A Recommendation Approach to the Cold Start Problem

——Based on Implicit Feedback from Instacart Orders

Group 13

Contributors

Huang Lishan, Lyu Xueyuan, Yang Qing, Yuan Ailin, Li Yushan

Agenda

Recommendation System (RS) Introduction

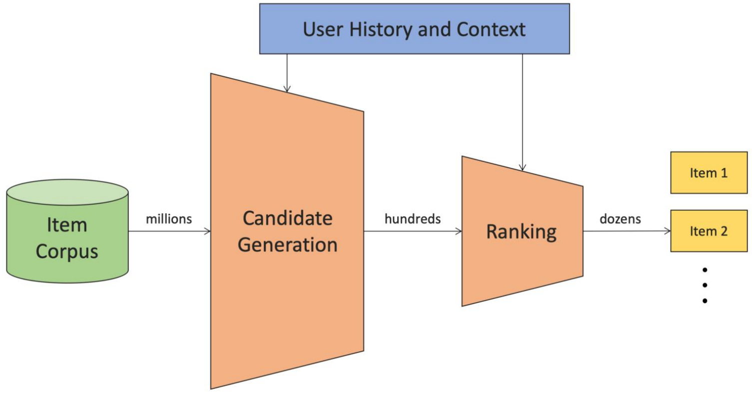
Recommendation System Introduction

Problem Statement: Implicit Feedback, Cold Start Problem

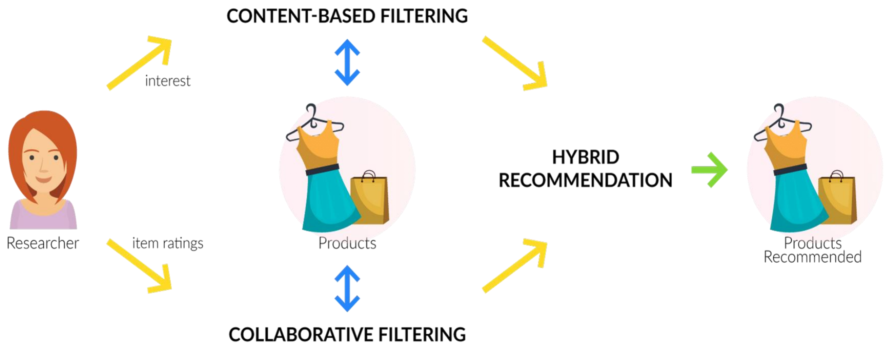
01 Recommendation System Introduction

- How AI shopping carts make smart recommendations?

- A Comprehensive RS: Candidate Generation——Ranking



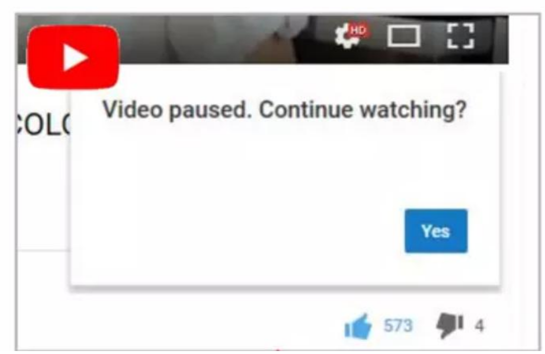
- Types of Recommendation Systems



01 Problem Statement——Implicit Feedback

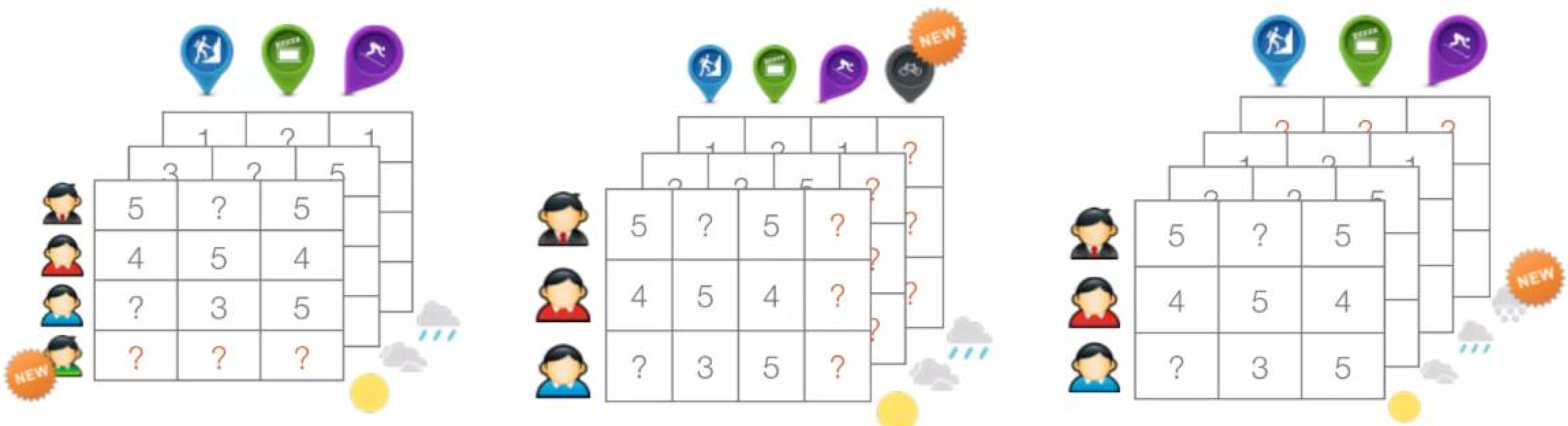
- **Explicit feedback:** Users' rating or likes on items (rate 1 to 5, like/dislike...). But it is not always available.

- **Implicit feedback** indirectly reflects opinions through behavior.
- **Examples:**purchase history, browsing history, search patterns
- **Implicit feedback** is much more abundant, but also more **difficult** to use
- **Difficulties : Noise/No negative feedback/ Evaluation metric**
    - **Example:**TV shows, r_{ui} = how many times u fully watched show i
    - **r_{ui}= 0.5** →user (got bored?) stopped watching at half show
    - **r_{ui}=2** → user (loved it? fell asleep and played in loop?) watched the show twice



01 Problem Statement——Cold Start Problem

- **New user problem:** How do you recommend to a new user?
- **New item problem:** How do you recommend a new item with no ratings?
- **New context problem:** How do you recommend in a new context?



- **Cold Start Problem → missing information → getting information from users and items**

- **Difficult to get more information → using other methods to solve it**

Recommendation System Methodology

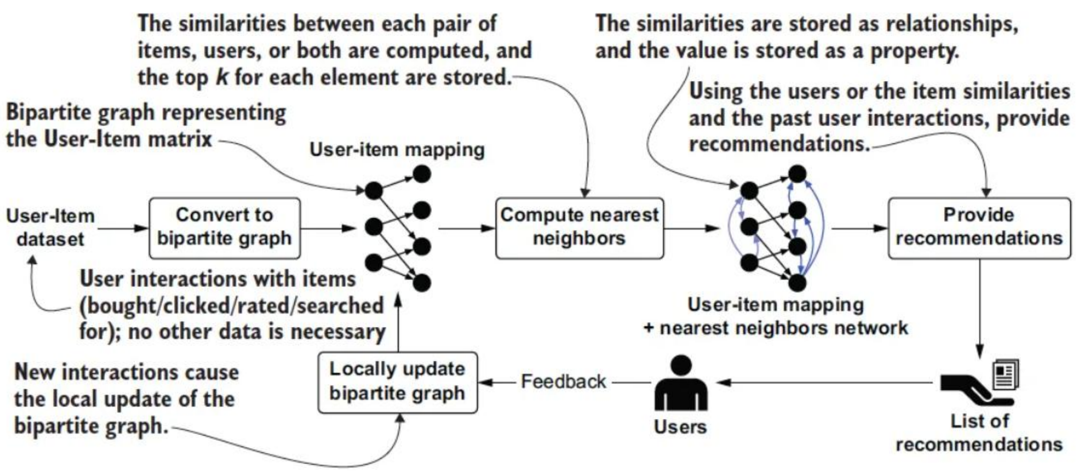
Model1: Item-based Collaborative Filtering

Model2: User-based Collaborative Filtering

Model3: LightFM——a Hybrid Recommendation System

Model4: Sentence Bert+FAISS

02 Model1: Item-based Collaborative Filtering



Item-based CF recommends items similar to those a user has purchased, assuming similar items match user preferences.

product_name	correlation	common_users
Soda	1.000000	8000
Organic Chives	0.915934	20
Organic Fresh Basil	0.888540	16
Taboule Salad	0.856117	16
Mexican Casserole Bowl	0.841896	31
Moroccan Mint Herbal Tea	0.808685	17

Workflow of item-based CF:

- **User-Item Matrix:**

  Built from implicit feedback (purchase counts)

- **Item-Based Similarity:**

  Calculate pairwise item similarity using Pearson correlation

- **Weighted Scoring:**

  For each user, score unpurchased items by the weighted sum of similar items they've bought.

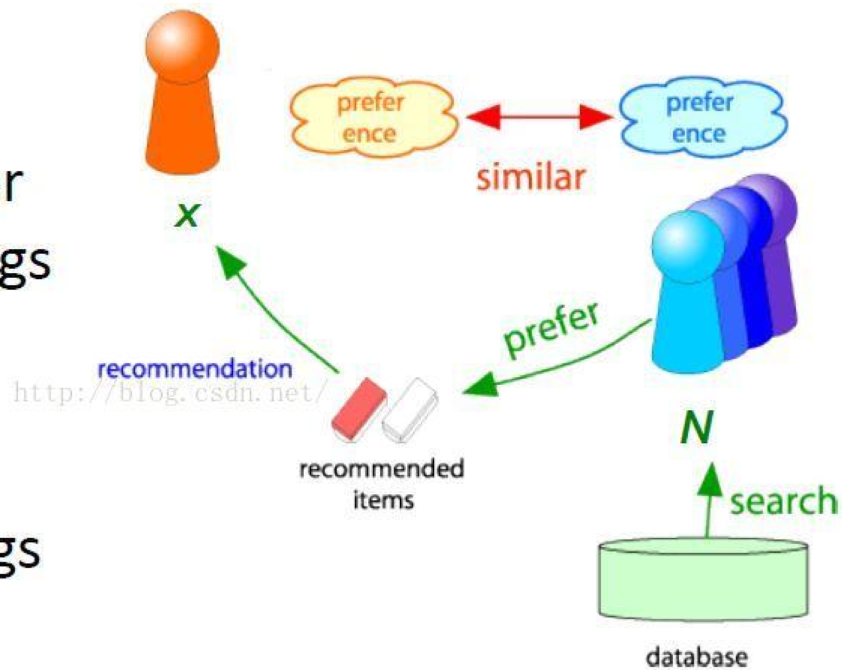
- **Top-N Recommendation**:

  Recommend top-N highest-scored items not yet purchased.

02 Model2: User-based Collaborative Filtering

Collaborative filtering (CF) is a memory-based recommendation algorithm that predicts a user's preferences for items by finding and analyzing the preferences of other users who are similar in behaviors.

- Consider user ***x***
- Find set ***N*** of other users whose ratings are "**similar**" to ***x***'s ratings
- Estimate ***x***'s ratings based on ratings of users in ***N***



02 Model2: User-based Collaborative Filtering

Workflow of CF:

- Look for “similar” users with the active user
- Aggregate the preferences of active user’s neighbors to identify the set of items to be recommended

Filtering

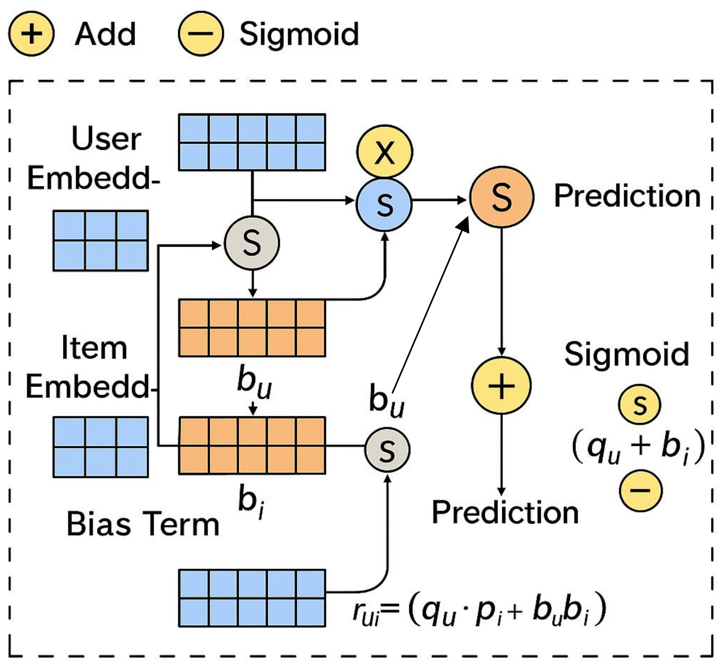
Construct user-item interaction matrix

Pearson correlation and vector cosine based similarity

Collaboration

Weighted aggregation of k-nearest users’ preference vectors

02 Model3: LightFM——a Hybrid RS



How LightFM solve cold start problem?

- **Using user, item and interaction features**
- LightFM performs **at least as well as pure content-based models**, outperforming them when either **collaborative information or user features** are included

The latent representation of user u and item i are given by the sum of its features latent vectors:

$$\boldsymbol{q}_u = \sum_{j \in f_u} \boldsymbol{e} \qquad \boldsymbol{p}_i = \sum_{j \in f_i} \boldsymbol{e}_j^I$$

The bias term for user u and item i are given by the sum of the features biases:

$$b_u = \sum_{j \in f_u} b_j^U \qquad b_i = \sum_{j \in f_i} b_j^I$$

The model s prediction for user u and item i are:

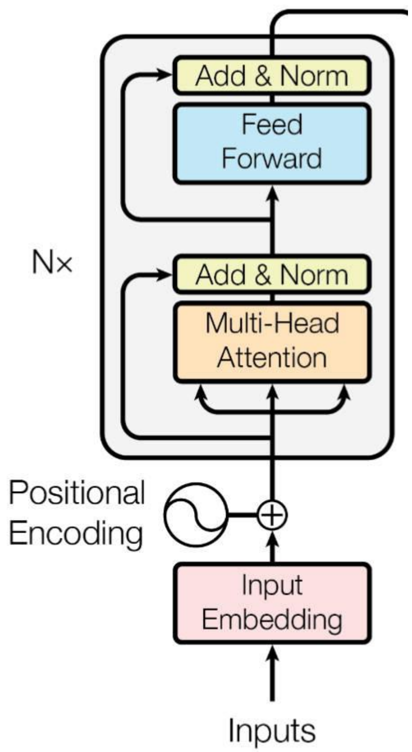
$$\widehat{r}_{ui} = f\left(\boldsymbol{q}_u \cdot \boldsymbol{p}_i + b_u + b_i\right)$$

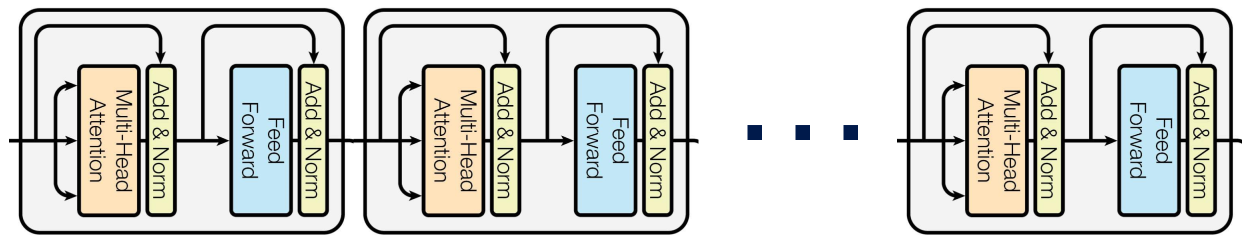
The likelihood is given by

$$L\left(\boldsymbol{e}^U, \boldsymbol{e}^I, \boldsymbol{b}^U, \boldsymbol{b}^I\right) = \prod_{(u,i) \in S^+} \widehat{r}_{ui} \times \prod_{(u,i) \in S^-} \left(1 - \widehat{r}_{ui}\right)$$

02 Model4: Sentence Bert+FAISS

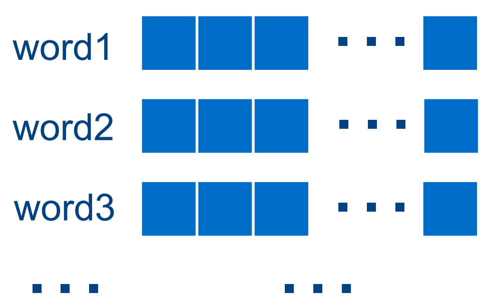
Encoder from Transformer





Bert

Bidirectional Encoder Representation from Transformer



02 Model4: Sentence Bert+FAISS

- FAISS (Facebook AI Similarity Search)
  - Goal

  Find the most similar K vectors in a large amount of vector data.
  - Process

  Create indexes and search.

  Methods of creating indexes: Flat, Locality Sensitive Hashing (LSH), Hierarchical Navigable Small World (HNSW), Inverted File Index (IVF)...

Instacart Data Preprocessing

Dataset Introduction

Exploratory Data Analysis

Feature Engineering

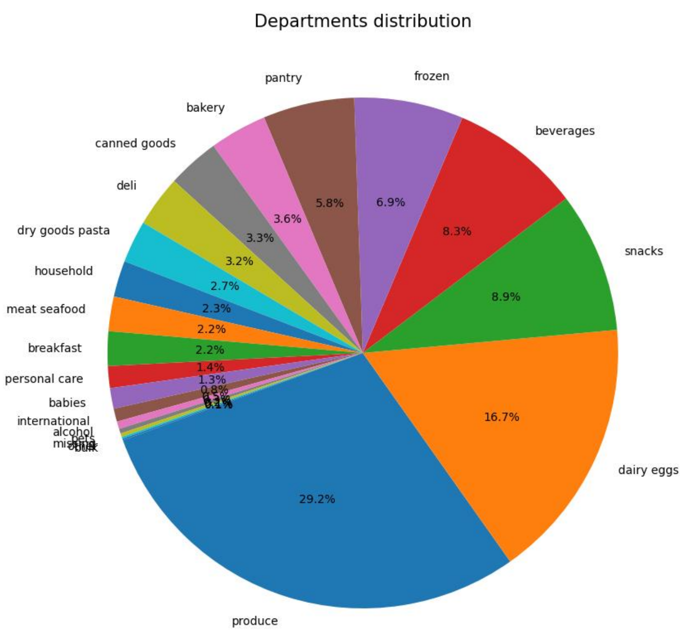
Association Rules

03 Dataset Introduction and EDA

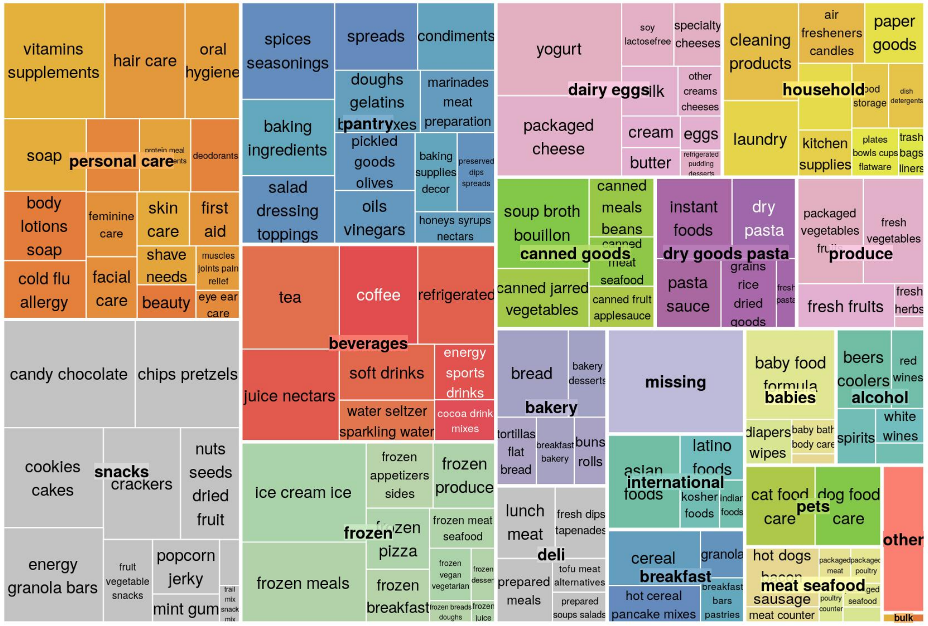
Instacart: Groceries delivered from your favorite stores

Real Shopping Behavior Records

- **6 tables from Kaggle:** orders, train, priori, products etc.
- **Data:** 21 departments, 134 aisles, 49688 products, 32434489 orders
- **Implicit feedback:** add to cart, reorder, days since prior order etc.



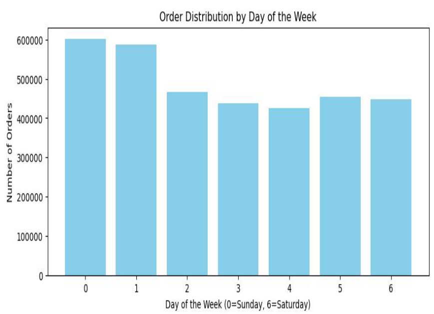
department

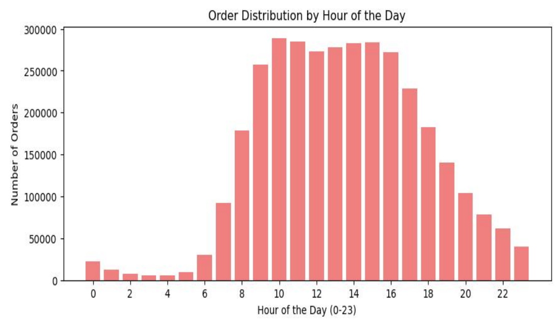


products

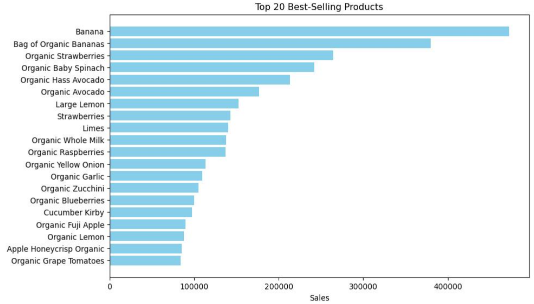
03 Exploratory Data Analysis (EDA)

- Peak period: between 10 am and 4 pm; Sunday.
- Low Peak period:between 2 am and 6 am; Wednesday.

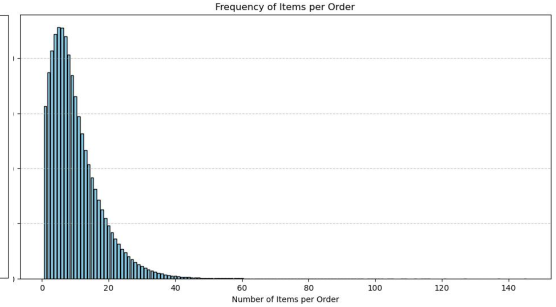




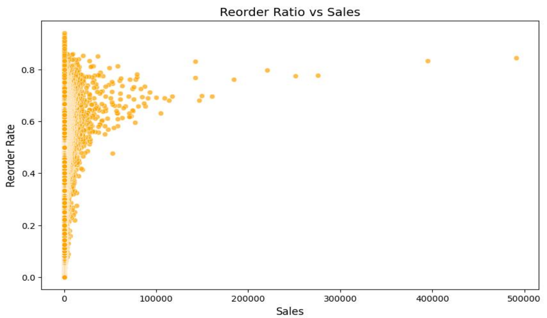
Best-selling: fruits and vegetables



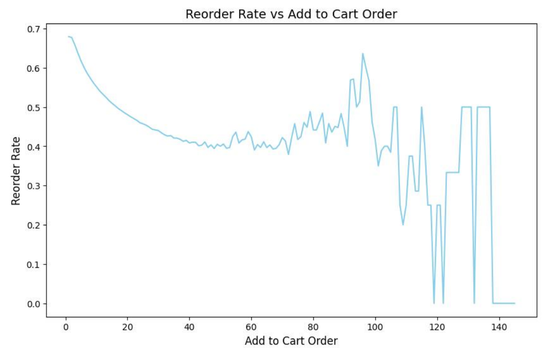
Most of the people buy 1-15 items in an order



- The lower the add-to-cart-order, The higher the reorder percentage.



- Many people try different product once and they do not reorder again. Some user buy certain products regularly.



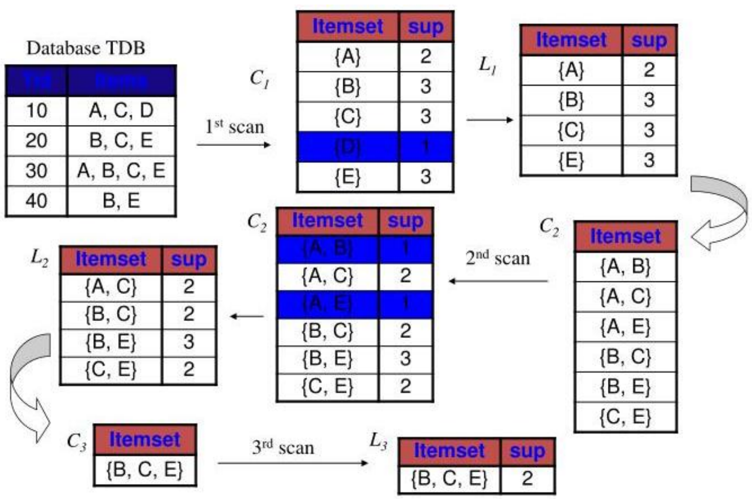
03 Feature Engineering——Feature Construction

6 original features → 28 new features

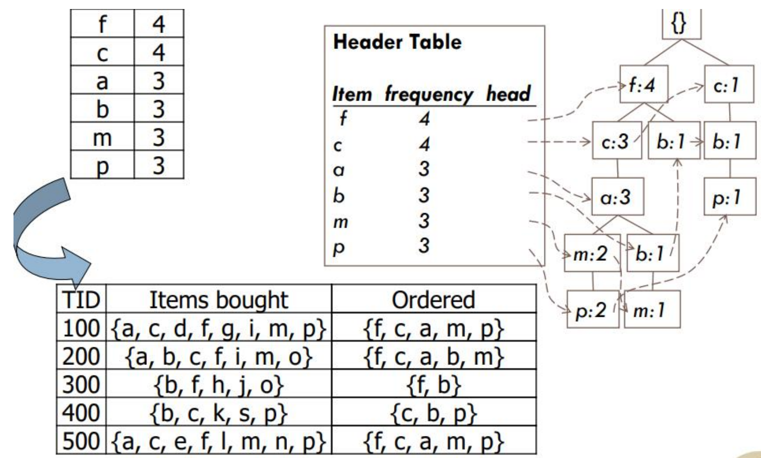
- **Product-Level Features**
  - Popularity & Visibility → Product's average add-to-cart-order
  - Reorder Behavior → Reorder percentage of a product
  - Aisle and department level features → Total orders of an Aisle/ a Department
- **User-Level Features**
  - Habits → User's average and std day-of-week of order
  - Engagement & Diversity → Total orders by a user
  - Reordering Tendency → User's overall reorder percentage
- **User-Product-Level Features**
  - Personal Preference Strength → Average days between purchases of the product
  - Recent Interactions → Whether reordered in the last 3 orders

03 Association Rules

Apriori Algorithm



FP-Growth



- **Scanning the entire dataset**
- **wastes a lot of time**

- **Recursively mining frequent itemsets**
- **Avoids multiple database scans**

Examples: **s**{*Boneless Skinless Chicken Breasts* '} →{*Banana', 'Honeycrisp Apple', 'Organic Avocado'*}

Recommendation Model Construction

Model1&2: Item or User Based Collaborative Filtering

Model3: LightFM

Model4: Bert+FAISS

04

Model1&2: Item or User Based Collaborative Filtering

- Optimization Techniques

○ Data Filtering

Stratified sampling to improve robustness and diversity

○ Interaction matrix

Frequency-weighted, Time-decay weighted to combine more personalized preferences

○ Similarity Capturing

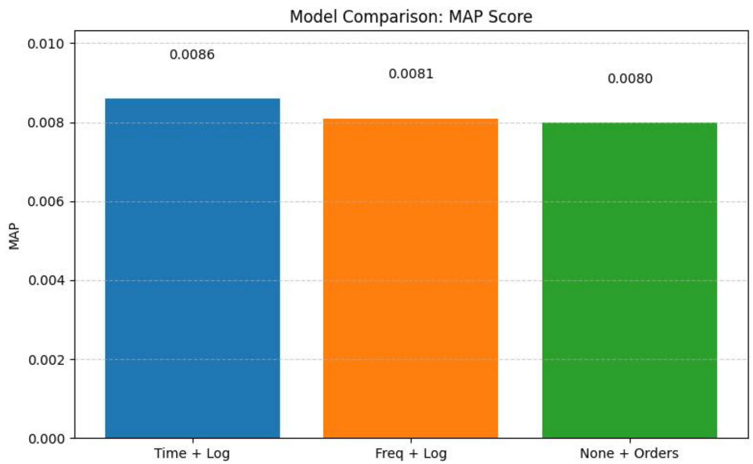
Singular Value Decomposition(SVD) to remove noise and sparsity

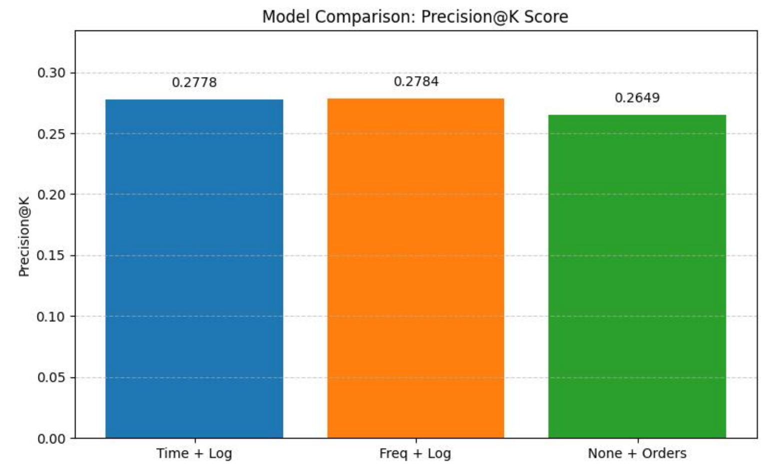
○ Recommenders

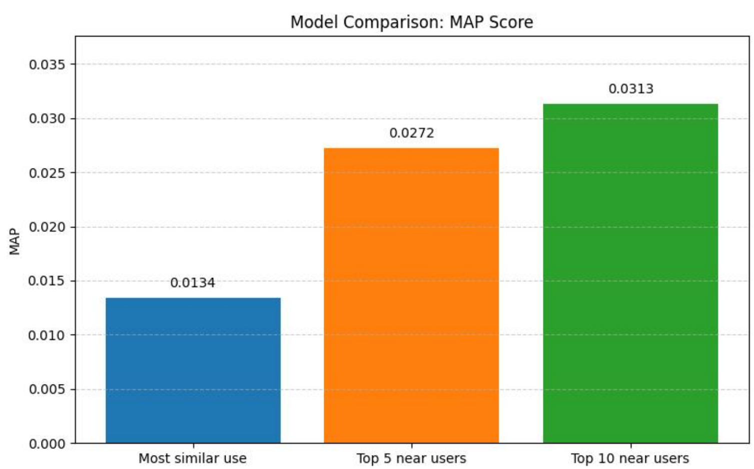
K-nearest weighted average to enhance diversity

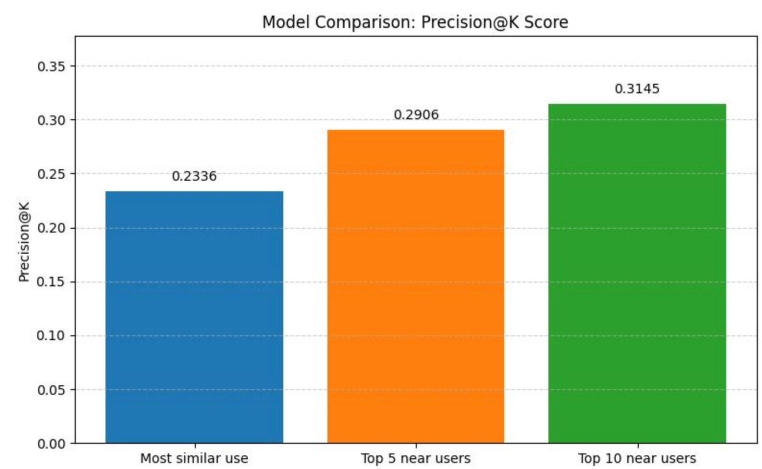
04 Model1&2: Item or User Based Collaborative Filtering

• Optimization Outcomes









04 Model3: LightFM

1 LightFM with metadata (tags)

Interaction matrix : user_id, item_id ,User–Product Purchase Frequency

2 LightFM with features

- **Feature engineering：** Binning to discretize, PCA to reduce dimensionality
- **Step-by-Step Feature Integration：** + All User Features, + All Item Features, + All User & Item Features
- **Using Grid Search for Hyperparameter Tuning**

3 Model results

model	Precision@5	Precision@10	Precision@20
Lightfm(tags)	**0.95**	**0.95**	**0.93**
Lightfm(tags + ids)	**0.06**	**0.06**	**0.03**
Lightfm(tags + uds)	**0.07**	**0.06**	**0.01**
Lightfm(tags + about)	**0.05**	**0.05**	**0.01**

04 Model4: Sentence Bert+FAISS

- Fine-tune SBert
  - Randomly select 80% of the source data as training data.
  - Construct positive and negative sample pairs for input. Aisle same for positive sample pairs and different for negative sample pairs.
  - Set cosine similarity loss as the loss function.

```
model.fit(
        train_objectives=[(train_dataloader, train_loss)],
        epochs=3,
        warmup_steps=100,
        output_path=model_save_path    # Save path of the fine-tuned model
)
end = time.time()
print('Running time is {:.2f} seconds.'.format(end-start))
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be re
Using the `WANDB_DISABLED` environment variable is deprecated and will be re
```

[627/627 2:20:20, Epoch 3/3]

Step	Training Loss
500	0.143100

04 Model4: Sentence Bert+FAISS

- How we use FAISS?
  - Create Indexes
  - Similarity Search

```
# Create FAISS indexes
dimension = product_embeddings.shape[1]   # Dimensions of embedded vectors
index = faiss.IndexFlatL2(dimension)   # L2 distance (Euclidean distance)

# Adding vectors to indexes
index.add(product_embeddings)

# Save index and product ID mapping
faiss.write_index(index, "product_index.faiss")
np.save("product_ids.npy", df_products.product_id.values)

# Loading Indexes and Product IDs
index = faiss.read_index("product_index.faiss")
product_ids = np.load("product_ids.npy")
```

Experimental Results

- Precision@K
- Item-based CF
- User-based CF: stratified + svd, all optimizations
- LightFM: tags, tags + ids, tags + uds, tags + about
- SBert+FAISS

05 Experimental Results

model	Precision@5	Precision@10	Precision@20
Item-based CF	**0.0400**	**0.0200**	**0.0100**
User-based CF (stratified + svd)	**0.1135**	**0.0671**	**0.0400**
User-based CF (All optimizations)	**0.1283**	**0.0746**	**0.0431**
Lightfm(tags)	**0.95**	**0.95**	**0.93**
Lightfm(tags + ids)	**0.06**	**0.06**	**0.03**
Lightfm(tags + uds)	**0.07**	**0.06**	**0.01**
Lightfm(tags + about)	**0.05**	**0.05**	**0.01**
SBert+FAISS	**0.6016**	**0.5696**	**0.5336**

- Precision@k
  - Measures the average proportion of relevant items in the top-k recommendations among all users.
  - Higher mean precision@K means better
  - Formular: $Precision@k = \frac{Recommended_K \cap Relevant}{K}$

Conclusion and Further Works

Advantages and Limitations

Future Works

06 Conclusion

Advantages:

Efficient for Large Datasets

Computationally efficient for sparse datasets, no complex model training.

Cold Start for New Items

New items lack enough interaction data, causing cold start problems.

Works Well with Implicit Feedback

Effective for systems with implicit feedback (e.g., purchase counts).

Limitations:

Sparsity of Data

Sparse user-item interactions lead to less reliable similarity calculations.

Scalable

Pearson correlation scales well with dataset growth, especially with item popularity filtering.

Limited to Item-Item Similarity

Ignores user preference changes or broader contexts (e.g., seasonality).

06 Future Works

- Collaborative Filtering
  - Incorporate product metadata (category, name, etc.)
  - Combine with neural models (NCF, LightGCN, etc.)
  - Use hybrid filtering (content + collaborative)
- LightFM
  - Obtain more useful features(user: age, gender, and location, etc. item: item description , etc.) and explicit feedback(rating , etc.)
  - Using other feature engineering method
- Sentence Bert+FAISS
  - Add more item text like product description

THANK YOU