UNSW
SYDNEY

COMP9727 Project Report

Fashion Recommendation System

Haokai Zhao
Jiaqi Wang
Baoxi Liu
Ziang Zhang

Contents

1 Introduction

With the rise of e-commerce, customers now have the convenience of browsing millions of products from the comfort of their homes, avoiding the need to visit multiple stores, stand in long queues, or try on garments in dressing rooms. However, given the plethora of options available, an effective recommendation system is necessary to properly sort, order, and communicate relevant product material or information to users. Effective fashion recommendation systems can have a noticeable impact on billions of customers' shopping experiences and increase sales and revenues on the provider side.

In this project, we built a hybrid system involving content-based, rule-based, and collaborative filtering for fashion recommendations. Our target scenario was to promote sales when customers browse the fashion app without specific purchase intentions. By offering personalized recommendations, our system enhances the shopping experience and increases the likelihood of users purchasing products.

2 Problems

2.1 Competitor Analysis

Fashion brands typically have their own e-commerce websites that require users to select several filters, such as product categories and types, before presenting products according to those filters [1, 2]. While such search-based systems can provide relevant items based on users' specified needs, they fall short in offering good recommendations when users do not have a clear idea of what they want to buy.

To bridge this gap, we aimed to build a personalized fashion recommendation system that can recommend garments to users without any filters defined by the users. By analyzing users' and items' metadata and users' purchase history, our recommendation system can infer users' personalized preferences, providing tailored suggestions that anticipate users' needs even when they are uncertain about their choices.

2.2 Problem Definition

The fashion item recommendation can be defined as a ranking problem. Our objective is to rank items based on how likely each user is to purchase them.

(a) UI (Phone) (b) UI (Computer)

Figure 1: Mock-up User Interfaces

This ranking is personalized, tailored to each individual user's preferences and purchases. Given users' metadata, such as age and postal code, items' metadata, including image, text description, and product name, and users' purchase history, we aimed to find a scoring function f that maps each pair of user u and item i to a score. Formally,

$$f : (u, i) \to \mathbb{R} \tag{1}$$

where u represents the user, characterized by metadata such as age and postal code, and i represents the item, characterized by metadata including image, text description, and product name. The score $f(u, i)$ indicates the probability of user u purchasing item i. The system will rank the items i for each user u such that the items with the highest scores $f(u, i)$ are recommended to the user.

2.3 User Interface

Figure 1 illustrates a mock-up user interface of our recommendation system. Approximately 10 items will be recommended to the user on a computer web

page, with fewer items displayed on a phone interface at first glance. Users can purchase from the recommended items or scroll down to explore more recommendations. Each user's purchase operation will be recorded to build and update the recommendations.

3 Datasets

We used the dataset from Kaggle: *H&M Personalized Fashion Recommendations* [3]. It contains 3 types of data:

- 105,542 **items** information, including item's image, text description and some other descriptive attributes.
- 1,371,980 **customers** metadata information, including customer's age, postcode and active status etc.
- 31,788,324 **interactions** between user and item, each record describes user u purchased item x at time t with price p at sales channel s.

3.1 Exploratory Data Analysis

The goal of Exploratory Data Analysis (EDA) is to develop an intuitive understanding of the dataset, identify the potential problems or bias and access the feasibility of our task. There are three aspects to consider: 1. whether the items have enough representative feature, 2. whether the user data are rich in feature and 3. the most important one, whether there are enough interactions between user and items for each category.

3.1.1 Item feature explore

The item dataset have 25 columns in total. There are 22 columns contains 2 to 300 distinct values, as shown in Table 1. Although some columns are correlated, such as color group code and color group name, it still has some important and independent attributes about items, such as color, department and graphical appearance etc. Additionally, each item is associated with a detailed description ('detail desc') and an image linked to its article ID, as illustrated in Figure 2. Overall, the dataset provides sufficient information to develop distinctive and representative features for items.

Table 1: Number of distinct values for each column in items data

Column	Unique	Column	Unique
article_id	105542	graphical_appearance_name	30
product_code	47224	colour_group_code	50
prod_name	45875	colour_group_name	50
product_type_no	132	perceived_colour_value_id	8
product_type_name	131	perceived_colour_value_name	8
product_group_name	19	perceived_colour_master_id	20
graphical_appearance_no	30	perceived_colour_master_name	20
department_no	299	index_code	10
department_name	250	index_name	10
index_group_no	5	section_no	57
index_group_name	5	section_name	56
garment_group_no	21	garment_group_name	21
detail_desc	43405		

3.1.2 User feature explore

User data has 6 feature columns: FN (Fasion News), Active, club_member_status, fashion_news_frequency, age and postal_code. Those attributes could be used to group users together and provide some information beyond users' likes for user-based collaborative filtering. For example, customers with the same age could have the similar interests. The distribution of some columns is shown in Figure 3. While the dataset only provides moderate information about the customers, those attributes will likely to be used for retrieval rules rather than building latent features like items' image and text.

3.1.3 User-item interaction explore

The dataset provide extensive user-item interaction records (31,788,324!) and information including time, price and sale channel of purchase are also provided for each transaction. While such amount of data allow us to build complex and meaningful interaction-based latent space, it still perverse some problems. As shown in Figure 4a, there are about 10,000 items got purchased more than 1000 times, and there are 10,000 items got purchased less than 5 times. For user-based collaborative filtering, those items got limited number

Figure 2: Examples of image and text description for items

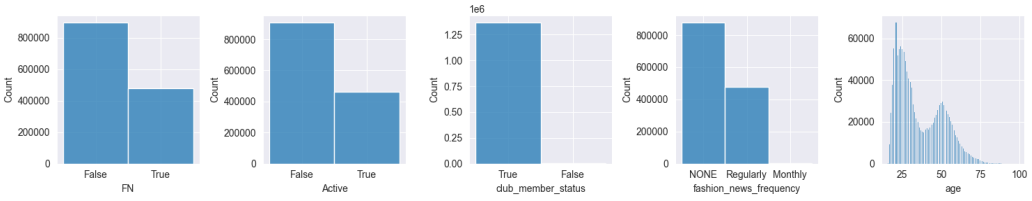


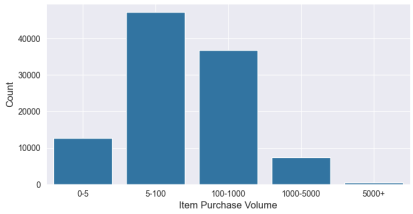
Figure 3: User Attributes Distributions

of purchased will almost never be recommended since the more popular items will always get a higher score. Furthermore, as shown in Figure 4b, there are more than 400,000 users made purchase less than 5 times. It will be really hard to perform high quality recommendations for those users, as their user profiles are heavily biased to a few items.

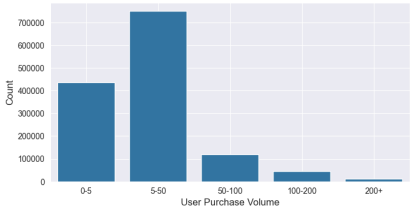
3.2 Summary of the Dataset

In summary, the dataset exhibits 5 key characteristics that warrant attention:

- **Extensive user-item interaction:** This dataset provides a vast amount of user-item interactions. This is key motivation for us to choose the dataset as interaction data usually plays a crucial role in building recommendation system.
- **Extensive item data and features:** The dataset contains a large number of items, each with rich and diverse features. While this wealth of features enhances the potential for building sophisticated recommendation systems, the large number of items could be time costly for real-time computation and recommendations.



(a) Number of products with different group of purchase counts



(b) Number of users with different group of purchases

Figure 4: Purchase volume group for items and users

- **Large user data:** The dataset contains a large number of users, though limited information are provided for each user. This will shift the construction of user feature to focus on user-item interactions.
- **Long-tail problem for items:** Many items in the dataset have limited purchase histories. This poses a challenge for traditional user-based collaborative filtering methods, which may struggle to recommend these less popular items effectively.
- **Long-tail problem for users:** A significant portion of users have made only a few purchases. It will be hard to make high quality recommendations to those users since the recommendations will always be biased to the few items that they purchased.

4 Methods

4.1 Overview

Our fashion recommendation system involves three modules: feature extraction, candidates generation and ensemble recommendation. In the feature extraction module, we used DINO v2 [4] and GloVe [5] to extract image and text feature, respectively, for all the items. For the candidates generation module, we developed 2 content-based methods, 4 rule-based methods and 3 collaborative filtering methods to select recommendation candidates. We then used an ensemble module to aggregate the recommendations from each methods to make the final recommendations. Figure 5 shows an overview of our developed fashion recommendation system.

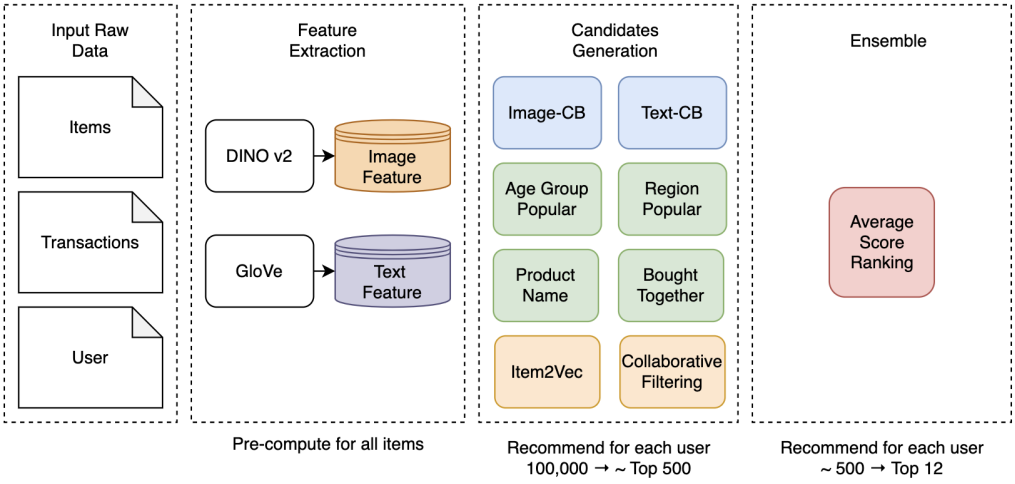


Figure 5: Recommendation System Overview

4.2 Feature Extraction

The feature extraction module aims to build feature vector for each items in the database. The distance between feature vectors are expected to describe how similar are the corresponding items, thus can be used for content-based recommendations.

4.2.1 Image Feature Extraction

We used the open-source pre-trained DINOv2 model [4] for image feature extraction. Developed by Meta AI, DINOv2 is a large-scale self-supervised learning model based on Vision Transformer [6] for image representation learning. The model processes images as sequences of fixed-size patches that are linearly embedded, with a [CLS] token added at the beginning of the sequence to represent the entire image for downstream tasks.

For implementation, we utilized `AutoImageProcessor` and `AutoModel` from the `transformers` package to load the image processor and pre-trained model, respectively. We extracted the first output token (the [CLS] token) from the model's output as the image feature vector for each item. Each image feature is a vector with a shape of 384. We performed inference on all 100,000 images using an NVIDIA TITAN RTX, which took approximately 4 hours.

4.2.2 Text Feature Extraction

We used GloVe [5] with 6B tokens and a word vector size of 300. We concatenated all words from the following item attribute columns to construct a document for each item:

`prod_name`	`index_name`
`product_type_name`	`index_group_name`
`product_group_name`	`section_name`
`graphical_appearance_name`	`garment_group_name`
`colour_group_name`	`department_name`
`perceived_colour_value_name`	`detail_desc`
`perceived_colour_master_name`	

We excluded words not found in the GloVe dictionary. For each item's document, we retrieved the GloVe word embeddings and computed the feature-level mean of all word embeddings to form the item's text feature vector. The final text feature vector for each item has a shape of 300.

4.3 Candidates Generation

We developed nine recommendation pipelines for item candidate generation: two content-based (image and text), four rule-based (age group, postal code, product name, and bought together), and three collaborative filtering (matrix factorization and two Item2Vec-based). Each pipeline recommends N items for a query user, with N fixed at 100 for all pipelines. The recommendations from each pipeline can either be directly presented to users or fed into an ensemble module to produce the final recommendation.

4.3.1 Content-Based: Image

For image content-based recommendations, we computed the nearest neighbors for each item based on their image features extracted from DINOv2. We used the inverse of Euclidean distance as our similarity score function:

$$similarity(i, j) = \frac{1}{1 + \|i - j\|_2^2} \tag{2}$$

where i and j are item feature vectors, and a larger score indicates a greater similarity between i and j. We used `faiss` to accelerate the computation process. Excluding the item itself, we obtained the top N similar items ordered by their similarity scores for each item. This pre-computation allows for quick retrieval of similar items during the recommendation process.

For each user, we used the precomputed item similarity information to perform recommendations. We retrieved the items the user had purchased and fetched the top N similar items and their scores for each of these items. These candidates and their similarity scores were concatenated and sorted in descending order of similarity. The top N unique items were then selected as recommendations.

4.3.2 Content-Based: Text

For text content-based recommendations, we used the same algorithms and implementation as in image content-based recommendations, except that we replaced the item features with text features extracted and constructed from GloVe. Again, the inverse of Euclidean distance (Eqn. 2) is used as the similarity function.

4.3.3 Rule-Based: Age Group Popularity

For age group popularity-based recommendations, we grouped customers into age intervals of size 5. Each customer was assigned an age group based on their age. Next, we compute the purchase count during training period for each item, as an indication of popularity of items. The items were then sorted by popularity within each age group. During the recommendation process, we used the pre-computed popularity rankings to recommend the top N items that were most popular within the user's age group.

4.3.4 Rule-Based: Region Popularity

For region popularity-based recommendations, we used the customers' postal codes to group them by region. We computed the purchase count during training period for each item, as an indication of popularity of items. The items were sorted by popularity within each region. During the recommendation process, we used the precomputed popularity rankings to recommend the top N items that were most popular within the user's region.

4.3.5 Rule-Based: Items Product Name

For product name-based recommendations, we first identified the product codes associated with each item in the user's purchase history. We then retrieved items in the database that share the same product code with user purchased items. The items were then sorted based on their overall popularity. During the recommendation process, we recommended the top N popular items that had the same product code as the items the user had previously purchased.

4.3.6 Rule-Based: Items Bought Together

For the recommendation rule based on items bought together, we analyzed the co-purchase patterns of items. Using a co-purchase matrix C, we recorded the number of times for each pair of items was bought together during training period. Formally speaking,

$$C_{i,j} = \sum_{u} \mathbb{1}_{i \in P_u} \cdot \mathbb{1}_{j \in P_u} \tag{3}$$

where $C_{i,j}$ is the co-purchase count for items i and j, P_u is the set of items purchased by user u, and $\mathbb{1}_x$ is an indicator function that equals 1 if x is true and 0 otherwise.

For each item in the user's purchase history, we retrieved the top N items that were most frequently bought together with it. During the recommendation process, we used the precomputed co-purchase data to recommend the top N items that were frequently bought together with the user's previously purchased items.

4.3.7 Collaborative Filtering: Matrix Factorization

For collaborative filtering based on matrix factorization, we utilized the Alternating Least Squares (ALS) algorithm [7, 8] from the `implicit` package to generate recommendations based on users' purchase history. The algorithm first maps user and item IDs to integer indices and constructs a sparse user-item interaction matrix, where each entry is a binary scalar indicating whether a user purchased an item during the training period.

Matrix factorization works by decomposing this user-item interaction matrix into two lower-dimensional matrices: one representing users and the

other representing items. Each user and item is associated with a latent factor vector, capturing their underlying preferences and attributes. The ALS algorithm iteratively adjusts these vectors to minimize the difference between the predicted and actual interactions.

We trained the ALS model on the user-item interaction matrix to learn the latent factors. During the recommendation stage, we recommended the top N items that have the highest dot product values with the query user's latent vector.

We used `AlternatingLeastSquares` from package `implicit`. We set `factors`= 50, `alpha`= 2.5, `iterations`= 15, and all other hyper-parameters as default value for model training.

4.3.8 Collaborative Filtering: Item2Vec Similarity

The Item2Vec [9] model adapts the principles of Word2Vec [10] for generating item embeddings based on users' purchase histories. By treating sequences of purchased items as sentences and each item as a word, the model learns item embeddings that capture the contextual similarity between items.

To train the Item2Vec model, we aggregate users' purchase histories into sequences, where each sequence represents the items bought by a user. These sequences are used to train a Word2Vec model using the Skip-gram algorithm. The training objective is to maximize the probability of observing context items given a target item. Formally, given a sequence of items $\{i_1, i_2, \ldots, i_T\}$, the objective is:

$$\max \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log P(i_{t+j} \mid i_t) \tag{4}$$

where c is the context window size, and $P(i_{t+j} \mid i_t)$ is the probability of observing item i_{t+j} given item i_t. This probability is defined using the softmax function:

$$P(i_{t+j} \mid i_t) = \frac{\exp(v'_{i_{t+j}} \cdot v_{i_t})}{\sum_{k=1}^{V} \exp(v'_k \cdot v_{i_t})} \tag{5}$$

where v_{i_t} and $v'_{i_{t+j}}$ are the input and output vector representations of items, and V is the vocabulary size.

After training, the model generates embeddings for each item, which are used to compute similarities between items and user profiles. For recom-

mendations, we employ the similarity recall method. Each user's profile is calculated as the average of the embeddings of items they have purchased, resulting in a single vector representing their preferences:

$$u = \frac{1}{|I_u|} \sum_{i \in I_u} v_i \tag{6}$$

where u is the user profile vector, I_u is the set of items purchased by the user, and v_i is the embedding vector of item i. The cosine similarity between the user profile vector and all item vectors is then computed:

$$\text{sim}(u, v_i) = \frac{u \cdot v_i}{\|u\| \|v_i\|} \tag{7}$$

The items with the highest cosine similarity scores are selected as the top-N recommendations, ensuring that the recommended items are contextually similar to those previously interacted with by the user.

For Item2Vec training, we used `Word2Vec` from the `gensim` package. We set `window`=9999, `vector_size`=256, `epochs`=15, `negative`=5, `sg`=1, `hs`=0, and `min_count`=10, with all other hyper-parameters set to their default values. The large window-size ensures all items in one user's purchase history are modeled together.

4.3.9 Collaborative Filtering: Item2Vec Cluster

Building on the trained model and latent vectors for items and users described in Section 4.3.8, we developed a variation of the Item2Vec similarity method. While the Item2Vec similarity method recommends items based on the similarity between a user's profile and item vectors—essentially functioning as an item-based collaborative filtering—the Item2Vec Cluster method implements user-based collaborative filtering.

In this approach, we first identify the top M users most similar to the query user, with M fixed at 200 in our implementation. We then collect all items purchased by these M users during the training period and rank the items by their popularity. Cosine similarity (Eqn. 7) is used to find the closest users, and we utilized `faiss` to accelerate the computation.

4.4 Ensemble

We used an ensemble module to aggregate recommendation results from each candidate generation pipeline. The goal of this module is to combine the strengths of different recommendation methods to produce a final set of recommendations for each user.

The intuition for our ensemble method is to recommend those items got high ranks in majority of the pipelines. We first constructed a candidate set S for each user, which is the union of items recommended by the selected pipeline set P. Each item x_i in this set receives a score from each pipeline P_j based on its ranking within that pipeline. Formally, if an item X_i appears in pipeline P_j with rank r_{ij}, it receives a score calculated as

$$\text{score}_{ij} = \frac{N - r_{ij}}{N} \tag{8}$$

where N is the number of top items considered from each pipeline. The score will be ranging from 0 to 1, and a higher score means a higher rank for an item in the pipeline. If the item does not appear in a pipeline, the item's score for that pipeline is zero.

For each item in the candidate set S, we compute the final score by summing its scores across all selected pipelines:

$$\text{final\_score}(X_i) = \sum_{j \in \text{pipelines}} \text{score}_{ij} \tag{9}$$

The items are then sorted based on their final scores in descending order, and the top K items are selected as the final recommendations for each user.

We tested three combinations of pipelines for ensemble in this project. The pipelines used for each ensemble methods are listed in Table 2, 3 and 4

Table 2: Pipelines Used for Ensemble (1)

Ensemble	**Content-Based**	**Rule-Based**	**Collaborative Filtering**
Ensemble (1)	4.3.2 Text-CB	4.3.4 Postal	4.3.9 Item2Vec Cluster

Table 3: Pipelines Used for Ensemble (2)

Ensemble	Content-Based	Rule-Based	Collaborative Filtering
Ensemble (2)	4.3.2 Text-CB 4.3.1 Image-CB	4.3.5 Product 4.3.4 Postal	4.3.9 Item2Vec Cluster 4.3.8 Item2Vec

Table 4: Pipelines Used for Ensemble (3)

Ensemble	Content-Based	Rule-Based	Collaborative Filtering
Ensemble (3)	4.3.1 Image-CB 4.3.2 Text-CB	4.3.5 Product 4.3.4 Postal 4.3.3 Age Group 4.3.6 Also Buy	4.3.7 User-CF 4.3.9 Item2Vec Cluster 4.3.8 Item2Vec

4.5 Recommendation for New Users

For users without any purchase history, we use their age and postal code to perform recommendations based on age group popularity (4.3.3) and region popularity (4.3.4). These methods do not require purchase history, making them well-suited for handling cold-start scenarios.

4.6 System Updates

We precomputed and stored recommendation items for all users, allowing us to provide fast recommendations by retrieving these precomputed results during the serving stage. Recommendations for new users are detailed in Section 4.5. After a set period, such as three days or one week, the system can be retrained and updated using newly collected data.

5 Experiments

5.1 Data Split

We used all users and articles for training and testing, performing cross-validation based on transaction data. Each model was trained on 12 weeks of data to make recommendations, and the following week was used to test

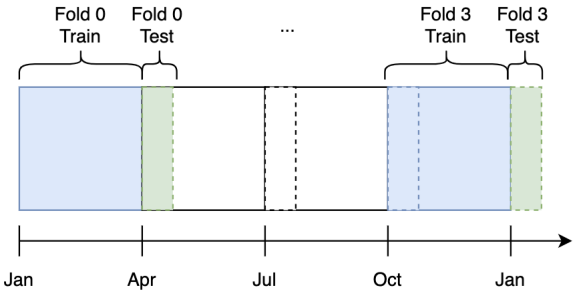


Figure 6: Train and test cross validation strategies

whether users purchased the recommended items. We performed 4-fold cross-validation using all transactions from 2019, as illustrated in Figure 6.

The time-based data split simulates real-world conditions where future user behavior is predicted based on past interactions. By training on sequential time windows and testing on subsequent weeks, we maintain temporal consistency and prevent data leakage, leading to more reliable evaluation of our recommendation systems.

5.2 Computation Resources

All of our recommendations were computed on a machine equipped with 256 GiB of RAM, 48 CPU cores, and one NVIDIA TITAN RTX GPU. The average time required to complete the recommendations for all users in each fold is listed in Table 5.

5.3 Evaluation Metrics

We focused on three evaluation metrics: Precision, Recall, and Mean Average Precision (MAP) to evaluate our recommendation system. For all methods and their ensembles, we recommended 12 items to each user and evaluate with the three metrics.

5.3.1 Precision

Precision measures the proportion of recommended items that are actually purchased in test period, providing insight into the accuracy of the recom-

Table 5: Average Computation Time for Each Recommendation Method

Method	**Average Time**
4.3.1 Image Content-Based	60 minutes
4.3.2 Text Content-Based	60 minutes
4.3.4 Postal Code	Less than 1 minute
4.3.3 Age Group	Less than 1 minute
4.3.5 Product Code	Less than 1 minute
4.3.6 Bought Together	3 minutes
4.3.7 Matrix Factorization	3 minutes
4.3.8 Item2Vec Training	3 minutes
4.3.8 Item2Vec Similarity	2 minutes
4.3.9 Item2Vec Cluster	Less than 1 minute
4.4 Ensemble	6 minutes

mendations. Formally, for a given user, Precision is defined as:

$$\text{Precision} = \frac{|\text{Actual Purchase} \cap \text{Recommended Items}|}{|\text{Recommended Items}|} \tag{10}$$

5.3.2 Recall

Recall evaluates the proportion of items that were actually purchased during the test period and were successfully recommended, offering a perspective on the model's ability to capture the entire set of purchased items. For a given user, Recall is defined as:

$$\text{Recall} = \frac{|\text{Actual Purchase} \cap \text{Recommended Items}|}{|\text{Actual Purchase}|} \tag{11}$$

5.3.3 Mean Average Precision (MAP)

MAP is used to summarize the precision across multiple users, accounting for the order of recommendations. It is calculated by taking the average of the Average Precision (AP) across all users. Average Precision for a single user is defined as the mean of the precision values calculated at the ranks where purchased items during test period appear in the recommendation list:

$$\text{AP} = \frac{1}{K} \sum_{k=1}^{K} P(k) \cdot \text{rel}(k) \tag{12}$$

where K is the number of recommended items, $P(k)$ is the precision at rank k, $\text{rel}(k)$ is an indicator function that is 1 if the item at rank k is relevant, and 0 otherwise. The MAP is then computed as the mean of AP over all users:

$$\text{MAP} = \frac{1}{U} \sum_{u=1}^{U} \text{AP}_u \tag{13}$$

where U is the total number of users. Precision and Recall provide a direct measure of the relevance of the recommendations, while MAP accounts for the ranking of the recommended items, offering a comprehensive evaluation of model performance.

5.3.4 Trade-off between Metrics

Precision, recall, and MAP evaluate our recommendation system from different perspectives. Since users see only a few items, it is crucial that these items are highly relevant to ensure satisfaction, making precision a key metric. However, focusing solely on precision can bias the evaluation, particularly favoring users who make many purchases during the test period. Recall complements precision by measuring how well the system captures all items relevant to a user's interests, providing a broader view of the system's effectiveness.

In our scenario, where items are presented in order, we assume that earlier exposure increases the likelihood of purchase. While precision and recall do not account for item order, MAP does, offering a more comprehensive evaluation by considering both relevance and ranking, ensuring that the most relevant items appear first.

5.4 Real User Feedback

While precision, recall, and MAP offer valuable insights into our recommendation system's performance, they cannot fully capture the nuances of user experience. These metrics rely on historical data, but user preferences can change, and the context in which recommendations are made is often

dynamic. Therefore, real user feedback is crucial to gauge how well the recommendations align with current user needs and satisfaction.

To collect real user feedback, we designed a controlled study with the following steps:

1. **User Selection**: Choose 8 typical users' purchase history from the training period of the first fold, ensuring a diverse representation based on age and the number of purchases.

2. **Volunteer Matching**: For each volunteer, present the purchase histories of these 8 users and ask them to select the one that best matches their preferences.

3. **Recommendation Presentation**: Show the volunteer 12 recommendations generated from four different methods: a content-based method (image), a rule-based method (age group), a collaborative filtering method (Item2Vec similarity), and an ensemble method, all based on the selected user's purchase history.

4. **Feedback Collection**: Ask the volunteer to rate each set of recommendations on a scale of 1 to 5. Additionally, request general feedback and suggestions on the quality and relevance of the recommendations.

6 Evaluations

6.1 Cross Validation Results

6.1.1 All Existing Users

Table 6 reports the 4-fold average performance of various methods and ensembles, evaluated on all existing users. Figure 7 illustrates the same results for easy comparison.

From the results, product and postal code rule-based methods performed well, with the product rule-based method showing a strong recall of 0.0401 and a relatively high precision of 0.0088. Another outstanding method is Item2Vec, particularly in terms of MAP, with the Cluster and Similarity variations achieving MAP scores of 0.0203 and 0.0180, respectively.

Content-Based methods, especially Image-CB and Text-CB, showed lower performance compared to rule-based and collaborative filtering methods, indicating that they may be less effective in capturing user preferences solely based on content features.

Table 6: Various methods and ensembles on all existing users

Method	**Precision@12**	**Recall@12**	**MAP@12**
Image-CB (4.3.1)	0.0010 ± 0.000	0.0043 ± 0.001	0.0016 ± 0.000
Text-CB (4.3.2)	0.0026 ± 0.000	0.0108 ± 0.000	0.0044 ± 0.000
Product (4.3.5)	0.0088 ± 0.001	0.0401 ± 0.001	0.0127 ± 0.000
Postal (4.3.4)	0.0102 ± 0.001	0.0365 ± 0.003	0.0110 ± 0.001
Age Group (4.3.3)	0.0029 ± 0.000	0.0106 ± 0.002	0.0035 ± 0.001
Also Buy (4.3.6)	0.0014 ± 0.000	0.0052 ± 0.000	0.0013 ± 0.000
User-CF (4.3.7)	0.0036 ± 0.000	0.0133 ± 0.001	0.0048 ± 0.000
Item2Vec (4.3.9)	0.0099 ± 0.001	0.0419 ± 0.003	0.0203 ± 0.001
Item2Vec (4.3.8)	0.0090 ± 0.002	0.0407 ± 0.005	0.0180 ± 0.002
Ensemble (Tab 2)	0.0102 ± 0.001	0.0448 ± 0.003	0.0202 ± 0.001
Ensemble (Tab 3)	**0.0112 ± 0.001**	**0.0502 ± 0.003**	**0.0213 ± 0.001**
Ensemble (Tab 4)	0.0100 ± 0.001	0.0440 ± 0.002	0.0176 ± 0.001

6.1.2 New Users

Figure 8 illustrates the performance for Age Group Popularity and Region Popularity for users without any purchase history (User cold start). Although we applied region and age group similarity rule filtering, our system still performs poorly in the cold start scenario. The best MAP is 0.0213 (Ensemble 2) for existing users, and only 0.004 (region popularity) in cold start scenario.

6.2 User Feedback Results

6.2.1 Feedback Comparison

We collected 13 volunteers' feedback according to procedure described in 5.4. Figure 9 illustrated the average rating for image content-based 4.3.1,

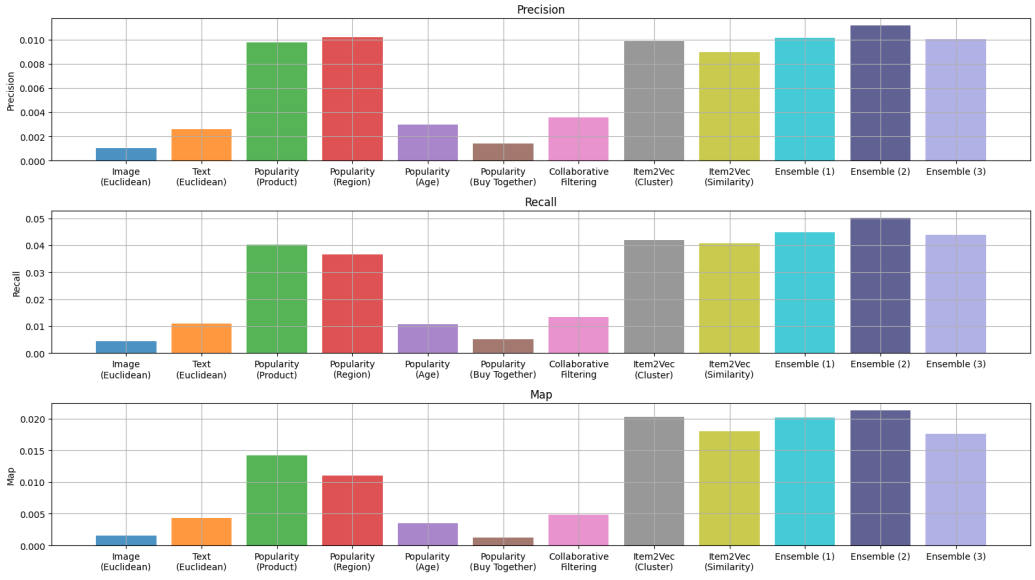


Figure 7: Various methods and ensembles on all existing users

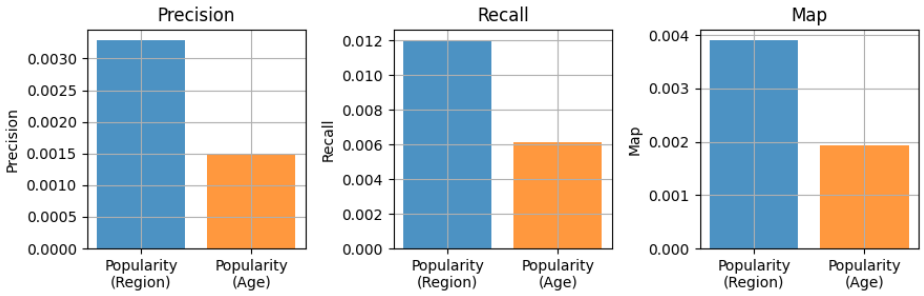


Figure 8: Region and Age group popularity recommendation performance for users without purchase history

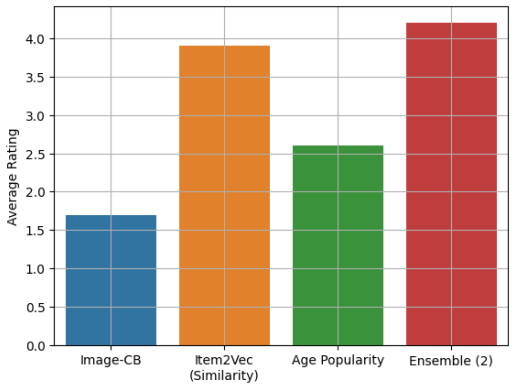


Figure 9: 13 Volunteers' average rating for each surveyed recommendation methods

item2vec similarity 4.3.8, age popularity 4.3.3 and ensemble 3. The recommendation from Item2Vec and Ensemble received high rating among the surveyed methods.

6.2.2 Case Study

We conducted a detailed case study on the purchase history and recommendations for the user with customer ID ca43bff-4f12b-59ad24-68c6d0-3b2977-10fe4b-a23a57-9f7467-4c5e12-9c8966-6124. The purchase history includes 9 items purchased during the training period, and we recommended 12 items based on this history. The four methods surveyed are illustrated in Figure 10. We observed that the content-based methods show the highest consistency with the purchase history but lack diversity. The Item2Vec method produces the most diverse recommendations, while the ensemble method strikes a good balance between consistency with the purchase history and diversity in the recommended items.

(a) 4.3.1 Image-CB (b) 4.3.8 Item2Vec

(c) 4.3.3 Age Group (d) Tab. 3 Ensemble (2)

Figure 10: Case Study for 4 surveyed methods. The 9 items on the left are purchase history and the 12 items on the right are what we recommended.

7 Reflections

7.1 What Went Well

We successfully developed a hybrid recommendation system that aggregated various methods. Extensive experiments were conducted, enabling us to evaluate the system thoroughly and compare different approaches comprehensively. Additionally, we built the project following strong software engineering practices, which allowed us to perform large-scale testing across all users and items.

Despite the challenging nature of our recommendation task (selecting 0.1% items from a pool of 100,000), we achieved reasonable performance and received some positive user feedback. Our system shows potential commercial viability.

7.2 What Could Have Been Improved

In our initial proposal, we aimed to build machine learning classification models to ensemble the results from different candidate generation pipelines. However, we were unable to complete this due to time constraints. Additionally, we only utilized a subset of the features provided in the dataset; several user attributes, such as *Active* and *Club Member Status*, were not incorporated into our models. Furthermore, although the dataset included timestamps for purchase operations, we did not develop sequential-based recommendations to leverage this information effectively. Lastly, our strategies for handling cold start scenarios showed limited performance.

7.3 Improvements and Future Work

Several improvements can be made in future work. First, incorporating sequential recommendation methods would better utilize the timestamps provided in the dataset, aligning the recommendations more closely with user behavior over time. Second, the average score ensemble method we employed may be overly simplistic; exploring more sophisticated ensemble methods, such as weighted averages or machine learning models, could further enhance performance.

However, introducing additional methods will increase the system's complexity. Therefore, carefully designed experiments are necessary to validate the effectiveness of each module and their combinations.

References

[1] H&M. H&m official website. `https://www2.hm.com/en_au/index.html`, 2024. Accessed: 2024-08-07.

[2] UGG. Ugg official website. `https://www.uggoutletstore.com.au`, 2024. Accessed: 2024-08-07.

[3] Carlos García Ling, ElizabethHMGroup, FridaRim, inversion, Jaime Ferrando, Maggie, neuraloverflow, and xlsrln. H&m personalized fashion recommendations. `https://kaggle.com/competitions/h-and-m-personalized-fashion-recommendations`, 2022. Kaggle.

[4] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[5] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[7] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pages 263–272. Ieee, 2008.

[8] Gábor Takács, István Pilászy, and Domonkos Tikk. Applications of the conjugate gradient method for implicit feedback collaborative filtering. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 297–300, 2011.

[9] Oren Barkan and Noam Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop*

on Machine Learning for Signal Processing (MLSP), pages 1–6. IEEE, 2016.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.