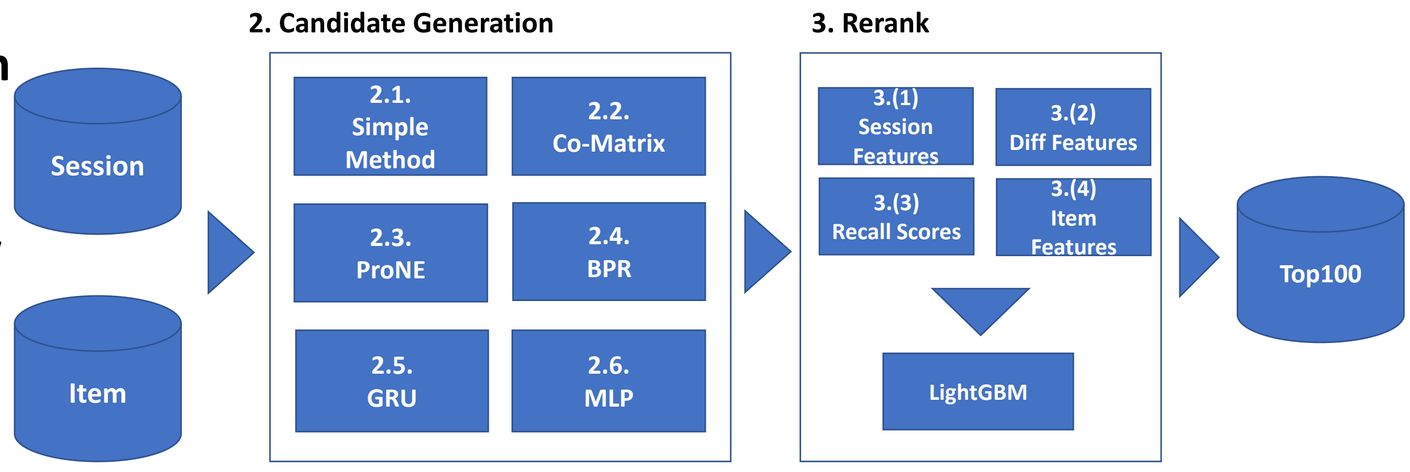
# Candidate Reranking Solution using Many Variant Features for KDDCup2023, Team YAMALEX Solution

Hiroki Yamamoto, Takashi Sasaki, Shin Higuchi, Tomonori Fujiwara, Shun Yoshioka
Acroquest Technology Co., Ltd.

## 1. Overview

**We built a Recommend system that generates candidates and then reranks them in detail to produce the final recommendation.**
**Our candidate generation method use many variant features, simple method, co-matrix, ProNE, BPR, GRU, and MLP. Reranking uses LightGBM and many feature.**




## 2. Candidate Generation

### 2.1. Simple Method

We used three simple methods:

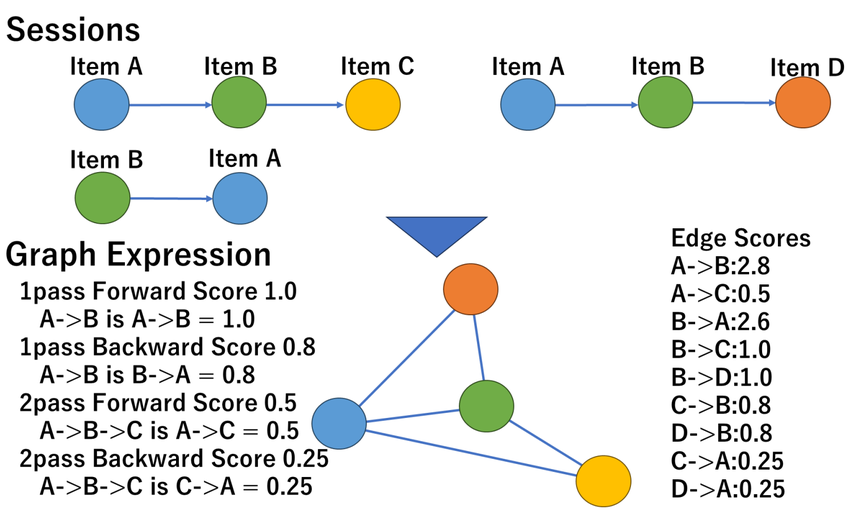
(**1**) the top 100 most frequently purchased in the given locale.
(**2**) given the last product in the session, the top 100 products that were purchased after the last product
(**3**) 100 items having same attribute, brand, author, or model with the last.

### 2.2. Co-Matrix

**A Co-Matrix** illustrates the co-occurrence patterns of items in user interactions, detailing the count and intensity of co-occurrences **between item pairs, thereby offering insights into item relationships.** This intensity is measured using the count and inverse of distance. Two kinds of Co-Matrix scores are employed: 'last5' and 'all items in session'.

### 2.3. ProNE

**ProNE**, an affordable algorithm for large scale network embedding, **analyzes both network connectivity and co-occurrence patterns**. It was used to embed bi-directional graphs created from sequential purchases in a session to estimate the nearest items from last item.

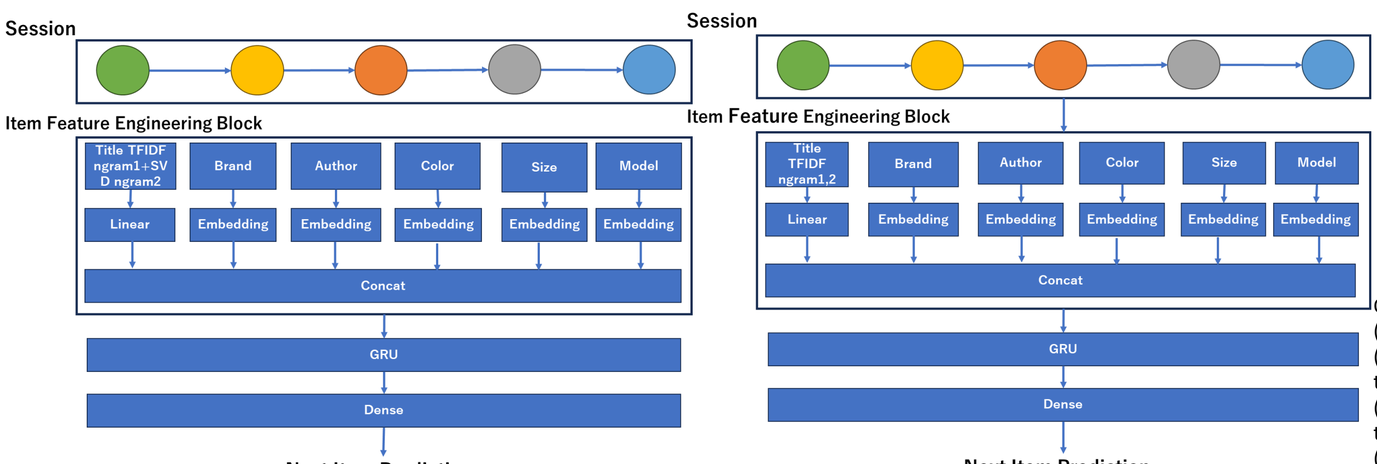



### 2.4. BPR

**BPR (Bayesian Personalized Ranking)** is a collaborative filtering algorithm designed for personalized ranking in recommender Systems. **BPR learns to model the relative ranking of items based on user preferences.** This algorithm is particularly suited for implicit feedback data, where user preferences are inferred from behavior.

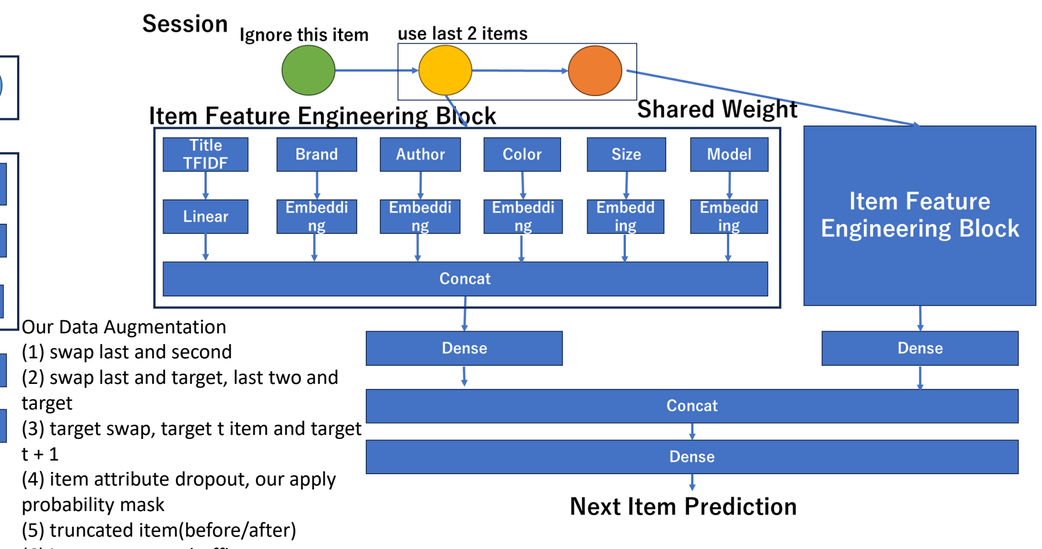
### 2.5. GRU

**Gated Recurrent Unit (GRU),** a Recurrent Neural Network (RNN) variant, mitigates the vanishing gradient problem and captures long-term dependencies. It uses a product's TF-IDF score and feature embeddings as input, predicting the probability of the next purchase.
**Type1** uses three dense layers linked to a GRU unit, with the GRU's output fed into another dense layer for next-item probability.
**Type2** is like Type1 but uses SVD-processed TF-IDF output for enhanced memory efficiency.




### 2.6. MLP

**MLP**, vectors for TF-IDF and embedding of features are used for training and prediction. MLP inputs are second-to-last and last item in the session. Both MLP parameters are shared. The outputs from the two models are aggregated using concat and dense layers. Also, we apply data augmentation to MLP for regularization.




Our Data Augmentation
(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. Reranking

After generating several recommendation candidates, **we performed reranking with LightGBM.**
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, namely: Levenshtein distance, Jaro-Winkler, title distance in title. word2vec, flag of same attribute, and difference in price.
**(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, item attributes

Session have about 300 candidates per session on average, but when training re-ranking model, we cannot put all features on memory. Therefore, we applied negative down sampling, a method where we omit some negative items in session. Our negative down-sampling rate for Task1 is 7% and 70% for Task2.

| 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**

| 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**