A Diverse Models Ensemble for Fashion Session-Based Recommendation

Benedikt Schifferer
bschifferer@nvidia.com
NVIDIA
Germany

Jiwei Lui
jiweil@nvidia.com
NVIDIA
Vancouver, British Columbia, USA

Sara Rabhi
srabhi@nvidia.com
NVIDIA
Canada

Gilberto Titericz
gmoreira@nvidia.com
NVIDIA
Brazil

Chris Deotte
cdeotte@nvidia.com
NVIDIA
USA

Gabriel de Souza P. Moreira
gmoreira@nvidia.com
NVIDIA
Brazil

Ronay Ak
ronaya@nvidia.com
NVIDIA
USA

Kazuki Onodera
konodera@nvidia.com
NVIDIA
Japan

ABSTRACT

Session-based recommendation is an important task for domains like e-commerce, that suffer from the user cold-start problem due to anonymous browsing and for which users preferences might change considerably over time. The RecSys Challenge 2022, organized by Dressipi, is focused on the session-based recommendation problem for the fashion e-commerce domain. In this paper, the NVIDIA RAPIDS and NVIDIA Merlin teams present their solution that placed 3rd in the challenge. Among the most effective techniques we found sessions augmentation and ensembling a very diverse set of statistical, machine learning and deep learning models. Our recommendation pipeline is composed of three stages, where the first level is focused on candidate generation and the others refine the recommendation ranking for more robust and accurate recommendations.

CCS CONCEPTS

• **Information systems** → Personalization; Recommender systems; • **Computing methodologies** → Neural networks.

ACM Reference Format:
Benedikt Schifferer, Jiwei Lui, Sara Rabhi, Gilberto Titericz, Chris Deotte, Gabriel de Souza P. Moreira, Ronay Ak, and Kazuki Onodera. 2022. A Diverse Models Ensemble for Fashion Session-Based Recommendation. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3556702.3556821

RecSys '22, September 18–23, 2022, Seattle, WA, USA

ACM ISBN 978-1-4503-9856-5/22/09.
https://doi.org/10.1145/3556702.3556821

1 INTRODUCTION

In many recommendation domains it is not possible to trust in long-term information about users. Users might browse anonymously (user cold-start problem) or their preferences might vary a lot depending on their current intent or seasonality. Session-based recommendation is a popular task that targets such scenarios for domains like e-commerce and news recommendation [5, 6, 11].
The RecSys Challenge 2022, organized by Dressipi, was focused on fashion session-based recommendations. The task was to predict the item (product) that will be purchased in a session given a sequence of items that were clicked by the user before. In addition, the fashion domain also suffers from the item cold-start problem, as new items are released every week for seasonal or commercial reasons, with no past interactions available. To deal with such fresh items, Dressipi made available a very comprehensive set of item metadata features (e.g. color, length, neckline, etc.) that could be used to infer which users could be interested in such brand new products.
In general, the train dataset covered sessions between Jan 2020 and May 2021. Teams should predict the purchased items for sessions of the test set from the month of June 2021. The evaluation metric was MRR@100, which assessed the ranking quality of the top-100 recommended items. In this paper, we present the NVIDIA RAPIDSAI solution for the RecSys Challenge 2022, where we placed 3rd in the final leaderboard. Our solution [1] key components involve data augmentation techniques and ensembling diverse types of models. Our recommendation pipeline is designed to provide more accurate and robust predictions, being composed of three stages. In the Stage 1 we build of a diverse set of candidate generation models based on statistical algorithms, e.g. Session kNN, machine learning models using Gradient Boosting Decision Trees (GBDT) and neural networks based on Multi-Layer Perceptron (MLP), Recurrent Neural Networks (RNN) and Transformers. For Stage 2 we ensemble predictions from Stage 1 models with other features using weighted

[1] The solution code will be published for reference and reproducibility on https://github.com/NVIDIA-Merlin/competitions/

average and GBDT models. Finally, for Stage 3, we use weighted average to combine predictions from the previous stage.

2 PREPROCESSING AND FEATURE ENGINEERING

The dataset has a tabular data structure and contains for training 4,7M interactions distributed in 1M sessions, with average session length of 4.74. In previous challenges [7, 15–17], feature engineering was a key component of our solutions to make information more easily accessible to model and data augmentation to increase the training dataset.

2.1 Feature Engineering

The nature of the dataset limited the options for feature engineering. User information is not available and session information contains only the viewed items and timestamp. We have created features based on the session length (number of viewed items and duration) and elapsed time between two consecutive views. The item features provided in the dataset are anonymized sparse categorical features, which cannot be interpreted by humans. Therefore, we could not use expert domain knowledge to design specific features for the fashion domain.

2.2 Feature Selection

Different sets of features were used by our models. For some models, the item features were pre-selected based on their frequency and the stability of their distribution over time. In order to quantify the feature's stability, we calculated how often this particular feature is present during each month of train data. In general, features were selected by assessing their individual impact in the models validation score and public LB.

2.3 Data Augmentation

The train dataset contained 1M sessions. Our most important preprocessing technique was data augmentation, so that we could make training data 5-10x larger. Our insight for that came from the fact that viewed items co-occur in sessions very often with each other and with the purchased items. Thus, we created variations of existing sessions like randomly truncating or shuffling viewed items, swapping the purchased item with a viewed item (which then becomes the target) and other approaches, like illustrated in Figure 1. It should be noted that data augmentation is applied to the **training data only**. In inference, test data are used as is without any augmentation.

2.4 Validation set design

A common strategy in competitions is to generate a local validation dataset to evaluate experiments before submitting a solution for Cross-Validation (CV). We reserved the last month (May 2021) as our local validation dataset, using available data before that for training. As Dressipi augmented the test dataset by randomly truncating the sessions by 0-50%, we applied the same strategy to our local validation dataset. We evaluated a model based on the original sessions of May 2021 and also on truncated versions of the sessions. But even with that approach it was hard for the team

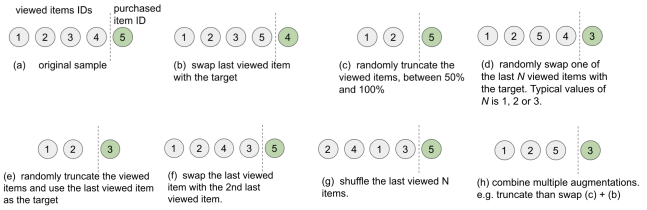

Figure 1: Augmentation functions illustration. (a) is an original session with a sequence of clicked items and the rest are variations from the original session.

to prepare a validation set with high correlation with the CV and public Leaderboard (LB) scores.

3 MODELS AND ENSEMBLING

Our solution consists of a three-stage pipeline of ensembling and stacking multiple deep learning, boosted-tree and statistical models, as shown in Figure 2. Similar to previous RecSys competitions, ensembling diverse models has shown to be beneficial for making predictions more robust and improving recommendation accuracy.

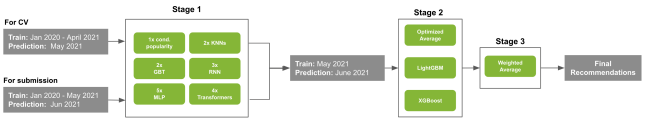

Figure 2: NVIDIA RAPIDS AI team solution - three-stage pipeline of ensembling and stacking diverse models.

3.1 Stage 1

The first stage models generate top-k candidate items with prediction scores. The 17 models we used for stage 1 can be grouped in six types: (1) conditional popularity, (2) Session k-Nearest Neighbors (kNN) (3) LightGBM (4) recurrent neural networks (5) multilayer perceptron neural networks and (6) transformers networks. Each of the 17 models can be an ensemble of an algorithm or base architecture with different hyperparameters, some architecture variation (e.g. ensembling a GRU with an LSTM model) and/or trained with different random seeds, often called bags. Those model types are described in the next sections.

3.1.1 Time Weighted Conditional Popularity

. This model is based on the pairwise co-occurrence frequency between a clicked item and the purchased item, and between a clicked item with the previous 1st, 2nd and 3rd clicked item. The frequencies of each combination are concatenated and we applied a weighted decay based on time, so that older combinations get less weight. For each item in each session we retrieve a list of most frequent 100 co-occurring items, which are then combined and sorted by the time decay. The frequency features are calculated using training data only.

3.1.2 Session kNN

. In a recent empirical study on session-based recommendation

algorithms [14] it was demonstrated that simpler statistical algorithms based on Session k-Nearest Neighbors (kNN) could be able to outperform neural based models, like the ones based on RNNs like GRU4Rec [8] and on Graph Neural Networks (GNN) like SR-GNN [19]. We also observed the effectiveness of the Session kNN algorithms as baselines in the Transformers4Rec paper [6], which are comparable to some of the neural models when using only the item id as input feature. The Session kNN algorithms have in general two stages: (1) the sessions neighbor search, where the k most similar sessions are retrieved based on session items and (2) scoring, where the items from the neighbor sessions are scored and ranked for recommendation. For this competition, we decided to include two Session kNN algorithms in our ensemble solution to add some diversity: SkNN [13] and VSTAN [14]. We implemented two customizations for both algorithms. First, as for the competition dataset it was ensured that the purchased item does not appear as a clicked item in the sequence (to avoid easy recommendation), we ignored the neighbor sessions whose purchased item is present as a clicked item in the current session. Second, we included a configurable additional weight for the last item (purchased product) of neighbor sessions, so that purchased items could have a higher weight during the scoring stage. We ensembled the prediction scores of four customized V-SkNN models using different periods before the test set start date: last 120 days, last 90 days, last 60 days, last 30 days. For the VSTAN algorithm, we ensembled predictions using the original VSTAN implementation and the one with our customizations. Both ensembles used weighted averages of prediction scores.

3.1.3 *Gradient Boosting Decision Trees (GBDT)*

. GBDT is a classic model for tabular data. In this challenge, we construct pairwise samples, where each sample consists of a sequence of viewed items and a candidate item. The target of the pairwise sample is binary, indicating the candidate item is purchased or not in the session. For each session, 200 candidates are generated using the algorithm described in 3.1.1. Hence 200 pairwise samples are generated for each session. We handcraft features for each pairwise sample using only the train dataset such as count of viewing and purchasing the candidates in the past 30 days, the frequency of coexistence of the viewed items and candidates and so on. The complete feature sets are detailed in the Appendix C.4. We implemented two GBDT models: LightGBM with lambda rank objective and Xgboost with binary classification logistic regression objective.

3.1.4 *MultiLayer Perceptron (MLP)*

. In recent ML and RecSys competitions deep learning based models have performed as the top solutions. One of the neural networks uses the MultiLayer Perceptron (MLP) block as the main component (see Figure 4 in Appendix C.1). The model's input are the viewed item ids and item features, which are fed through an embedding layer. The sequential embeddings are averaged and concatenated with the embeddings of the last viewed item. The resulting vector is fed through an MLP block. Before the softmax activation function, weight-tying (see Appendix B.2) was applied which improved the model performance. Label smoothing and softmax temperature (see Appendix B.1) are used to avoid the effects of noisy browsing in sessions. In general, multiple data augmentation techniques were used to increase the dataset, including adding the next item prediction task. In some version (e.g. MLP Ensemble 2), a context vector for the event type was added (see Appendix B.4), but many other versions (e.g. MLP Next Item Prediction, Ensemble 1) were trained without differentiation between the event type. Another modification (e.g. MLP Ensemble Multi-Task) is to extend the categorical cross entropy by a multi-task learning loss. Instead of predicting only the target item, the models predict additionally the item features of the target. For example, the model does not only predict the specific item, but also predicts the color, category, material of the item. The training process used a stepwise learning rate decay and truncated the training dataset to use more recent data over training progress. For some models, the training data was sorted by time in the training process. Each model was trained 6 times with random seeds (bagged). Ensembles are using the same main strategies, but applying different hyperparameters, which can be found in our solution online.

3.1.5 *RNN*

. Recurrent neural networks are the most popular choice for modeling sequential data, especially when the short term dependency is stronger than the long term dependency. We designed a customized RNN which achieves a 0.2044 public leaderboard score. The model architecture is shown in Figure 6 of Appendix C.3. We use unidirectional GRUs [3] to encode the last N viewed items based on both item_id and item features. We consider all the 28,144 items as the candidates and encode them using the same shared embedding table, and a MLP consisting of multiple fully connected layers and Relu activation functions. The dot product of the sequential encoding and the candidate items encoding is calculated to provide logits before softmax. Multiple techniques are used to improve the performance of RNN including pipelined data augmentation, sample rolling dataloader and candidate masking. The details can be found in Appendix C.3.

3.1.6 *Transformers*

. In Transformers4Rec [6], we conducted extensive empirical experiments showing the Transformer model's effectiveness in short-sequences typically observed in session-based recommendations. Additionally, the paper demonstrated the importance of considering side information, such as item category and time features. Motivated by these observations and the results achieved in previous session-based competitions [15, 16], we built different variants of Transformer-based architectures using the open-source Transformers4Rec library. In particular, we used the XLNET [20] architecture as the core transformer block [18], which was trained with a variety of prediction tasks (described in Appendix B.3) and inputs. We subselect two groups of item features: features that are present for more than 80% of items and features with non-constant distribution over time. We tested different input combinations and using the two groups of features in addition to the item ids led to the best CV score. Similar to the MLP model, weight-tying (Appendix B.2), label smoothing, and temperature (Appendix B.1) are used in the output layer of all transformers models. As all prediction tasks predicted views and purchases, we leveraged the target event type using Latent Cross technique (see Appendix B.4).

3.1.7 *Models prediction post-processing*

. For all Stage 1 candidate generation models we perform two filtering approaches on predicted items, based on the dataset and

competition design. First, the dataset contains a white list of items purchased for the test dataset. We create a similar list for our validation set and filter out from model predictions any item which is not included in the list. Second, we remove recommended items which were viewed during the specific sessions, as the host removed the viewed event of the target item from the session.

3.2 Stage 2

A common technique in competitions is to ensemble models by stacking, where prediction scores from input models are combined. For the second stage, we take as input the 17 models predictions for validation set (May 2021) and combine with other session and candidate features. We use three stacking approaches to combine the predictions from the first stage: (1) Optimized Weighted Average of items ranking, (2) LightGBM and (3) XGBoost.

3.2.1 *Additional session and candidates features*

. For each item feature, we add a boolean variable indicating whether the candidate and last viewed item share the same value. In addition, we measure the percentage of features shared between the candidate and all items viewed in the session.

3.2.2 *Optimized Weighted Average*

. This approach is a weighted average of stage 1 models. We learn the optimal weights by optimizing the MRR on the validation set. As each model has only one weight, we do not apply a separate train-validation split.

3.2.3 *LightGBM*

. LightGBM [12] for stage 2 is trained with the lambda rank loss objective. We filter the candidates generated from stage 1, keeping only items recommended by at least two stage 1 models. We average 25 bags. For each bag, the data is randomly splitted by session ids into 85% train and 15% validation.

3.2.4 *XGBoost*

. XGBoost [2] for stage 2 is trained with binary logistic objectives. The candidates are filtered in the same way as for the LightGBM model. The stage 1 validation data (May 2021) is used for training, splitted into 4 folds of equal size, where each fold with consecutive days. The model is trained using each fold as a validation dataset and the remaining folds for training. The test dataset is predicted for each training run. The pipeline is repeated for 3 bags with different seeds.

3.3 Stage 3

Finally, in the third and last stage, we ensemble our 2nd stage models' predictions by a weighted average of items ranking, using the same approach we used in Stage 2 for CV. The final weights were 0.5 for optimized average, 0.16 for LightGBM and 0.32 for XGBoost. We used a conservative approach by providing the highest weight to the optimized average ensemble as it has the lowest risk of overfitting.

4 RESULTS

In Figure 3, we observe that the Session k-NN (V-SkNN and VSTAN) statistical algorithms provide strong baselines for the session-based recommendation, by outperforming XGBoost and transformer models trained with only item ids and CLM. Our machine learning models used different sets of features and data augmentation techniques. But in general we can observe that LightGBM, MLP, and RNN-based models trained with augmentation techniques described in section 2.3 were able to outperform transformer architectures (with different training tasks). It suggests that data augmentation was very effective for the proposed relatively small dataset. Finally, we observed that multi-task learning improved MLP (MLP Multi-Task Ensemble) and Transformers (Transformer RTD-All) models performance scores by 0.001 and 0.002, respectively.

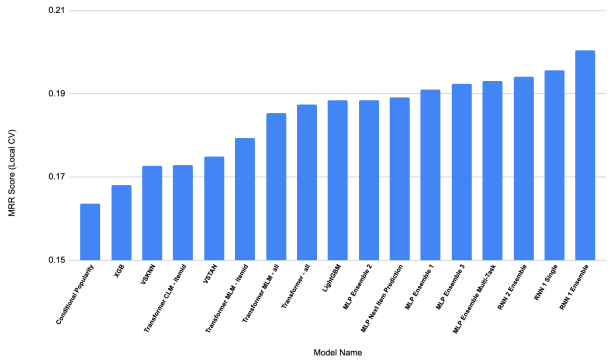

Figure 3: Validation (CV) scores of the 17 models used in the Stage 1 of our pipeline.

Table 1: LB scores for our Stage 2 and Stage 3 models

	Stage 2			stage 3
	Optimized Weighted Average	**XGBoost**	**LightGBM**	**Weighted Average**
Public LB	0.2073	0.2074	0.2076	0.2082
Final LB	-	-	-	0.2086

In Table 1 we present the LB scores for Stage 2 and 3 models. In particular, LightGBM achieved the best score out of all three proposed stage-2 models. The third stage weight-average model gave the best performance of 0.2082, suggesting that the three 2nd-level models captured different prediction information.

5 CONCLUSION

In this paper, we present the solution of the NVIDIA RAPIDSAI team for the RecSys Challenge 2022, which focused on session-based recommendation for the fashion e-commerce domain. Our three stage pipeline included different types of models, including statistical models, GBDTs and neural networks like MLP, RNN and Transformers. We could observe from the results that our data augmentation approach for generating new sessions from existing ones helped most of the models to improve their performance. In addition, ensembling a diverse set of models has shown to be effective to improve prediction accuracy.

ACKNOWLEDGMENTS

The authors wish to thank our colleagues from NVIDIA RAPIDS and Merlin teams that have developed those great libraries which we leverage in our solution.

REFERENCES

[1] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Jia Li, Vince Gatto, and Ed H Chi. 2018. Latent cross: Making use of context in recurrent recommender systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 46–54.

[2] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, et al. 2015. Xgboost: extreme gradient boosting. *R package version 0.4-2* 1, 4 (2015), 1–4.

[3] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. arXiv:1406.1078 [cs.CL]

[4] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555* (2020).

[5] Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. 2018. News session-based recommendations using deep neural networks. In *Proceedings of the 3rd workshop on deep learning for recommender systems*. 15–23.

[6] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4Rec: Bridging the Gap between NLP and Sequential/Session-Based Recommendation. In *Fifteenth ACM Conference on Recommender Systems*. 143–153.

[7] Chris Deotte, Bo Liu, Benedikt Schifferer, and Gilberto Titericz. 2021. GPU Accelerated Boosted Trees and Deep Neural Networks for Better Recommender Systems. In *RecSysChallenge'21: Proceedings of the Recommender Systems Challenge 2021*. 7–14.

[8] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[9] Hakan Inan, Khashayar Khosravi, and Richard Socher. 2016. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462* (2016).

[10] 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*. 44–49.

[11] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017. Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts. *User Modeling and User-Adapted Interaction* 27, 3 (2017), 351–392.

[12] 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. *Advances in neural information processing systems* 30 (2017).

[13] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 28, 4 (2018), 331–390.

[14] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2021. Empirical analysis of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction* 31, 1 (2021), 149–181.

[15] Gabriel de Souza P Moreira, Sara Rabhi, Ronay Ak, Md Yasin Kabir, and Even Oldridge. 2021. Transformers with multi-modal features and post-fusion context for e-commerce session-based recommendation. *arXiv preprint arXiv:2107.05124* (2021).

[16] Benedikt Schifferer, Chris Deotte, Jean-Francois Puget, Gabriel de Souza Pereira Moreira, Gilberto Titericz, Jiwei Liu, and Ronay Ak. 2021. Using Deep Learning to Win the Booking. com WSDM WebTour21 Challenge on Sequential Recommendations.. In *WebTour@ WSDM*. 22–28.

[17] 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 2020*. 16–23.

[18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[19] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 346–353.

[20] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

A RELATED WORK

Many competitions on recommender systems have been organized recently. The most traditional one is the RecSys Challenge, which is organized yearly by the ACM RecSys community. In the last two years, NVIDIA have participated and managed to win four of those contests in a row, namely the RecSys Challenge 2020 [17] and 2021 [7] (organized by Twitter), the WSDM 2021 WebTour Workshop Challenge (organized by Booking.com) [16], and the SIGIR 2021 Workshop on E-commerce Data Challenge (organized by Coveo) [15]. NVIDIA have been using those competitions as an opportunity to test and improve its frameworks for GPU-accelerated Data Science - RAPIDS - and for Recommender Systems - Merlin - the latter including the NVTabular preprocessing library, the Merlin Data Loader and Transformers4Rec library. In [10] it was presented an analysis of the top solutions for the RecSys Challenge competitions between 2017 and 2019 which demonstrates that Deep Learning models were in general surpassed by non neural models, in particular based on Gradient Boosted Trees. That was also the case for our winning solution for the RecSys Challenge 2020 [17]. Since 2021, we have observed the success of Deep Learning models in the top competition solutions, including ours for the other three competitions we won [7, 15, 16]. In particular, the WSDM 2021 WebTour Workshop Challenge and SIGIR 2021 Workshop on E-commerce Data Challenge were focused on the session-based recommendation task similarly to this RecSys Challenge 2022, where information is restricted to what happened in the current session only, with no past user interactions access. Our current solution borrows some techniques we used for those competitions, like data augmentation [16], Transformer architectures [15, 16], tying embeddings (weight-tying the item id embedding table and the output layer) [15, 16] and post-fusion context using latent cross [15] of contextual features, which we describe later in Appendix B.4.

B MODELING TECHNIQUES

We highlight in this section some important techniques used by some of the models.

B.1 Label Smoothing and Temperature

Datasets in the recommender system domain are often noisy. A user is not able to review the full catalog and make the best possible decision. As the data is collected over a longer time period, the item catalog changes and sometimes are items out of stock. The collected labels are noisy and imperfect. The gradients of noisy labels can be relatively large compared to "correct" labels and therefore, dominates the training process. To avoid the effect of noisy labels, we can control the gradients by label smoothing and temperature. Both techniques avoid large gradients for high confident, but wrong, predictions. Label smoothing changes the one-hot labels for cross-entropy into smoothie versions where the correct class is reduced from 1 to $1 - \epsilon$ and the incorrect classes get the label $\frac{\epsilon}{k}$, where k is the number of classes. Temperature scaling divides the output logits by a constant to smooth the output softmax distribution.

B.2 Weight-tying

This technique was first proposed in the NLP domain [9], where both inputs and outputs of the language are words, so they can

be represented in the same vector space. Specifically, the logits scores are computed by multiplying the item id embedding table by the final hidden representation returned by the encoder model. This technique reduces model memory requirements, as it uses the item embedding table as the output layer. Weight tying can also be viewed as a matrix factorization operation between the item representation and the final representation of the user or session. Motivated by the results shown in previous works [6, 15, 16], we used this technique when training our models. We extended the technique in our RNN 1 models. As the dataset contains item features, the item representation for the final matrix factorization operations was generated by MLP layers combining the item embedding and item features embeddings.

B.3 Training tasks of transformer-based models

As part of this competition, we added the purchased item to the sequence of interactions and trained the model to predict both events 'views' and 'purchases'. For that, we used four different techniques to define the labels. First, we used a sliding window of size 1 to predict the target position N based on the past sequence of 1,..., N-1 viewed items. We refer to this first approach as causal language modeling (CLM). The second approach, called masked language modeling (MLM), involves randomly selecting a subset of positions as targets and masking them from the input sequence. The model is trained with future context information in this scenario. In the third strategy (MLM-HigherProb), we increased the probability of masking the last item (purchased product) in the MLM task to ensure that purchased items are updated more frequently. Finally, to learn from all input items, we used an ELECTRA language model-based generator-discriminator architecture [4]. This task, called replacement token detection (RTD), requires replacing the masked positions defined by the generator (trained with MLM-HigherProb) with random tokens, then training the discriminator to identify whether a given item-id is original or replaced. Specifically, we share the Transformer-block between the Generator and Discriminator tasks and compute the final loss as the average of both tasks. In the test phase, we drop the discriminator head and get the top-100 predictions from the generator.

B.4 Latent Cross technique

To provide contextual information about the item to be predicted, we used the Latent Cross technique [1]. It consists of combining the context embedding with the hidden states of the sequential model right before the prediction layer. For the SIGIR'21 Challenge [15], the NVIDIA Merlin team leveraged this technique in order to make use of all available context information such as search and page view events, while also learning to differentiate between contextual events and product interactions during evaluation. Similarly, our proposed solution uses the event-type embeddings to provide context about what type of event is associated with the item to be predicted (whether a view or a purchase). In the test phase, we always set the event type to purchase, since we are only interested in predicting purchases. Training with the Latent event embeddings boosted the performance of the baseline model score (validation MRR) by 0.003.

C STAGE 1 MODEL ARCHITECTURES DETAILS

C.1 Multi-Layer Perceptron (MLP)

Figure 4 describes the MLP architecture. Historical input data is passed through an embedding layer. The embeddings are averaged per feature. The item embedding and item feature embeddings of the last viewed item will be kept concatenated with the averaged embeddings. The results are fed through fully connected layers. A context vector (event type) can be added. In the case of multi-task learning, each task has a separate matrix factorization head. Embedding tables of the same feature are shared.

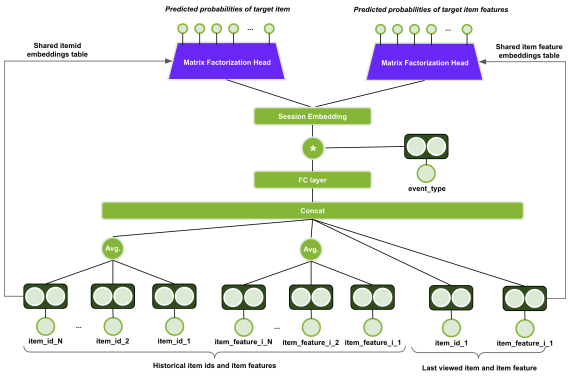

Figure 4: Visualization of the MLP architecture.

C.2 Transformer-based model

Figure 5 describes the Transformer-based architecture. Historical data of item ids and their features are passed through embedding layers. For each position, we concatenate the embeddings from all features and project them using an MLP layer to the same hidden dimension of the XLNET model. The resulting sequence of interactions are passed through a series of XLNET blocks to produce the sequence of prediction vectors. We leverage the event type embedding using a simple concat and the final context-aware prediction vectors are fed to the matrix factorization head to get the final output probabilities scores.

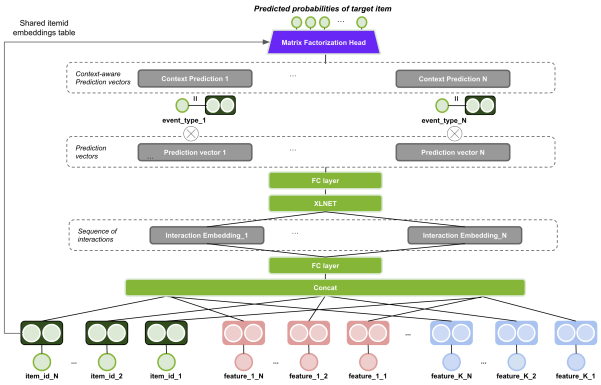

Figure 5: Visualization of the transformer architecture.

C.3 RNN-based model

Our best single model is a customized recurrent neural network, which achieves 0.2028 public leaderboard score. Averaging many variants of this neural network achieves 0.2044 public leaderboard score. The model architecture is shown in Figure 6. The last N viewed items go through a shared embedding lookup stage which encodes both item_id and item_feature. The embeddings are sequentially encoded using a GRU layer. The candidate items also go through the same embedding lookup stage and are encoded with a MLP consisting of multiple fully connected layers and Relu activation functions. We calculate the dot product between the sequential encoding of the view items and the MLP encoding of all candidate items. The final result is a vector of length 28144, representing the probability of being the purchased item.

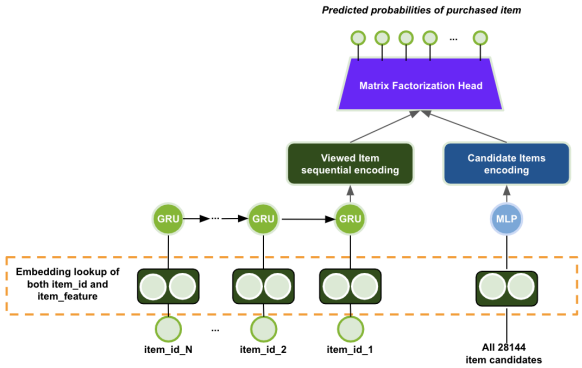

Figure 6: The customized RNN architecture.

One key design choice is to keep the complete feature information of items in embedding lookup. The challenge is that each item_id corresponds to a different number of feature categories. We use an item ID to item feature lookup table so that each item ID maps to a sparse item feature category vector where non-existing feature categories are padded with zero. As shown in Figure 7, the item ID embedding vector and item feature embedding vector are concatenated into one vector.

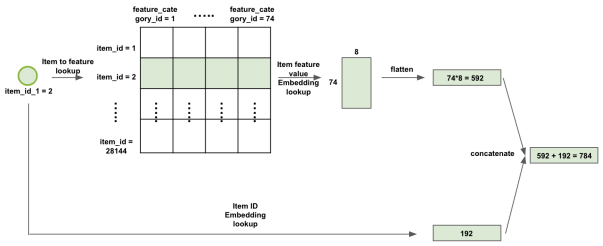

Figure 7: Embedding lookup of both item_id and item_feature. The item ID to item feature table is sparse and non-existing features are padded with zero.

We designed three variants of the proposed neural network architecture in terms of different sequential encoding implementations. As shown in Figure 3, the most intuitive choice is using the GRU output of the last viewed item as the encoding of the entire sequence. In this case, the gradients can only back propagate from the last item through time to previous items and could suffer from gradient vanishing. Therefore, we designed Figure 8 (b) and (c) to allow gradients to flow directly to previous items to improve convergence. Our best single model is achieved by option (c).

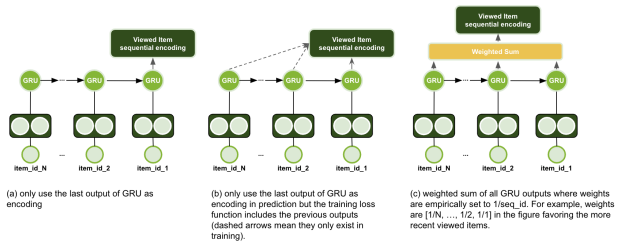

Figure 8: Three variants of sequential encoding of viewed items.

In addition to model architecture, the training dynamics is also critical to performance. The major techniques used in this RNN are as follows:

(1) Extensive dynamic augmentations. As shown in Figure 1, we implemented 5 different augmentation functions, which can be applied selectively and sequentially in any order and combined into one augmentation function like sklearn's pipeline. An interesting augmentation pipeline found is "random truncation + random swap one of the last two viewed items with the target". The augmentation is very efficient and applied dynamically to the samples in the dataloader. Such dynamic implementation enables fast testing of different augmentations.
(2) Sample rolling in training dataloader. We observe that a strong temporal pattern exhibits in the dataset. The newer session data has significantly more predictive power than the older data. However, simply removing older data from training only degrades the performance, possibly due to loss of information of items which only exist in older data. Therefore, we implemented a sample rolling mechanism in the training dataloader so that newer samples are trained more often than old ones over the epochs. The dataset is sorted based on session ID and timestamp so the training samples are fed to the model in the order of time from oldest to newest in each epoch. For example, in epoch 1 all the training samples are used; in epoch 2 the oldest 20% training samples are ignored and so on.
(3) Masking viewed items in training. Since the purchased item is not viewed in the session, we use a binary mask representing viewed item IDs in the session so that their probability of being purchased is reset to zero. This technique lets the model focus on the true legit candidates and improves convergence.

C.4 Gradient Boosting Decision Trees

C.4.1 Candidates generation

. We enumerated the combinations of products viewed and purchased and extracted 200 of each in order of frequency. e.g. Let's say session_id = 1 is [a, b, c, d], and d is the purchased item. Then we got these candidates, [[a, d],[b, d],[c, d]] item_id a, b and c are item_id_x, and d is item_id_y. When inference, let's say session_id = 2 is [a, e, f], the candidate of item_id_x a is item_id_y d. So we predict the probability of item_id_y d for item_id_x a. In addition to it, 200 best-selling products of the past 7 days were generated as popular candidates.

C.4.2 Feature engineering

. Using generated candidates, we created GBM. Features are generated only using the train dataset below.

- purchase count of last 30 days
- view count of last 30 days
- min, mean, max and std of purchase frequency of combination of item_id_x and item_id_y of last 28 days
- min, mean, max and std of view frequency of combination of item_id_x and item_id_y of last 28 days
- unique number of session_id of item_id_y of last 28 days

Please note that test features are generated from N days before the end of May to the end of May.