Practical Content-aware Session-based Recommendation: Deep Retrieve then Shallow Rank

Yuxuan Lei*
Xiaolong Chen*
Defu Lian†
University of Science and Technology of China
Hefei, China
lyx180812@mail.ustc.edu.cn
chenxiaolong@mail.ustc.edu.cn
liandefu@ustc.edu.cn

Peiyan Zhang
The Hong Kong University of Science and Technology
Hong Kong, China
pzhangao@cse.ust.hk

Jianxun Lian
Chaozhuo Li†
Xing Xie
Microsoft Research Asia
Beijing, China
jialia@microsoft.com
cli@microsoft.com
xing.xie@microsoft.com

ABSTRACT

This paper presents the solution of our team unirec in the KDD Cup 2023 Multilingual Recommendation Challenge.

The goal of the competition is to explore ways to improve session-based recommendation in real-world multilingual and imbalanced scenarios. Our method comprises a two-stage retrieval-then-rank strategy. In the first stage, advanced deep single models are used to score the full set of items, enabling us to obtain a smaller candidates set along with the corresponding session-item score features. In the second stage, we employ the shallow but powerful XGBoost algorithm for ranking to derive the final recommendation results. Our method ranks 3rd place in the final leaderboard of Task1. Our implementation using the recommendation library RecStudio and UniRec is publicly available at this link: https://gitlab.aicrowd.com/CXL/unirec-task1-amazon-kddcup-2023.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Session-based Recommender Systems, Language Models, Tree Boosting System, KDDCup 2023

ACM Reference Format:
Yuxuan Lei, Xiaolong Chen, Defu Lian, Peiyan Zhang, Jianxun Lian, Chaozhuo Li, and Xing Xie. 2023. Practical Content-aware Session-based Recommendation: Deep Retrieve then Shallow Rank. In *Proceedings of Amazon KDD Cup 2023 Workshop: Amazon Multilingual Recommendation System (KDDCup '23).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

*Both authors contributed equally to this research.
†Corresponding author

KDDCup '23, Aug 09, 2023, Long Beach, CA, USA

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

Table 1: Statistics of the datasets

Locale	#Sessions	#Products	Avg. session length
DE	1,340,552	518,327	4.35
JP	1,192,053	395,009	4.48
UK	1,434,054	500,180	4.12

1 INTRODUCTION

Recommender systems are vital in e-commerce platforms as they can enhance users' shopping experience while also increasing the platform's revenue. Session-based recommendation, which employs interaction records within a short session to chronologically capture user interests, is a highly common scenario.

In order to improve the performance of the session-based recommender system in multilingual and imbalanced scenarios, Amazon provided the "Multilingual Shopping Session Dataset"[6] which contains millions of user sessions from six locales and hosted the KDD CUP 2023 Challenge.

1.1 Dataset Description

The dataset for each locale comprises two components: user sessions and product attributes. All sets encompass the sequence of historical items accessed by users within a session (referred to as prev_items), with the training set additionally providing the next item label (referred to as next_item). Product attributes encompass product ID, locale, title, price, brand, color, size, model, material, author, and description. The fundamental statistics of the datasets are summarized in Table 1.

1.2 Task Description

Task 1 focuses on next product recommendation, aiming to predict the subsequent product a user is likely to engage with, considering both the user session data and product attributes. The datasets for Task 1 encompass three locales: German (DE), Japanese (JP), and English (UK), all characterized by relatively rich interactions. Participants in Task 1 are required to submit a sorted top 100 items list for each session in the test set, and the online evaluation metric employed is mrr@100 (Mean Reciprocal Rank).

2 METHODOLOGY

In real-world recommender systems, a vast number of items are typically present, as demonstrated in Table 1. Predicting the next

item a user is likely to engage with from such a large-scale item set effectively and efficiently poses a significant challenge for a single model. Consequently, a two-stage retrieval-then-rank strategy is commonly employed in recommender system design. For Task 1, we also adopt this classical two-stage approach, utilizing multiple single models for candidate retrieval and feature construction, followed by a ranker to reorganize the candidates for enhanced recommendation performance. Figure 1 illustrates the pipeline of our solution, while the remainder of this section details the single models and the ranker employed in this challenge.

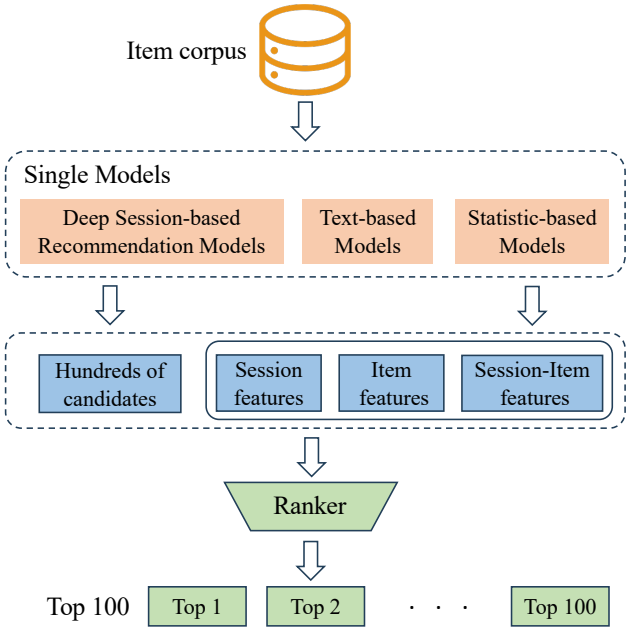

Figure 1: The pipeline of our solution.

2.1 Data Processing and Splitting

We describe our data processing and splitting in this section.

Data Processing. Price is treated as a numerical feature, and we identified an anomalous price of 40,000,000.07 in the dataset and replaced it with the mean price of items within the respective locale. The features brand, color, size, model, material, and author are deemed categorical features. To ensure consistent treatment of categories despite capitalization variations, we convert all these features to lowercase, and features that appeared fewer than three times are filtered out. Title and description are treated as text features, and we utilize both the original text features and those generated by concatenating the categorical features using the [SEP] token to obtain diverse models.

Data Splitting. Initially, for each locale, we divide the training sessions into training and validation datasets at a 0.92 to 0.08 ratio. The next_item field in sessions of validation dataset is used to assess model performance offline. To fully utilize the data, we expand the training dataset with sessions (specifically, the prev_items field) from validation dataset, as well as the test datasets of Task 1 and Task 3. In our experiments we found that such augmentation improved the model's online performance.

2.2 Single Models

In the first stage, our primary objective is to obtain a small, high-quality candidates set and construct features for the second stage. We employ three classes of models to achieve this goal: deep session-based recommendation models, text-based models, and statistic-based models. In the remainder of this section, we will discuss the details of each model type separately. It is important to note that all our single models are trained by locale, as this approach yields significantly better results compared to training a single model for all three locales.

2.2.1 Deep Session-Based Recommendation Models. Session-based recommendation has been explored for years. In this competition, we initially leverage multiple network architectures as backbone models to model sessions, followed by proposing various methods to enhance the backbone models. Thanks to RecStudio[1](a highly-modularized recommendation library) and UniRec(to be released), we can quickly implement models or directly use the models in the library.

The main models we adopted are:

(1) SASRec[7] is a transformer-based session-based recommendation model, using unidirectional self attention mechanism.

(2) LKNN is our modified version of SASRec to replace the multi-head attention module with a light convolution layer.

(3) Avghist employs mean pooling of all item embeddings within a session to generate the session representation.

(4) GRU4Rec[5] utilizes a GRU module for session modeling.

(5) NARM[8] enhances GRU4Rec by adding an attention mechanism to capture users' primary intent in the current session.

(6) SeqMLP concatenates representations of items in the session and uses MLP (Multi-Layer Perceptron) to model the session.

Regarding training details, Please refer to section A.1. Building upon existing models, we introduce the following three improvements based on the dataset's characteristics.

Feature augmentation. We adopt a direct and natural approach to incorporate categorical and numerical features into recommendation models. Specifically, we uniformly convert all feature fields into categorical types (for price, we use equal frequency buckets to divide it into 100 categories) and subsequently learn an embedding table for each feature field. We directly add embeddings of all feature fields to the item ID embedding to obtain the final item representations, which are used for both session modeling and item encoding.

Text augmentation. The dataset also includes substantial text information, such as product titles and descriptions. We leverage language models to enhance the performance of session-based recommendation models. Due to the significant training cost of jointly training the recommendation model and language model, we adopt a simple strategy. In detail, we train a text encoder to obtain pre-trained text embeddings for each item (refer to section 2.2.2 for details) and freeze the text embeddings. We then use a learnable MLP layer to map the text representation to the same vector space as the item ID embeddings derived from recommendation models. Similar to feature augmentation, we add the transformed text representation to the item ID embedding.

[1]https://github.com/ustcml/RecStudio

Continual training. As mentioned before, each session is divided into multiple slices for training. However, this creates a significant discrepancy in session length between model training and testing, potentially leading to suboptimal predictions. To alleviate this issue, we propose a continual training phase. To be specific, after normal training, we load model checkpoints and continue to train the model using only the next item field of each session as the training target. We find that this process is sensitive to the learning rate and typically requires a smaller learning rate than normal training. Nonetheless, it significantly improves the model's online performance.

2.2.2 Text-based Models. We employ bert-like[3] pretrained language models to obtain the hidden representations of items and BM25[12] models to model literal similarity between a session and an item. This serves two primary purposes: one is to enhance the performance of session-based models (refer to Section 2.2.1), and the other is to generate features for ranker training, as the text-based models themselves also function as content-based recommendation models. For bert-like pretrained language models, we have two implementations: user2item and item2item.

User2item. We utilize two language models (aka text encoders) with the same architecture to encode the representations of items and sessions, respectively. We regard the session and its corresponding next item field as a positive pair.

Item2item. We only use a single text encoder to encode item representations. We construct positive item pairs for training, considering every item pair in the same session as positives because we think they are highly relevant.

For training details, please refer to section A.2.

2.2.3 Statistic-based Models. In addition to the deep session-based recommendation models and text models that require training, we also employ statistic-based models to retrieve candidates and provide more direct and comprehensive features for the ranker, including classic ItemCF[11], UserCF, and heuristic co-occurrence-based models.

ItemCF. The core of the ItemCF is to recommend items similar to those visited in a user's session. Specifically, we construct an item similarity matrix based on the bipartite graph of sessions and items. The score for a new item depends on the sum of similarities between the items in the session and the new item.

UserCF. In contrast to ItemCF, the core of UserCF is to recommend items that similar users enjoy.

Co-occurrence-based models. The underlying principle of these models is based on the assumption that items appearing in the same session are highly related items. The similarity between two items is measured based on the frequency of their co-occurrence in the same session. We can retrieve the top n items with the highest co-occurrence frequency with all the items or the last item in a session as the recommendation list of the session.

2.3 Ranker

Through multiple single models, we can retrieve a candidate set for each session and provide various features for the ranker, enabling a more refined sorting of the candidates. We employ XGBoost[1] as our ranker (detailed in A.3). Next, we will elaborate on how we generate candidates for sessions and the features used in our solution.

2.3.1 Candidates Generation. The objective of candidates generation is to obtain a more comprehensive candidates set, improving the hit ratio of the candidates set compared to a single model's top 100 items set. Therefore, we use three single models from three different types: SASRec (with feature&text augmentation), xlm-RoBERTa, and a co-occurrence-based model, to retrieve 150 candidates for each session respectively. The final candidates for each session are obtained by deduplicating and merging the candidates retrieved by the three models. The hit ratio of the candidates set of the initial SASRec is 0.7235, which is improved to 0.7614 after merging the candidates from the other two models, with an average of 301 candidates per session in the final candidates set.

2.3.2 Feature Engineering. The features used in ranker can be categorized into three types: session features, item features, and session-item features.

Session features. The session features we use include the locale of the session and the average price of items in the session.

Item features. The item features we mainly use are the popularity of items, including *item frequency* and *next item frequency*. *Item frequency* refers to the number of occurrences of an item in data, measuring its popularity. *Next item frequency*, on the other hand, counts the occurrences of an item being the next item field in a session. The motivation for introducing the concept of next item frequency is the assumption that the behavior associated with the last item in a session is likely to be a purchase, which better reflects the popularity and quality of an item. During the final stages of the competition, *next item frequency* significantly contributed to our performance improvement, as demonstrated in the experimental results in Section 3.

Session-item features. Session-item features refer to the features generated by interactions between sessions and candidates, mainly involving the scores assigned to candidates by various single models. We utilize three different types of single models (mentioned in section 2.2) to provide diverse features to the ranker, including implicit feedback features learned by deep session-based models, text-based latent representation features and literal features, features obtained from statistics, and their fusion. Furthermore, we find that new features obtained by certain transformations on original scores also improved the performance of the ranker. We normalize the scores obtained from models that use softmax normalization during training by applying softmax function within each session's candidates set. For some other models like BM25, we normalize their scores using the min-max method.

3 RESULTS AND DISCUSSION

In this section, we present our main results and ablation studies for some crucial components.

3.1 Overall Performance

Table 2 presents the results of some single models and the ranker.

Single models. For deep session-based recommendation models, except for SeqMLP, we augment them with categorical, numerical, and text features. Among the models we used, SASRec achieved

Table 2: Main Results for Task 1

Method	Offline mrr@100	Online mrr@100
SeqMLP (ID only)	0.3003	–
Avghist	0.2823	–
GRU4Rec[5]	0.3164	–
NARM[8]	0.3178	–
LKNN	0.3343	–
SASRec[7]	0.3345	0.38665
xlm-RoBERTa-base[2]	0.1851	–
mbart-large-50[9]	0.2177	–
bert-base[3]	0.2238	–
ItemCF[11]	0.2600	–
UserCF	0.2600	–
XGBoost[1]	0.3599	0.40470
XGBoost ensemble	–	**0.40477**

the best performance, with an mrr@100 of 0.3345 on the offline validation dataset and 0.38665 on the leaderboard. Text-based models performed worse than session-based recommendation models, which we believe is because text-based models require significant training overhead and it is difficult for them to leverage a large number of negative samples and undergo sufficient training iterations. Additionally, without using item ID information, they struggle to effectively differentiate between similar items.

Ranker. Using about 100 features, a single ranker achieves an mrr@100 of 0.3599 on the validation dataset and 0.40470 on the leaderboard. It is worth noting that while text-based models and statistic-based models may not perform as well as deep session-based recommendation models, the features they generate significantly improve the performance of the ranker. On the last day of the competition, we ensembled three XGBoost models with different parameters, averaging their scores on the candidates. Finally, we achieved an mrr@100 of 0.40477 on the leaderboard.

3.2 Ablation Study

Table 3 and Table 4 present the ablation study for SASRec and the ranker respectively.

For SASRec, feature and text augmentation all resulted in performance improvements. Furthermore, after continual training, SASRec's performance experienced a significant enhancement.

In Table 4, XGBoost Base represents a basic ranker obtained in the initial stage of Phase 2 using partial features. After incorporating the *next item frequency*, the performance of the ranker significantly improved, with an mrr@100 0.40097 on the leaderboard. Subsequently, we further tuned the models (both single models and the ranker) and introduced more features to the ranker. Despite some features overlapping with those used by XGBoost Base, the performance of the ranker still experienced significant enhancement.

Table 3: Ablation Study for SASRec

Method	Offline mrr@100	Online mrr@100
SASRec	0.3245	0.38069
+ feature	0.3282	0.38239
+ text	0.3345	0.38665
+ continual training	**0.3374**	**0.39105**

Table 4: Ablation Study for Ranker

Method	Offline mrr@100	Online mrr@100
XGBoost Base	0.3514	0.39576
+ next item freq	0.3547	0.40097
+ all features & parameter tuning	0.3599	0.40470
+ ensemble	–	**0.40477**

4 CONCLUSION

In this paper, we introduce our pipeline for the KDD CUP 2023 Challenge. We adopt a classic two-stage approach. Specifically, in the first stage, we employ three different types of models for candidates retrieval and features generation, including deep session-based recommendation models, text-based models, and statistic-based models. We also introduce several effective methods to enhance the performance of traditional deep session-based recommendation models, consisting of content (feature&text) augmentation and continual training. In the second stage, we merge the scoring features of various models and hand-designed features to further rank the candidates set and ultimately ranks 3rd in the Task1 leaderboard.

REFERENCES

[1] 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*. 785–794.

[2] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116* (2019).

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[4] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821* (2021).

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

[6] Wei Jin, Haitao Mao, Zheng Li, Haoming Jiang, Chen Luo, Hongzhi Wen, Haoyu Han, Hanqing Lu, Zhengyang Wang, Ruirui Li, Zhen Li, Monica Xiao Cheng, Rahul Goutam, Haiyang Zhang, Karthik Subbian, Suhang Wang, Yizhou Sun, Jiliang Tang, Bing Yin, and Xianfeng Tang. 2023. Amazon-M2: A Multilingual Multi-locale Shopping Session Dataset for Recommendation and Text Generation. (2023).

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

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

[9] Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics* 8 (2020), 726–742.

[10] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[11] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[12] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and language models examined. In *Proceedings of the 19th Australasian Document Computing Symposium*. 58–65.

Table 5: Hyperparameters for XGBoost

Parameter	value
max depth	6
subsample	0.87
colsample bytree	0.47
learning rate	0.046
seed	12158
lambda	2.0
alpha	1.0
objective	rank:map

A APPENDIX FOR REPRODUCIBILITY

In this section, we mainly present some details of model training to enhance the reproducibility of our method. For more details, please refer to our repository: https://gitlab.aicrowd.com/CXL/unirec-task1-amazon-kddcup-2023.

A.1 Deep Session-Based Recommendation Models

During training, each session is divided into multiple slices, using every item except the first as the training target. Formally, given a user session $(i_1, i_2, i_3, ..., i_n)$, we use (i_1) to predict i_2, (i_1, i_2) to predict i_3, and so on, up to $(i_1, i_2, ..., i_{n-1})$ to predict i_n. We train models using the full softmax loss function on the entire item set, with dot product serving as the similarity score between users and items. We use 1 32GB NVIDIA Tesla V100 GPU for training a single model and the batch size is set to 2048.

A.2 Text-based Models

During training, we employ cross-entropy loss and in-batch negatives for contrastive learning[4], with the aim of pulling positive pairs together and pushing negative pairs apart. We also use dot product for similarity computation. We concatenate text features (such as title and description) of an item and input it into the model to obtain the item embedding. The text for a session is derived by concatenating text features of up to 5 historical items from the session. For the user2item setting, we train three models respectively (each model is trained by locale): facebook/mbart-large-50[9], xlm-RoBERTa-base[2], and bert-base[3]. For the item2item setting, we use distilbert-base-multilingual-cased[10]. It should be noted that bert-base models are a little bit different for each locale: we train bert-base-german-cased for DE, bert-base-japanese-whole-word-masking for JP and roberta-base for UK. All these models can be accessed through the HuggingFace library[2]. We obtain the final item and user representation for different models by using either mean pooling or the [CLS] token of the last layer output of the encoder. This approach is intended to increase the difference between models and produce a better ensemble result. For the base model, we use 3 NVIDIA RTX 3090 GPUs with 24GB each, and for each positive pair, we randomly select additional 5 negative examples. For the large model, we use 8 NVIDIA Tesla V100 GPUs with 32GB each.

A.3 XGBoost

We train our XGBoost on validation dataset through 5-fold cross-validation with about 100 features. We do not perform extensive hyperparameter tuning. We find that the main useful parameters are max depth, subsample, colsample bytree, and learning rate. All parameters are listed in Table 5.

[2]https://huggingface.co/