Candidate Reranking Solution using Many Variant Features for KDDCup2023, Team YAMALEX Solution

Hiroki Yamamoto
Acroquest Technology Co., Ltd.
Yokohama, Kanagawa, Japan
yamamoto@acroquest.co.jp

Takashi Sasaki
Acroquest Technology Co., Ltd.
Yokohama, Kanagawa, Japan
sasaki@acroquest.co.jp

Shin Higuchi
Acroquest Technology Co., Ltd.
Yokohama, Kanagawa, Japan
higuchi@acroquest.co.jp

Tomonori Fujiwara
Acroquest Technology Co., Ltd.
Yokohama, Kanagawa, Japan
fujiwara@acroquest.co.jp

Shun Yoshioka
Acroquest Technology Co., Ltd.
Yokohama, Kanagawa, Japan
s_yoshioka@acroquest.co.jp

ABSTRACT

It is essential for e-commerce stores to model customers' shopping intentions which directly lead to user experience and engagement. Although there is an increasing interest in utilizing session data to predict what the user will purchase next, there has not been many studies on session-based recommendation using real-world multilingual and imbalanced scenarios. In the Amazon KDD Cup 2023, Amazon presented the "Multilingual Shopping Session Dataset" with millions of user sessions from six different locales, namely: English, German, Japanese, French, Italian, and Spanish. The dataset introduces imbalance by having fewer data for French, Italian, and Spanish compared to other locales. In this paper, we present our approach to this challenge using two stage approach to generate the candidates.

ACM Reference Format:
Hiroki Yamamoto, Takashi Sasaki, Shin Higuchi, Tomonori Fujiwara, and Shun Yoshioka. 2023. Candidate Reranking Solution using Many Variant Features for KDDCup2023, Team YAMALEX Solution. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

For e-commerce stores, giving personalized recommendations is for user experience and engagement. However, there has not been enough studies that used real-world session data with imbalancement to test their session-based recommendation system. The Amazon KDD Cup 2023[3], organized by Amazon, introduced a dataset and standard evaluation metrics to be used to assess model performance. The competition is expected to provide practical solutions that benefit worldwide customers. The competition has three tasks. The three tasks are set to tackle recommendation for imbalanced data and cold-start data. The dataset contains millions of user sessions from six different locales. The dataset is imbalanced among the locales, French, Italian, and Spanish having less data compared to other locales, English, German and Japanese. The dataset is split into three splits: train, phase-1 test, and phase-2 test. The same train set is used for all the tasks. The test dataset is prepared specifically for each task objectives. English, German, and Japanese test data are used for Task1; French, Italian, and Spanish data for Task2. The test data for Task3 contains products that do no appear in the training set. For Tasks1 and 2, participants should predict 100 product IDs in descending confidence order. The output is evaluated using Mean Reciprocal Rank. For Task3, participants need to generate one title for each session. The generated title is evaluated using BLUE. The structure of this paper is as follows. In Section 2 explains the overview of our systems. Sections 3 and 4 explain the details of each stage of our two-stage approach: candidate generation and reranking, respectively. This paper is concluded in Section 5.

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 OVERVIEW

We built a Recommend system that generates candidates and then reranks them in detail to produce the final recommendation. Fig.1 is our system overview. Our candidate generation method use many variant features, simple candidate generation, co-matrix, BPR, ProNE, GRU, amd MLP. Reranking that comes after candidate generation uses LightGBM and many useful features.

3 CANDIDATE GENERATION

3.1 Simple Method

In the three simple methods for candidate generation.First, we took the top 100 products that were most frequently purchased in the given locale. Second, given the last product in the session, the top 100 products that were purchased after the last product are added to the candidates. This gives the locale-wide recommendations as the candidates, as in, if it's popular in the locale, users might want it. Finally, we add same attribute items, same brand, same author, same model that count within 100. it's indicate similar item.

3.2 Co-Matrix

A CoMatrix represents the frequency or strength of co-occurrence between elements in a dataset. In the context of recommendation systems, it captures the co-occurrence patterns of items within user interactions. Each row and column correspond to unique items, and the matrix cells contain the count or strength of co-occurrence

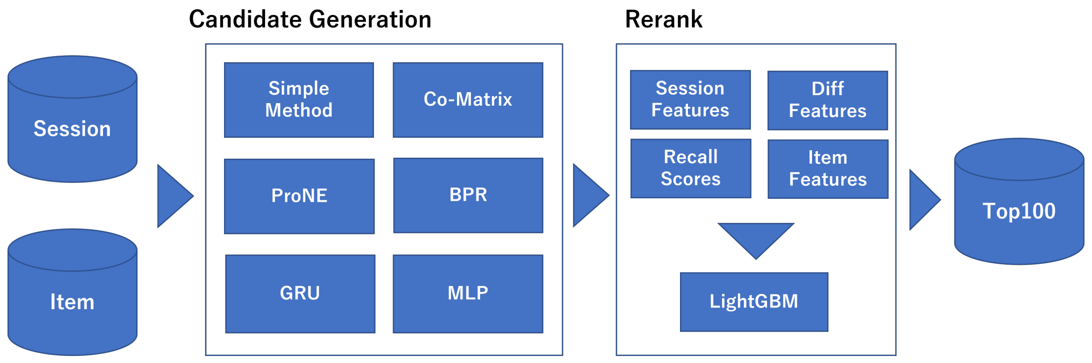


Figure 1: Overview of our approach

between item pairs. CoMatrices provide insights into relationships and associations between items based on their co-occurrence patterns. In our approach the strength is calculated based on count and inverse of distance. I use two type of Co-Matrix score, last5 and all items in session.

3.3 ProNE

ProNE (Proximity Network Embedding) is an algorithm for network embedding that captures structural information by mapping nodes into a low-dimensional vector space[7]. It estimates node proximity based on network connectivity and co-occurrence patterns. ProNE constructs a graph representation of the network and applies an iterative optimization framework to generate meaningful node representations. The algorithm leverages both topological and contextual information to preserve proximity between nodes. It has been shown effective in tasks such as node classification, link prediction, and community detection. ProNE provides a scalable and efficient approach for capturing network structure and proximity relationships. Comparing to other graph embedding algorithms such as Node2Vec[2], ProNE is computationally inexpensive and fit for large scale data. ProNE estimates item proximity by analyzing both network connectivity and co-occurrence patterns. It considers direct connections between items and their co-occurrence in local neighborhoods. The algorithm assigns higher proximity values to items with stronger connectivity and co-occurrence relationships.

Graph structure is well-suited for session-based recommendation because it can express co-occurrence and sequential dependencies with its nodes and edges.

In our approach, we first created bi-directional graph by connecting consecutive purchases in a session. Then, ProNE was used to generate a embedded version of the graphs.Fig.2 is graph expression. The generated graphs can be used to predict the next item given a session leading to that point. Getting embedding item vectors from ProNE, after we find a top similarities from last item and other items.

3.4 BPR

BPR (Bayesian Personalized Ranking)[6] is a collaborative filtering algorithm designed for personalized ranking in recommender systems. It addresses the challenge of item ranking by focusing on pairwise preferences of users. Rather than predicting explicit

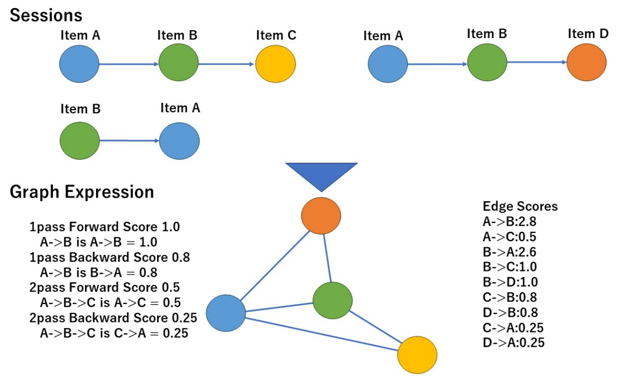


Figure 2: Overview of prone

ratings, BPR learns to model the relative ranking of items based on user preferences. It employs matrix factorization to capture latent factors for users and items. During training, BPR constructs pairs of items using positive and negative user interactions and optimizes model parameters using stochastic gradient descent. BPR's Bayesian framework maximizes the likelihood of ranking positive interactions higher than negative ones. This algorithm is particularly suited for implicit feedback data, where user preferences are inferred from behavior. By considering pairwise preferences and offering scalable recommendations, BPR has been successfully applied across domains such as e-commerce, social media, and content streaming platforms.

3.5 GRU

Neural networks are also employed to generate recommendation candidates. The Gated Recurrent Unit (GRU) is a variant of the Recurrent Neural Network (RNN) architecture designed to capture long-term dependencies and address the vanishing gradient problem[1]. It utilizes gating mechanisms to control information flow. The update gate selectively integrates new information, while the reset gate determines which past information to forget. The candidate activation function combines input and reset gate-applied previous hidden state to generate a candidate value for the current hidden state. The GRU effectively captures long-term dependencies and exhibits computational efficiency with fewer parameters compared to other RNN variants like LSTM. It finds applications in natural language processing, speech recognition, machine translation, and time series analysis. The GRU's gating mechanisms enable it to retain relevant information, resulting in improved accuracy and pattern recognition. In summary, the GRU is a powerful architecture for modeling sequential data, providing a solution to the vanishing gradient problem and allowing for the capture of long-term dependencies. For each models, TF-IDF score and embedding of features of a product is used as their input. Input is TF-IDF and each attributes. calculates the probability of the item purchased after the last/second-last product in the history.

In our approach, we used two types of input to train and predict with the model. our architecture show Fig.3 Type1 is used for Task1. The model has three dense layers each connected to GRU unit. The output of GRU is fed into another dense layer to get the result. The first dense layers get inputs which are TF-IDF vectors and embedded features.

First, Item convert to dense item vectors, Title TF-IDF feature is used dense layer, other features are used embedding layers, then concat these features. it input GRU architecture. GRU output apply dense layer. it's output is next item probability.

Type2 is used for Task2. it's very similar. The basic structure is the same with Type1. However, it uses TF-IDF output that is processed with SVD for more efficiency. because Type1 vocabularies are very wider, we can't put on memory.

Moreover, we apply some data augmentation in session when training GRU. it method help model regularization.

(1) swap last and second
(2) swap last and target, last two and target
(3) target swap, target t item and target t + 1
(4) item attribute dropout, our apply probability mask
(5) truncated item(before/after)
(6) item sequence shuffle

3.6 MLP

In MLP, vectors for TF-IDF and embedding of features are used to train and predict with MLP. We have trained MLP models, MLP inputs are second-to-last and last item in the session. both MLP parameters are same. The outputs from the two models are aggregated using concat and dense layers. also we apply data augmentation to MLP. it also help regularization.

4 RERANKING

After generating several recommendation candidates, we performed reranking. This phase is crucial because there are no relationships between scores of each candidate generation methods. Getting high score with ProNE does not necessarily mean it has overall high score. The reranking phase solves this problem by reordering the generated candidates using the candidate generation method scores and other additional features so that the final score reflects the overall score. Our approach uses LightGBM[4] to perform the reranking. In this phase, We use these features.

(1) Session features: session aggregation, e.g. mean price, session size, max price, mean price.
(2) Diff Features: difference between last/last-two and candidate.
(3) Recall Scores: recall score calculated in recall methods and rank of score in session.
(4) Item Features: candidate item price, candidate attribute count encoding.

We used aggregation session statistics. session size, and aggregation price(max, mean) as session features In addition, last5/10 percent of candidate attributes were used.

Diff features indicate similarity between last/last-two product and the candidate. Our team use levenstain distance, jaro winkler, title distance in title. word2vec[5], flag of same attribute, and difference price of last/last-two product and aggregation session price statistics.

Recall Scores are recall method scores, namely, co-matrix score, MLP/GRU output probability, ProNE similarity, and BPR Score. Item features use price and count encoding of item attributes. We used label encoding attribute when task2. However, the score went down when we used it for task1.

Session have about 300 candidates per session on avarage. Because we cannot put all features on memory, we applies negative down sampling. Negative down sampling is a method where we pickup some negative items in session. our negative down-sampling rate for Task1 is 7% and 70% for Task2.

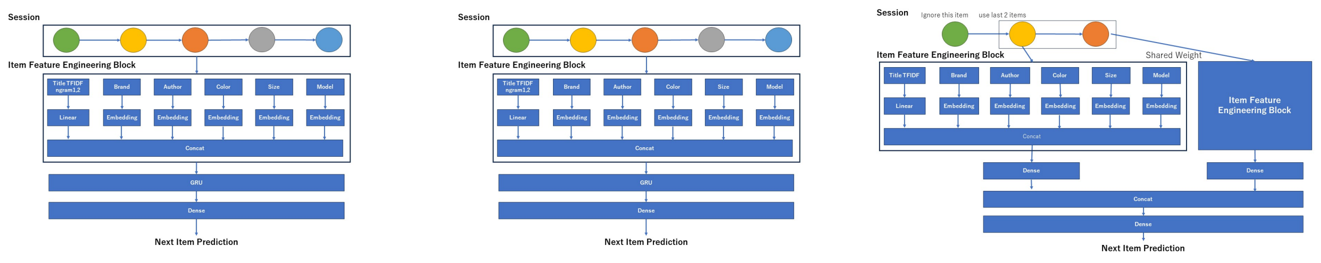
Finally, our team achieved 15th in Task1(Table.1) and 9th in Task2(Table.2) in KDDCup2023.

Rank	Team	Score
1	NVIDIA-Merlin	0.41188
2	MGTV-REC	0.41170
3	unirec	0.40477
4	gpt_bot	0.40476
5	LeaderboardCar	0.40339
6	AIDA	0.40317
7	piggy-po	0.40476
8	iCanary	0.39651
9	wxd1995	0.39592
10	xuy	0.39566
15	**[Acroquest]YAMALEX(ours)**	**0.38957**

Table 1: Comparison of task1 scores

5 CONCLUSION

In the research presented in this paper, we have employed a two-stage approach that includes a candidate generation process and a re-ranking framework. In the initial candidate generation phase, we incorporated a multitude of algorithms encompassing graph-based techniques, co-matrix models, BPR, neural networks, and a variety of simple features. Following this, the candidates were subjected to a re-ranking process. For this re-ranking, we used the LightGBM model, leveraging the differences between the final and candidate items' characteristics (such as word associations, pricing, etc.), as well as the scores obtained during the candidate generation stage. Our framework achieved commendable placements in the tasks, securing the 15th place in Task 1 and the 9th place in Task 2.



(a) Type1 (b) Type2 (c) MLP

Figure 3: Overview of Neural Network

Rank	Team	Score
1	NVIDIA-Merlin	0.46845
2	MGTV-REC	0.46758
3	gpt_bot	0.46011
4	AIDA	0.45047
5	piggy-po	0.44914
6	chimuichimu	0.44798
7	iCanary	0.44747
8	QDU	0.44618
9	**[Acroquest]YAMALEX(ours)**	**0.44380**
10	DX2	0.44101

Table 2: Comparison of task2 scores

REFERENCES

[1] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[2] Grover, A., and Leskovec, J. node2vec: Scalable feature learning for networks. *CoRR abs/1607.00653* (2016).

[3] Jin, W., Mao, H., Li, Z., Jiang, H., Luo, C., Wen, H., Han, H., Lu, H., Wang, Z., Li, R., Li, Z., Cheng, M. X., Goutam, R., Zhang, H., Subbian, K., Wang, S., Sun, Y., Tang, J., Yin, B., and Tang, X. Amazon-m2: A multilingual multi-locale shopping session dataset for recommendation and text generation.

[4] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems* (2017), I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc.

[5] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems* (2013), C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26, Curran Associates, Inc.

[6] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. BPR: bayesian personalized ranking from implicit feedback. *CoRR abs/1205.2618* (2012).

[7] Zhang, J., Dong, Y., Wang, Y., Tang, J., and Ding, M. Prone: Fast and scalable network representation learning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19* (7 2019), International Joint Conferences on Artificial Intelligence Organization, pp. 4278–4284.