In practice, we assign different weights $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ for the above 4 types of items. The formally defined loss can be

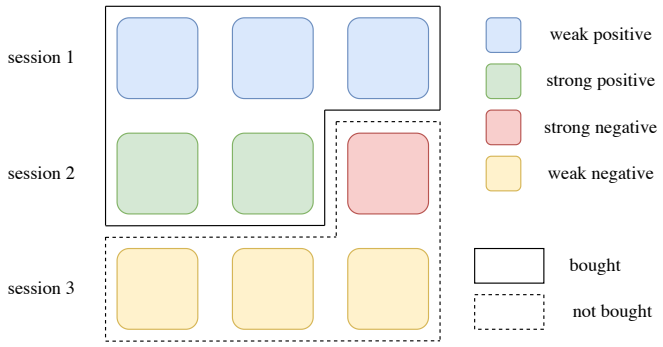


Fig. 5: Session-aware loss reweighting

written as follows,

$$\mathcal{L}_{\text{buy-reweight}} = \frac{1}{M}\sum_{u=1}^{M} \begin{cases} \sum_{j=1}^{9} \lambda_4 \Gamma_{u,j} & \hat{\mathbf{g}}_u = [1,0,0,0] \\ \sum_{j=1}^{3} \Lambda_{\lambda_2,\lambda_3,u,j} + \sum_{j=4}^{9} \lambda_4 \Gamma_{u,j} & \hat{\mathbf{g}}_u = [0,1,0,0] \\ \sum_{j=1}^{3} \lambda_1 \Gamma_{u,j} + \sum_{j=4}^{6} \Lambda_{\lambda_2,\lambda_3,u,j} + \sum_{j=7}^{9} \lambda_4 \Gamma_{u,j} & \hat{\mathbf{g}}_u = [0,0,1,0] \\ \sum_{j=1}^{6} \lambda_1 \Gamma_{u,j} + \sum_{j=7}^{9} \Lambda_{\lambda_2,\lambda_3,u,j} & \hat{\mathbf{g}}_u = [0,0,0,1] \end{cases}, \tag{8}$$

where $\Gamma_{u,j}$ and $\Lambda_{\lambda_2,\lambda_3,u,j}$ denote losses for weak positive/negative items and strong positive/negative items, respectively, which are formulated as follows,

$$\begin{aligned} \Gamma_{u,j} &= \text{BCE}(\hat{y}_{u,j}, y_{u,j}) \\ \Lambda_{\lambda_2,\lambda_3,u,j} &= \lambda_2\ \hat{y}_{u,j}\ \text{BCE}(\hat{y}_{u,j}, y_{u,j}) + \\ &\quad \lambda_3\ (1-\hat{y}_{u,j})\ \text{BCE}(\hat{y}_{u,j}, y_{u,j}). \end{aligned} \tag{9}$$

In our experiments, we replace the original $\mathcal{L}_{\text{buy}}$ with $\mathcal{L}_{\text{buy-reweight}}$, and set $\lambda_1 = 0.5, \lambda_2 = 1, \lambda_3 = 1, \lambda_4 = 0.5$. This design can greatly boost our score from 0.31475 to 0.33090 on validation set, and from 0.36258 to 0.38355 on test set.

Multi-tasking with click prediction (Config-F) Apart from the buy prediction network described in Fig. 4(a), we propose to use another click prediction auxiliary network (Fig. 4(b)) to assist the learning procedure. Note that the two networks share the same embedding layers. The click prediction network takes the following inputs: user profile features, the previously clicked items' id and features, target items' id, and features. It is trained to predict whether the user will click the target item or not. The loss function is defined as follows,

$$\mathcal{L}_{\text{click}} = \frac{1}{M}\sum_{u=1}^{M} \text{BCE}(\hat{c}_u, c_u), \tag{10}$$

where $\hat{c}_u, c_u$ are groundtruth and predicted feedback from user u to his/her target item, and

$$\begin{aligned} \text{BCE}(\hat{c}_u, c_u) = &- \hat{c}_u \log c_u \\ &- (1-\hat{c}_u)\log(1-c_u). \end{aligned} \tag{11}$$

is the binary cross entropy term. The loss is added with previous ones and back-propagated together as follows,

$$\mathcal{L} = \lambda_{\text{buy}}\mathcal{L}_{\text{buy-reweight}} + \lambda_{\text{group}}\mathcal{L}_{\text{group}} + \lambda_{\text{click}}\mathcal{L}_{\text{click}}, \tag{12}$$

where we set $\lambda_{\text{buy}} = 0.8, \lambda_{\text{group}} = 0.1, \lambda_{\text{click}} = 0.1$, and use the same network hyper-parameters as the buy prediction network here. With the auxiliary click prediction network multi-tasking, our score is improved from 0.33090 to 0.33323 and 0.38355 to 0.38805 on the validation set and test set, respectively.

Final submission Our final best submission during the competition (0.33687 on validation set, 0.39224 on test set) is achieved by **Config-F**, as shown in in Table V. That training instance shows much better performance than our ablation studies conducted after the competition. However, these methods are still suffering from the performance variances with different random seeds, which may be caused by the scale of the dataset. We leave the efforts to address the issue of unstable performances as future work.

B. Train/validation split by user portrait

Although the competition guidelines want us to recognize each buying entry as an individual user, we notice that there are entries with identical clicking histories and user portrait features (which means the same user produces two entries). Thus, it is more proper to split train and validation sets while taking the above observation into consideration. We propose to view all entries with identical user portrait features as the same user and use 85% users as train set and the rest 15% users as the validation set. This results in 243,775 and 16,312 entries for the train set and validation set, respectively.

C. Other settings

We use Adam [17] with default hyper-parameters in PyTorch [18]. The batch size is set to 32, and the learning rate is set to 1e-2 for ten epochs. Colab with one P100 GPU is used as our training platform, and each model takes about 2∼3 hours to train. Clicking data in the test set of both track-1 and track-2 are used during our training. Checkpoint with the best score on the validation set is used for evaluation. All continuous features are discretized into bins.

V. Conclusion

In this paper, we propose a framework for item combination prediction. Specifically, we propose several delicate designs to improve the performance, namely randomness-in-session augmentation, transformer backbone, two-headed prediction, session-aware loss reweighting, and multi-tasking with click prediction. Extensive experiments have proved the effectiveness of our framework.

We have also tried several things that conceptually make sense but did not improve the score. Firstly, we tried an attention-like deep interest network [14] to reweight user clicked items, however, it didn't improve the final score. Given that we do not know how click data is collected, we think that users might present different preferences in the scenario where click data is collected. And thus, making the model more

complex in this aspect doesn't help. Secondly, we tried to add user embedding into the network yet encountered severe over-fitting in training. Adding mini-batch aware regularization [14] can reduce over-fitting, however, it still cannot make improvements to the final score. Due to the fact that most users only have one training entry, this result is not very surprising. In addition, we tried adding timestamp as a feature, however, it also didn't help. We originally thought that weekends or holidays might affect user behaviors.

Future works shall include in-depth analysis and utilization with the actual meaning of user features, item features, and clicking data. It would also be interesting to investigate other network architectures that could address the multi-feedback item combination prediction scenario. Since our work does not introduce the model ensemble technique, it is also a promising direction for future works.

REFERENCES

[1] K. Wang, Z. Zou, Q. Deng, Y. Shang, M. Zhao, R. Wu, X. Shen, T. Lyu, and C. Fan, "Rl4rs: A real-world benchmark for reinforcement learning based recommender system," *ArXiv*, vol. abs/2110.11073, 2021.

[2] "Ieee bigdata cup 2021 compeition overview," https://www.kaggle.com/c/bigdata2021-rl-recsys, accessed: 2021-09-10.

[3] J. Chang, C. Gao, X. He, D. Jin, and Y. Li, "Bundle recommendation with graph convolutional networks," in *Proceedings of the International ACM SIGIR conference on Research and development in Information Retrieval (SIGIR)*, 2020, pp. 1673–1676.

[4] P. Resnick and H. R. Varian, "Recommender systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[5] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.

[6] L. Wu, X. He, X. Wang, K. Zhang, and M. Wang, "A survey on neural recommendation: From collaborative filtering to content and context enriched recommendation," *arXiv preprint arXiv:2104.13030*, 2021.

[7] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He *et al.*, "Graph neural networks for recommender systems: Challenges, methods, and directions," *arXiv preprint arXiv:2109.12843*, 2021.

[8] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 2009, 2009.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[10] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the International ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*, 2020, pp. 639–648.

[11] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," in *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2020, pp. 1001–1010.

[12] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 1725–1731.

[13] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018, pp. 1754–1763.

[14] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018, pp. 1059–1068.

[15] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the International Conference on World Wide Web (WWW)*, 2017, pp. 173–182.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998–6008, 2017.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference for Learning Representations (ICLR)*, 2015.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, pp. 8026–8037, 2019.