



RISK-AWARE RECOMMENDER FOR E-COMMERCE

by

Nguyen Trong Quoc Dat

Nguyen Nhat Minh

Tran Le Anh Tuan

FPT UNIVERSITY

RISK-AWARE RECOMMENDER FOR E- COMMERCE

by

Nguyen Trong Quoc Dat

Nguyen Nhat Minh

Tran Le Anh Tuan

Supervisor: Master. Phan Thanh Huy

DEPARTMENT OF ITS

FPT UNIVERSITY

A final year capstone project submitted in partial fulfillment of the requirement
for the Degree of Bachelor of Artificial Intelligence in Computer Science

April 2024

ACKNOWLEDGMENTS

With great respect, we would like to express our sincere gratitude to those who helped us to be able to complete this capstone project. We extend our deepest and most sincere gratitude to our advisor, Master Phan Thanh Huy, for always supporting us, giving us great advice and keeping this project on track.

Thanks to the teacher for taking the time to review our project and give us some really helpful feedback. Seriously, those comments helped us a lot in completing this project and we are very grateful for the insights you shared with us.

I truly appreciate my colleagues and friends at the FPT University for cheering us on and helping us when we needed it. Your support and willingness to share your knowledge were invaluable in contributing to the completion of this project.

Last but not least, this project could not have been completed without the help of the individuals above. Once again, we thank you from the bottom of our hearts for all your help and support throughout the project.

ABSTRACT

Nowadays, Recommender systems have played a crucial role in our daily lives. With the explosion of online shopping, also known as e-commerce, it is challenging for common recommendation tricks to work without much history of new users or items. For that reason, we have been digging deeper into this problem, testing new ways to make recommendation systems smarter even when they have little past data. We used several machine learning models to directly address this challenge. Hopefully, our work will advance personalized recommendations in the ever-changing world of online shopping.

In our project, we used several methods such as re-ranking, in addition to combining with base models such as CLCRec and CCFCRec to apply to this problem for training, testing and compare accuracy with existing base models but do not focus on cold-start phase. We will clarify each one in the following sections.

Keywords: Risk Awareness; Cold-start; Recommender system; CLCRec, CCFCRec.

CONTENTS

ACKNOWLEDGMENTS	3
ABSTRACT	4
CONTENTS	5
List of Figures	8
List of Tables	9
1. INTRODUCTION	11
<i>1.1 Overview</i>	11
<i>1.1.1 Overview Of Risk-Awareness</i>	15
<i>1.1.2 Fairness Context</i>	15
<i>1.1.2 Cold-start Context</i>	15
<i>1.2 Fairness Issues During The Cold-start Phase</i>	16
<i>1.2 Re-ranking</i>	16
<i>1.2 Our works</i>	17
2. RELATED WORK	18
<i>2.1 Fairness Definitions In Recommendation</i>	18
<i>2.2 Re-ranking method</i>	19
<i>2.3 Base model</i>	20
<i>2.3.1 CLCRec</i>	20
<i>2.3.2 CCFCRec</i>	25

2.3.2.1 Problem Fomulation	25
2.3.2.2 CCFCRec Architectures	26
2.4 Metrics	29
2.4.1 Precision@K	29
2.4.2 Recall@K	29
2.4.3 NDCG@K	30
2.4.4 NDCG@K Fairness	30
2.4.3 W	31
2.4.4 MDG	32
3. PROJECT MANAGEMENT PLAN	33
4. MATERIALS AND METHODS	38
4.1 Materials	38
4.1.1 Hardware	38
4.1.2 Data Management	39
4.1.3 Libraries And Frameworks	39
4.2 Method	40
4.2.1 Datasets	40
4.2.1.1 Statistic	43
4.2.1.2 Pre-processing	45
4.2.2 Algorithm	46
4.2.2.1 U-MMF Method	48
4.2.3 Grouped Item Recommendation Spread Metric	52

5. RESULTS	53
<i>5.1 Result of Base Model</i>	53
<i>5.2 Experiment</i>	57
<i>5.3 Result of Method</i>	58
6. DISCUSSIONS	64
7. DEPLOYMENT	66
8. CONCLUSIONS	71
9. APPENDIX A	73
10. APPENDIX B	74
11. APPENDIX C	76
12. REFERENCES	78

List of Figures

1.1 Figure 1. Illustrations the content-based filtering	12
1.1 Figure 2. Illustrations the collaborative filtering	13
1.1 Figure 3. Illustrations the hybird recommendation	14
2.2 Figure 4. Illustrations for different re-ranking types	19
2.3 Figure 5. Illustrations architecture of base model CLCRec	20
2.3 Figure 6. Illustrations architecture of U-I CEN	22
2.3 Figure 7. Illustrations architecture of R-E CEN	23
2.3 Figure 8. Illustrations architecture of base model CCFCRec	26
4.2 Figure 9. The distribution of item exposure of each dataset	44
5.1 Figure 10. The distribution of item exposure from the CLCRec result	55
5.1 Figure 11. The distribution of item exposure from the CCFCRec result	56
7.4 Figure 12. Homepage of our E-commerce website	67
7.4 Figure 13. Add Products page	68
7.4 Figure 14. Recommendation page	69
7.4 Figure 15. Interaction history page	70

List of Tables

3	Table 3.1 Weekly Project Plan	35
3	Table 3.2 Weekly Task Assignment	36
4	Table 4.1: Summary of the dataset [(S) is notation for small version]	43
4	Table 4.2: Statistic about the exposure of item in the dataset	44
4	Tabel 4.3: The dimension of features vectors in image, text, multi-hot	45
4	Table 4.4: Statistic for the experimental dataset	45
5	Table 5.1: Performance comparation between CLCRec and CCFCRec	53
5	Table 5.1.1: Statistics on item exposure resulting from the CLCRec model in each dataset	54
5	Table 5.1.1: Statistics on item exposure resulting from the CLCRec model in each dataset	56
5	Table 5.3: The hyperparameters for training model	57
5	Table 5.3: The time consuming of each algorithm for re-ranking methods	58
5	Table 5.4a: The recommendations performance of all base model and methods in dataset...	58
5	Table 5.4b: The recommendations performance of all base model and methods in dataset...	59
5	Table 5.4c: The recommendations performance of all base model and methods in dataset...	59
5	Table 5.4d: The recommendations performance of all base model and methods in dataset...	59
5	Table 5.4e: The recommendations performance of all base model and methods in dataset...	60
5	Table 5.5a: The recommendations performance of all base model and methods in dataset...	60
5	Table 5.5b: The recommendations performance of all base model and methods in dataset...	60
5	Table 5.5c: The recommendations performance of all base model and methods in dataset...	61
5	Table 5.5d: The recommendations performance of all base model and methods in dataset...	61
5	Table 5.5e: The recommendations performance of all base model and methods in dataset...	61
5	Table 5.6a: The recommendations performance of all base model and methods in dataset...	62
5	Table 5.6b: The recommendations performance of all base model and methods in dataset...	62
5	Table 5.6c: The recommendations performance of all base model and methods in dataset...	62

5	Table 5.6d: The recommendations performance of all base model and methods in dataset...	63
5	Table 5.6e: The recommendations performance of all base model and methods in dataset...	63
	Appendix Table B.1. Source	74
	Appendix Table C.1. Errors during import of base model	77

Chapter 1

INTRODUCTION

1.1 Overview

In the realm of customized electronic products, recommender systems are crucial for enhancing user experiences and assisting in finding appropriate products. They are utilized across various industries such as e-commerce, healthcare, education, etc. Users usually tends to stay on your website for a longer duration when your recommender system recommends a film they are likely to enjoy. The longer they stay, the more advertisements can be shown to them, resulting in increased revenue from advertising. In the current landscape, many retail stores have shut down some physical locations and transitioned to e-commerce as a fresh approach to promote and sell their products and services.

The e-commerce recommendation system works by suggesting items according to the user's preferences and interests. There are numerous approaches to pique a user's interest, and every recommendation system will identify it in a unique way. Recommendation systems can gather user preferences or interests through three primary ways: content-based filtering approach, collaborative filtering approach, and hybrid approach. Additionally, methods like demographic, knowledge-based, and community-based approaches exist, but we will only focus on the three most common types.

Content-based Filtering. Recommend items to the user based on their profile or the content they engage with, focusing on attributes similar to those of items they have previously selected. Show in Figure 1.

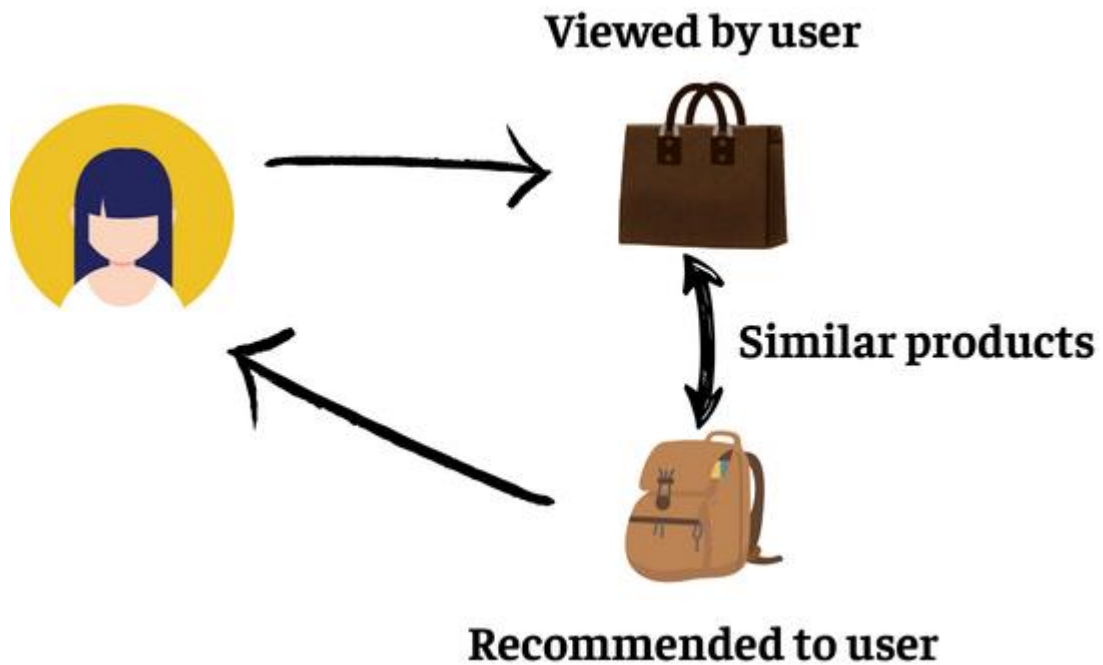


Fig. 1. Illustrations the content-based filtering

Collaborative Filtering. The collaborative filtering approach recommends items based on similarities between users or items. Simply explained, this is a way to suggest a user based on the behavior of similar users. Collaborative filtering can be done in two ways: the first is user-based collaborative filtering, which finds users who are most similar to one another and shares their preferences with other users [1]. Item-

based collaborative filtering, which is based on items and suggests related items that are appreciated by several users [2, 3], is the second kind. Show in Figure 2.

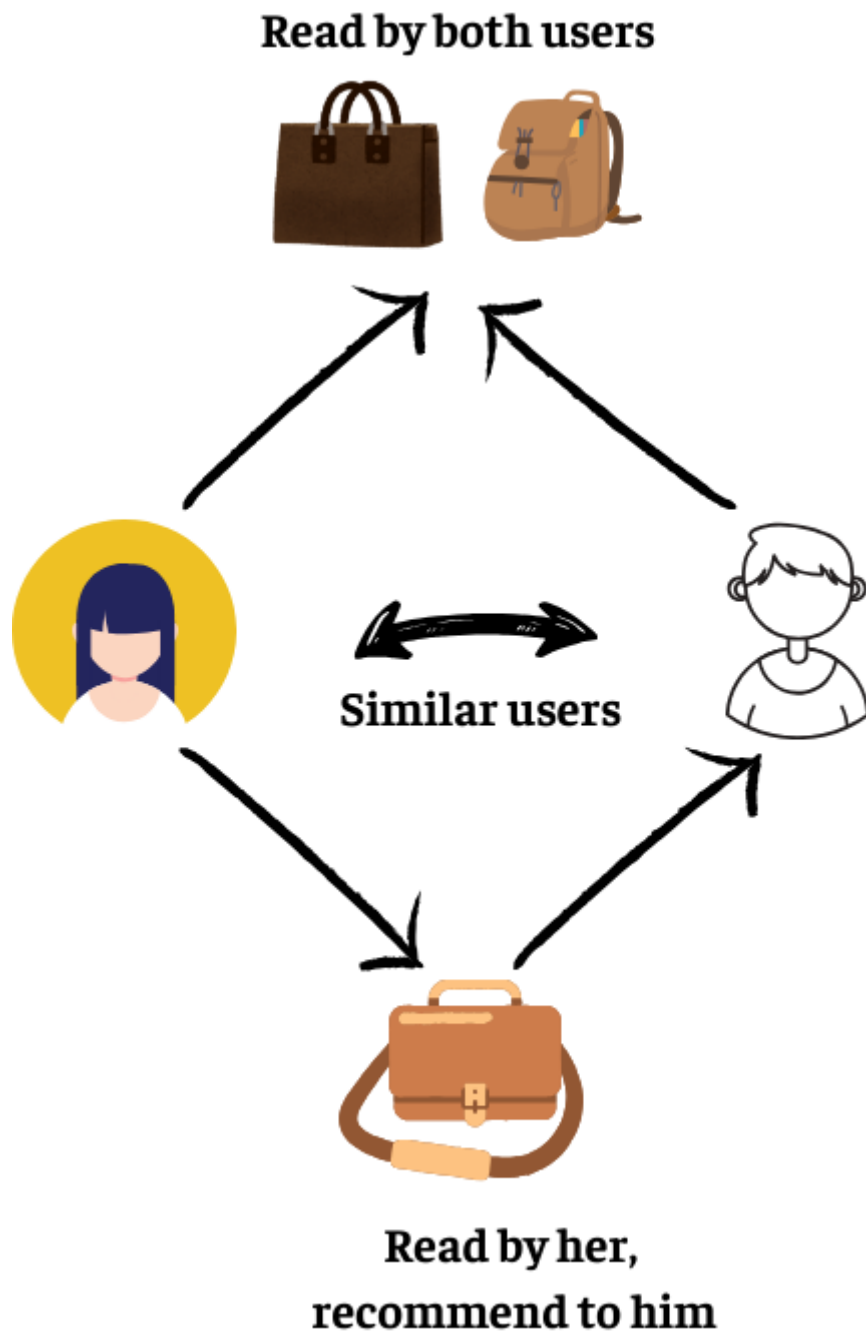


Fig. 2. Illustrations the collaborative filtering

Hybrid Recommendation. Hybrid approach is a method that combines the techniques mentioned above. For example, the CF method will have problems with new items, meaning it cannot provide suggestions for unrated items. However, this does not affect content-based because this method predicts items based on their descriptions. Combining these two methods can help the system take advantage of one technique to complement the other. Show in Figure 3.

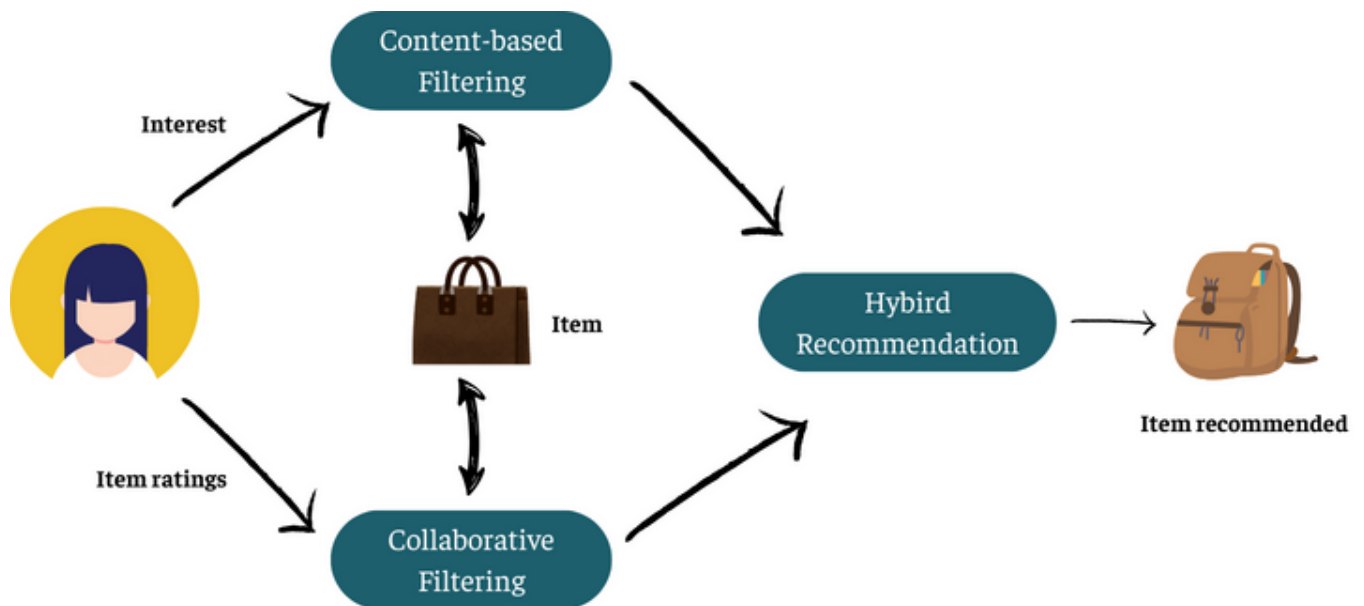


Fig. 3. Illustrations the hybird recommendation

In addition, there are two very interesting aspects of the recommender system: risk awareness and cold-start problem. We will go into further detail about these two factors below.

1.1.1 Overview Of Risk-Awareness

Recommendation systems today employ product introduction to draw in customers by taking into account factors like not upsetting users in particular situations or during particular times of the day. The following scenarios could happen: age, gender, race, time, context, etc. Therefore, the level of risk directly impacts

the performance of the system. These systems leverage advanced algorithms and user behavior analytics to provide relevant and personalized product recommendations for each shopper. With the huge amount of data generated by online transactions, browsing history, and user preferences, recommendation systems play a vital role in enhancing the overall shopping experience. By understanding each customer's unique needs and preferences, e-commerce platforms can not only increase customer satisfaction but also drive engagement and increase sales.

Risk definition: “Risk is disturbing or annoying the user, which can directly cause adverse effects on system performance.” [4].

1.1.2 Fairness Context

The first question we must ask is “Are the recommended items fair?” For example, in an online shopping product (item) recommendation engine that offers shopping suggestions to users, but whether users with different attributes such as gender, race, age, etc. Are treated equally? In the morning news, are different political stories presented fairly against each other? And are products from big companies popular with new users? Potential problems and suggestions are noted in the documents [5, 6, 7, 8, 9] with potential negative impacts on suppliers of those goods.

1.1.3 Cold-start Context

In most real-world applications, making recommendations on new items without any historical interaction from the user (formally known as cold start recommendation tasks), this is a challenge that has long existed in recommender systems. That is, how to onboard new users and new items without any historical interaction? ML methods often incorporate both user interaction data and existing cold boots (e.g. collaborative filtering methods). However, these approaches face many disadvantages such as the long distance from the conversion functions to the model's output layer, in addition to the consistency problem when applying the same conversion function to users. and various entries lead to poor transformations that produce poor model output (increasing the final recommendation error). Explore the cold start issue that arises when the system lacks any connection between the user and the items due to insufficient data availability. There are two types of cold start:

- 1. User cold-start problems:** When the system does not have any information about the user, it means the user has just registered and has not had any interaction with the system, so the system cannot provide personalized suggestions.

- 2. Item cold-start problems:** When there is almost no information about the item, it may be because the item has just been added, or the item already has some content information but no user interaction.

1.2 Fairness Issues During The Cold-start Phase

Addressing equity in the context of cold starts is imperative to ensure fair, enhanced user experiences and prevent the propagation of false biases. Achieving fairness in early recommender systems requires a delicate balance of incorporating diverse user preferences without perpetuating existing biases from historical data. In addition, sensitive user information must also be secured. This report explores the different methods used to improve fairness in cold start situations, including Separate-training methods [10, 11, 12, 13, 14] and joint-training methods [15, 16, 17]. By closely examining the fairness aspects of recommendation systems in e-commerce, this report aims to contribute to the development of more comprehensive and objective recommendation mechanisms that promote trust and user satisfaction across diverse and growing online markets. However, there is no concept of fairness in cold start systems. Specifically, in this article we will use two cold start recommendation systems CCFCRec [18], and CLCRec [19] to solve the fairness problem.

1.3 Re-ranking

The pursuit of optimizing recommender systems for e-commerce, by incorporating ranking methods, represents a strategic development in enhancing the effectiveness of existing techniques to address fairness in cold start situations. Re-ranking introduces a layer of post-processing that refines initial recommendations by considering additional dimensions such as user preferences, item characteristics, and fairness metrics. The combination of reordering with these start-from-scratch strategies gives greater adaptability to recommender systems, allowing them to dynamically adapt and fine-tune recommendations to relevant users with limited historical work. By integrating a re-ranking strategy, the recommendation system not only aligns with fairness goals but also personalized recommendations based on the user, tailoring them to individual user profile score. This report delves into the key role of rerating as a complementary technique to cold start problems, explores the synergies between these methods, and sheds light on their joint impact on fairness and user satisfaction in the field of e-commerce.

1.4 Our Works

In this article we are interested in two issues: cold start item for recommender system, and solving the issue of fairness in risk awareness using the reranking method. Our wish is to be able to handle the cold start problem of the recommender system while increasing the fairness of the system for users, specifically group fairness, without affecting the performance of the system. We will delve into the details of e-commerce recommendation systems and explore fresh ways to make them more accurate, fair, and personalized, especially when starting from scratch. We are all about shaking things up by

combining and upgrading models that already exist to fine-tune those initial suggestions. We will mix some theory on the tricky cold start problem and fairness for users with real-world examples from big players like Amazon. By doing this, we will see how combining cold start strategies with realignment methods can boost recommendation accuracy and maybe even make online shopping more inclusive, adaptable, and satisfying for everyone.

Chapter 2

RELATED WORK

2.1 Fairness Definitions In Recommendation

The recommendation system plays an important role in distributing information by providing personalized recommendations to users based on their interests or behaviors so that users can easily come into contact with items. Regarding distribution, there are two aspects that deserve attention. The first is the fair item allocation process, such as the fairness of the recommendation model. The second is the distributional outcome, such as the fairness of the information that users will receive. But in our article, we will only cover grouping by target. **Grouped by Target.** Based on whether the goal is to ensure fairness at individual-level or the group-level, fairness of outcomes can be classified into individual fairness and group fairness:

Group Fairness. In-group fairness dictates that protected groups should be treated similarly to advantaged groups as a result, which means that groups should be treated differently across the board. We can divide groups into many different ways, the most common is based on some obvious attributes related to some sensitive attributes such as gender, race, age, etc. In addition, when in groups with many related sensitive attributes, we can divide the groups into smaller groups. However, we still have to consider the fairness of these small groups because even when dividing unique attributes, it is still inevitable that these small groups are not treated fairly by the model [20, 21].

Individual Fairness. Individual fairness notes that individuals must be treated consistently by the model [22, 23].

Comparison between Group & Individual Fairness. Group equity is more complex than individual equity because within groups there are different divisions and the division of groups can be flexible, that is, an individual can belong to different groups depending on the situation different times [19]. Group fairness overlooks the value of each person, potentially resulting in the recruitment of less qualified team members. On the other hand, individual fairness can be viewed as a specific instance of group fairness, where every individual is part of a distinct group.

2.2 Re-ranking Methods

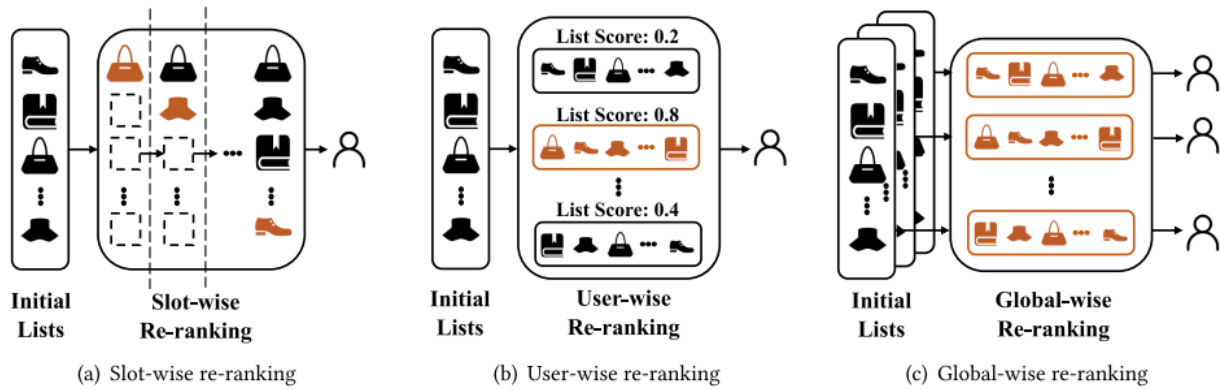


Fig. 4. Illustrations for different re-ranking types

To encourage fairness, modify the models' main reranking technique. One of the best ways to improve the fairness of the results is to use reranking methods, since their outcomes are almost exactly the same as the final ones. Furthermore, this method has a minimal coupling to the model which means we will not need to influence the original model directly but only use the output of that model to re-rank it. This will help us avoid having to modify or improve the recommender model architecture. However, the effectiveness of the reordering method can be compromised because the number of base model outputs fed into the reordering stage is often very small., so the effectiveness of the reordering method may be hindered. There are three types of re-ranking methods: slot-wise, user-wise, and global-wise [24]. Figure 1 depicts the differences between the three types. But we will only cover user-wise and global-wise in this article.

2.3.1 User-wise. This approach aims to identify the optimal recommendation list for the user directly, considering the overall optimization goal for the entire list. The re-ranking method is used for one recommendation list at a time.

2.3.2 Global-wise. In contrast to the above method, this method will re-rank multiple recommendation lists for multiple users at the same time.

2.3 Base Model

Cold-start Recommendation Models. There are many cold start models but we cannot test them all. Algorithms are typically categorized as joint-training, separate-training, combined, contrastive learning, and heuristic non-parametric methods. Our work selects algorithms that have good performance and are easy to implement. The CCFCRec method utilizes Contrast Collaborative Filtering for Cold-Start item Recommendation. It leverages collaborative signals from existing data to address the challenge of vague collaborative embeddings in cold-start item recommendation [18]. Additionally, we utilize another CLCRec method called the Contrast Learning-based method, which comprises three components: contrastive pair organization, contrastive embedding, and contrastive optimization modules [19].

2.3.1 CLCRec

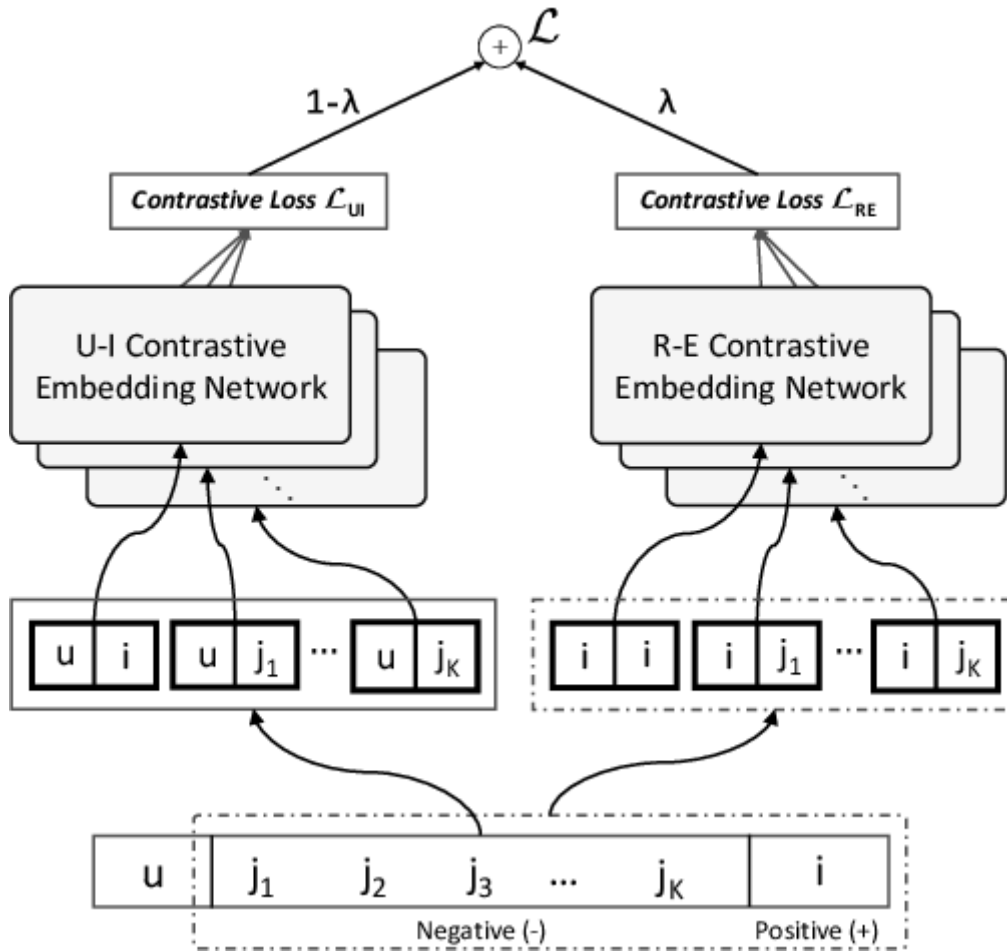


Fig. 5. Illustrations architecture of base model CLCRec

CLCRec, a cold-start recommendation system based on contrastive learning, comprises three primary components. The first is contrastive pair organization. For $U - I$ loss, Unobserved interactions are represented as negative user-item pairs, whereas past interactions are represented as positive user-item pairs. With $R - E$ loss, one item is matched with itself, while other items depict positive and negative item-item pairs, all of which are included in the self-discrimination task. After setting up the contrastive pairs $U - I$ and $R - E$, they will be embedded into contrastive embedding networks (CEN). The $U - I$ CEN will be responsible for creating collaborative embeds for pairs of users and items. The $R - E$ CEN contains an encoder that can be trained to find ways to represent features based on their content. The CLCRec framework consists of three components: contrastive pair organization, contrastive embedding network, and contrastive optimization.

Contrastive Pair Organization. identify positive pairs constructed by cases semantically with some negative pairs concatenated by different patterns. Therefore, it is important to organize users and items in contrasting pairs U-I and R-E.

$U - I$ Contrastive Pair. Pairs of user items observed in historical interactions, such as (u, i) depicted in Figure 5, are regarded as positive. Conversely, we randomly select sample K items (e.g., (j_1, j_2, \dots, j_K)), that have not been obtained by u to form negative pairs, as illustrated in Figure 1. Formally, positive and negative $U - I$ pairs can be defined as:

$$\{(u, i), (u, j_1), (u, j_2), \dots, (u, j_K)\}$$

Compared with negative pairs, positive pairs contain similar cooperative signals. This comparison helps identify the collaborative signals communicated through interactions.

$R - E$ Contrastive Pair. In contrast to individual pairs, pair-wise comparisons involve a self-discrimination task aimed at enhancing the mutual information of two distinct item representations. To form these pairs, consider item i in Figure 5 as an illustration: designate it as the anchor item and establish a connection with itself to create a positive pair. This method illustrates the semantic similarity between two representations of the same item. On the other hand, negative pairs that are arranged by anchors with different items have distinct semantic meanings. Illustrated in Figure 5, we combine the items to form the R-E pair:

$$\{(i, i), (i, j_1), (i, j_2), \dots, (i, j_K)\}$$

Where (i, i) is the positive pair and the others are negative ones.

Contrastive Embedding Network. We utilize contrast pairs to develop $U - I$ and $R - E$ contrast embedding networks (CEN) depicting users and items. The correlation of each pair is then computed using predefined density ratio functions.

$U - I$ Contrastive Embedding Network. To create collaboration signals based on user-item interactions, we initially retrieve their ID embeddings (e.g., e_u and e_i) from a lookup table specified by the parameter matrix:

$$E = [e_{i_1}, \dots, e_{i_N}, e_{u_1}, \dots, e_{u_M}].$$

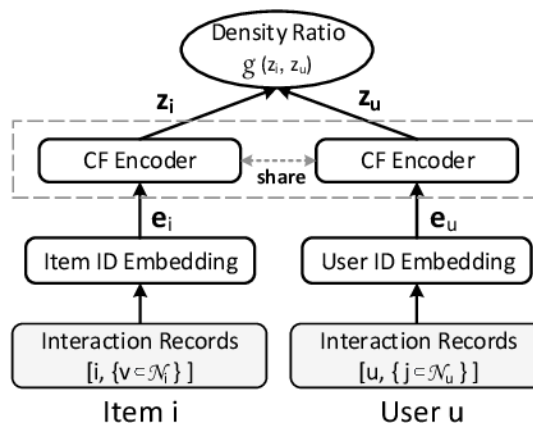


Fig. 6. Illustrations architecture of U-I CEN

Collaborative embeddings for users and items are learned using a shared CF encoder, as shown in Figure 6. We can apply different models, like MF-based [38, 39, 40], neural network-based [41], and graph neural network-based [42, 43, 44] models. We will keep the same architecture from this article [19], they use two simple yet efficient implementations, CEN_{MF} and CEN_{GCN} , these models based on MF [39] and LightGCN [42]. To be concise, we omit their details and present them as follows,

$$\begin{cases} z_i = \varepsilon(e_i, \{e_v | v \in \mathcal{N}_i\}) \\ z_u = \varepsilon(e_u, \{e_j | j \in \mathcal{N}_u\}) \end{cases}$$

Where, ε is the CF encoder takes id embedding as input; \mathcal{N}_i denotes the set of users who have interacted with item i , and \mathcal{N}_u denotes the set of items purchased by user u . Now we define a function to measure correlation after obtaining collaborative embeddings of user-item pairs:

$$g(z_i, z_u) = \exp(z_i^T z_u / \tau)$$

Where, τ represents a temperature parameter [45].

A hared CF encoder is utilized to acquire collaborative embeddings for users and items. This approach can be applied through different models, including MF-based, neural network-based models, and graph neural network-based models.

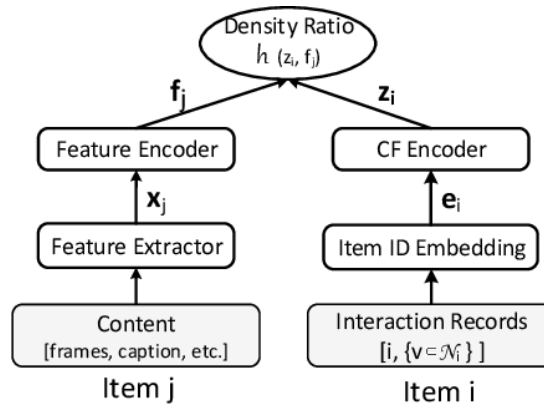


Fig. 7. Illustrations architecture of R-E CEN

R – E Contrastive Embedding Network. The network is utilized to assess the relationship between two perspectives on items – collaboration signals and content information. As show in Figure 7, it can be viewed as two parallel pathways for creating the item's feature representation and collaborative embedding, respectively. The appropriate way to embed collaboration is similar to the corresponding activities in the $U - I$ network.

$$f_j = W^{(2)}\phi(W^{(1)}x_j + b^{(1)}) + b^{(2)}$$

Where, $W^{(\cdot)}$ and $b^{(\cdot)}$ represents the trainable matrix and bias vector of the encoder. Furthermore, f_j denotes the desired feature of the item j , intending to condense the content feature while preserving the collaborative signal information to the fullest extent

After obtaining results from two pipelines, collaborative embedding and feature representation, density ratios are evaluated using a specific function:

$$h(z_i, f_j) = \exp\left(\frac{z_i^T f_j}{\|z_i\| \cdot \|f_j\|} \cdot \frac{1}{\tau}\right)$$

Contrastive Optimization. To maximize mutual information implement a contrastive training strategy to optimize the model parameters. Combine the defined density ratio functions.

$$\begin{aligned} \mathcal{L} &= \lambda \mathcal{L}_{RE} + (1 - \lambda) \mathcal{L}_{UI} + \eta \|\Theta\|_2^2 \\ &= -\lambda \mathbb{E}_{i \in I} \left[\ln \frac{\exp\left(\frac{z_i^T f_j}{\|z_i\| \cdot \|f_j\|} \cdot \frac{1}{\tau}\right)}{\exp\left(\frac{z_i^T f_j}{\|z_i\| \cdot \|f_j\|} \cdot \frac{1}{\tau}\right) + \sum_{k=1}^K \exp\left(\frac{z_i^T f_{j_k}}{\|z_i\| \cdot \|f_{j_k}\|} \cdot \frac{1}{\tau}\right)} \right] \\ &\quad - (1 - \lambda) \mathbb{E}_{(u, i) \in \mathcal{O}} \left[\ln \frac{\exp\left(\frac{z_i^T z_u}{\tau}\right)}{\exp\left(\frac{z_i^T z_u}{\tau}\right) + \sum_{k=1}^K \exp\left(\frac{z_{j_k}^T z_u}{\tau}\right)} \right] + \eta \|\Theta\|_2^2 \end{aligned}$$

\mathcal{L}_{RE} and \mathcal{L}_{UI} are used to represent the contrast loss to maximize the mutual information between $R - E$ and $U - I$. \mathcal{L}_{UI} shares the objective of enhancing Personalized Ranking (BPR) loss [25], and refines the recommendation model comprehensively without incorporating any additional loss function. Employ hyperparameters λ to maintain a balance between collaborative filtering and feature representation.

2.3.2 CCFCRec

We utilize a framework known as Contrastive Collaborative Filtering for Cold Start item Recommendation (CCFCRec). The primary concept involves instructing the CF module to retain the collaborative signals observed during training and correct the fuzzy CBCEs of new items based on these remembered collaborative signals.

This method operates on the specific foundation of the contrasting CF framework, comprising a content CF module and a co-occurrence CF module. As a result, for a matching training item, CFCRec is able to produce a CBCE and a Co-Occurrence Collaborative Embedding (COCE). When collaborative signals of co-occurrence are not active during the initial phase, the solution involves incorporating these signals into the content CF module while training. This ensures that the co-occurrence collaborative signals are utilized for training items instead of being directly encoded in CBCEs. CFCCRec uses complex contrastive learning to implement co-occurrence collaborative signals that are captured by COCE, remembered by the content CF module during the training phase, and made available during the application phase. This maximizes the information between the CBCE and COCE of an item. In a multi-task learning setup, the two collaborative filtering (CF) modules are simultaneously trained. Through multi-task optimization, there is a focus on maintaining accuracy in predicting engagement probability. This is achieved by the content CF module and the co-occurrence CF module leveraging an item's CBCE and COCE. This approach facilitates the effective transfer of co-occurring collaboration signals to the content CF module.

2.3.2.1 Problem Formulation

Let \mathcal{U} and \mathcal{V} is a set of users and set of items.

Let $\mathcal{O} = \{\mathcal{O}_{u,v}\}$ where $\mathcal{O}_{u,v}$ the interaction occurs between a user $u \in \mathcal{U}$ and an item $v \in \mathcal{V}$.

Let $\mathcal{V}_u \subseteq \mathcal{V}$ define as the set of items interacted with by user $u \in \mathcal{U}$ and users $\mathcal{U}_u \subseteq \mathcal{U}$ who interacted with those items $v \in \mathcal{V}$. For each item v iss associated with a attribute set X_v consiting of m attributes $\{x_1^{(v)}, \dots, x_m^{(v)}\}$. When representing an attribute, it can be shown as a one-hot vector (e.g., movie director), a multi-hot vector (e.g., movie genre), or a real-valued vector (e.g., item image).

Let $x_i^{(v)} \in \mathbb{R}^{d \times 1}$ be the embedding of the i th attribute $x_i^{(v)}$, where d is the dimensionality of embedding and $i \leq i \leq m$.

When provided with a warm training dataset \mathcal{D} consisting of $\{\mathcal{U}, \mathcal{V}, \mathcal{O}\}$, a cold-start item recommendation model can predict the likelihood $\hat{y}_{u,v}$ of a user u engaging with an item v : $\hat{y}_{u,v} = R(u, v, X_v)$, such that for any pair of $u^+ \in \mathcal{U}_v$, $u^- \notin \mathcal{U}_v$ in the training dataset, $\hat{y}_{u^+,v} > \hat{y}_{u^-,v}$.

2.3.2.2 CCFCRec Architecture

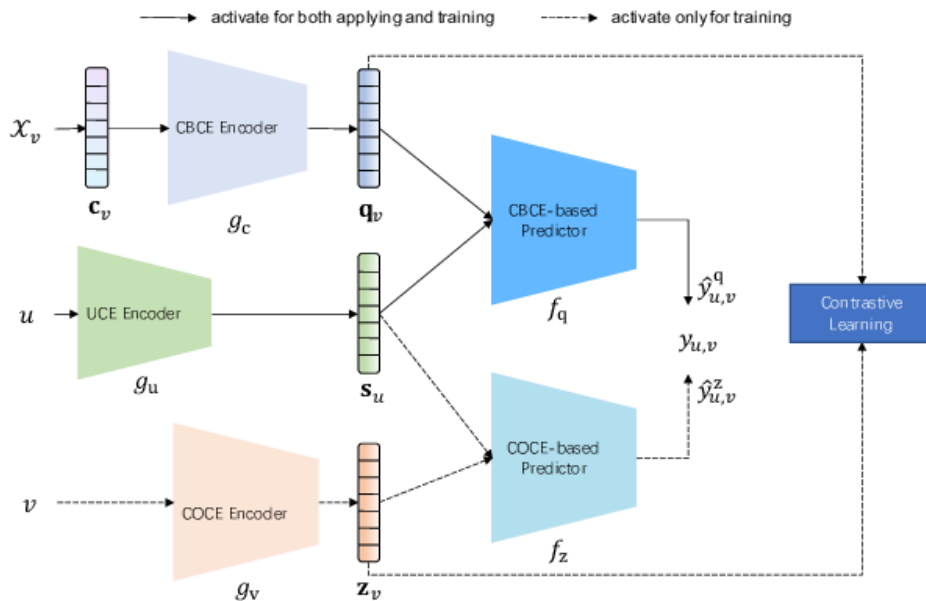


Fig. 8. Illustrations architecture of base model CCFCRec

In Figure 8, the content CF module includes the CBCE encoder g_c and the CBCE-based predictor f_q , the co-occurrence CF module includes the COCE Encode g_v and the COCE-based predictor f_z , and the UCE Encoder g_u is shared between two CF modules.

Collaborative Embedding

CBCE. Depending on how the attributes are encoded, first create the attribute embeddings $\{x_i^{(v)}\}$ for a training item v , where $i \leq i \leq m$. Given that $x_i^{(v)}$ is might be either a one-hot vector or multi-hot vector, followed by searching through a trainable embedding matrix $W_i \in \mathbb{R}^{d \times |x_i^{(v)}|}$ can yield its attribute embedding, i.e., $x_i^{(v)} = W_i x_i^{(v)}$. If $x_i^{(v)}$ is an image, the $x_i^{(v)}$ is obtained using a convolutional network that has been trained beforehand, such as VGG [37]. When the feature embeddings $\{x_i^{(v)}\}$ are ready, the content embedding $c_v \in \mathbb{R}^{md \times 1}$ be generated by a concatenation of $\{x_i^{(v)}\}$. At last, the CVCE q_v is generated using g_c with c_v as input:

$$q_v = g_c(c_v), \quad (1)$$

g_c construct a two-layer MLP with the LeakyReLU activation function to capture the nonlinear relationships between attributes.

COCE and UCE. The COCE z_c and UCE s_u encode the preferences of items v received from users and the preferences of user u offered to items, hrough hidden co-occurrence collaborative signals in the interactions observed between users and items. This process results in representing the item with a multi-hot vector $v \in \{0,1\}^{|u| \times 1}$, and the u -th component $v(u) = 1$ if $u \in \mathcal{U}_v$, or else $v(u) = 0$. In the same way, a user u is represented by a multi-hot vector $u \in \{0,1\}^{|v| \times 1}$, and the v -th component $u(v) = 1$ if $v \in \mathcal{V}_u$, or else $u(v) = 0$. For the sake of clarity, the COCE endcode g_v and the UCE encode g_u are put into action as linear combination across embedding matrix columns, i.e.,

$$z_v = g_v(v) = W_v u,$$

$$s_u = g_u(u) = W_u v,$$

Where $W_v \in \mathbb{R}^{d \times |u|}$ and $W_u \in \mathbb{R}^{d \times |v|}$ are matrices can be learnable.

CBCE captures user preferences for item attributes, while COCE captures co-occurring collaboration signals from interactions. However, CBCE cannot be directly encoded because it only contains items from the training phase. The content CF module will retain the collaborative signals present in its parameters during training to counter the unclear CBCEs from the initial entries. Cold start involves memory during the application phase. The model built for the training item v , $\mathcal{N}_v^+ = \{v^+ : \mathcal{U}_v \cap \mathcal{U}_{v^+} \neq \emptyset\}$ will be built as a positive sample set, meaning items that some users have interacted with v , and conversely it will be a negative sample set $\mathcal{N}_v^- = \mathcal{V} \setminus \mathcal{N}_v^+$. To maximize the mutual information between q_v and z_{v^+} and minimize the mutual information between q_v and z_{v^-} . The contrast loss function will be of the form:

$$\mathcal{L}_c = -\mathbb{E}_{v \in \mathcal{D}, v^+ \in \mathcal{N}_v^+} \left[\ln \frac{\exp\left(\frac{\langle q_v, z_{v^+} \rangle}{\tau}\right)}{\exp\left(\frac{\langle q_v, z_{v^+} \rangle}{\tau}\right) + \sum_{z_{v^-} \in \mathcal{N}_v^-} \exp\left(\frac{\langle q_v, z_{v^-} \rangle}{\tau}\right)} \right],$$

Where $\langle \cdot, \cdot \rangle$ represents inner product.

Interaction Prediction

Throughout the training phase, CFCCRec will generate two forecasts regarding the likelihood of interaction between a user u and an item v . These forecasts will be based on the COCE-based predictor f_z and the CBCE-based predictor f_q . Because of its simplicity, it will be implemented as a inner product prediction to rank the similarity of items and users.

$$\hat{y}_{u,v}^q = g_q(q_v, s_u) = \langle q_v, s_u \rangle,$$

$$\hat{y}_{u,v}^z = g_z(z_v, s_u) = \langle z_v, s_u \rangle.$$

For a training item v , each positive user $u \in \mathcal{V}_u$ shall be matched with a negative user u^- from sample $\mathcal{V} \setminus \mathcal{V}_u$. Then, by applying the popular pairwise ranking loss BPR [25], the loss functions for the joint training of the two predictors will be of the form:

$$\mathcal{L}_q = -\sum_{(v, u^+, u^-) \in \mathcal{D}} \ln \sigma(\hat{y}_{u^+, v}^q - \hat{y}_{u^-, v}^q),$$

$$\mathcal{L}_z = -\sum_{(v, u^+, u^-) \in \mathcal{D}} \ln \sigma(\hat{y}_{u^+, v}^z - \hat{y}_{u^-, v}^z),$$

where σ is sigmoid function.

The two CF modules notably have the same UCE s_u and are joint trained with the same supervision. By offering a constant optimization objective, this kind of multi-task learning makes sure that co-occurrence collaboration signals are actively transferred to the content CF module.

Overall Loss Function

The overall loss function is defined as:

$$\mathcal{L} = \mathcal{L}_q + \mathcal{L}_z + \lambda \mathcal{L}_c + \|\Theta\|,$$

Where λ is a component that regularizes the contrastive loss's contribution, and Θ represents the learnable parameters. Adam is choose for the optimizer.

2.4 Metrics

Now what do we use to evaluate the recommender system? Going through each question, we use four metrics to evaluate. We use Precision@K and Recall@K to evaluate the performance of recommender and ranking systems. Next use NDCG@K [26] to measure the quality of recommendation and information retrieval systems. Finally, to match our research goals, we use an additional metric called MDG [27]. We will go through each metric below.

2.4.1 Precision@K

Precision@K measures how many items in a long-K list have the same ratio. Simply put, it shows us how many of the suggested or retrieved items are related to each other.

Formula of precision:

$$Precision@K = \frac{\text{Number of relevant item in } K}{\text{Total number of item in } K}$$

Where K is a rank that you can choose arbitrarily to limit your evaluation. Let's say we are going to show the top 10 recommendations on an e-commerce site, then the value will be Precision@10. Now we have a binary sticker called The Relevance, which represents the interaction between the user and the item more easily understood than when the user has clicked on the product or added it to the cart. Precision answers the question of how many items will fit the user? For example, out of 10 suggested items (interaction results), there are 3 items that are relevant to each other, then the value

$$Precision@10 = \frac{3}{10} = 0.3.$$

2.4.2 Recall@K

Recall@K measures the ratio of relevant items identified in the top K recommendations to the total number of relevant items in the dataset. Simply put it allows us to find how many related items there are.

Formula of recall:

$$Recall@K = \frac{\text{Number of relevant item in } K}{\text{Total number of relevant items}}$$

Like Precision, K represents the cut-off point you consider for evaluation. Relevance is a binary label for situations like clicks, purchases, etc.

2.4.3 NDCG@K

NDCG@K stand for Normalized Discounted Cumulative Gain, it is a measure to evaluate the quality of recommendation and information retrieval systems. NDCG helps measure the ability to sort items based on relevance.

Formula of NDCG:

$$NDCG@K = \frac{DCG@K}{IDCG@K}$$

Where IDCG represent for a perfect ranking and DCG stand for Discounted Cumulative Gain. To calculate DCG we must identify related items that have less weight. This can usually be determined using a logarithmic penalty. Simply put, DCG will combine the scores of items in the recommended list and offer discounts according to their rankings. The formula will take the form:

$$DCG@K = \sum_{k=1}^K \frac{rel_i}{\log_2(i+1)}$$

We can calculate it by divided each item's gain with an increasing number as you proceed down the list, computed as an inverse logarithm of the position number. With rel_i is the relevant score of item position i .

2.4.4 NDCG@K Fairness

Distinguishing NDCG Fairness from Standard NDCG. When fairness index is factored in, it may diminish the quality of recommendation outcomes. This may lead to lower-ranked items being

elevated, impacting the provider's fairness. This article introduces an additional metric to evaluate the recommendation list post-method application versus the initial list. This step is necessary as the initial list typically aligns best with the customer's preferences in real-world scenarios.

$$DCG_{u,l} = v_{u,l[1]} + \sum_{i=2}^k \frac{v_{u,l[i]}}{\log_2(i+1)}$$

Dividing the original proposal list can help eliminate the customer's scoring habit. A value of 1 indicates a perfect match with the customer's preferences, while a smaller value signifies a lower quality of recommendation.

$$NDCG@K = \frac{DCG_{u,l_u}}{DCG_{u,l_u^{ori}}}$$

2.4.5 $W_\lambda@K$

In addition to the metrics given above, we also offer another metric called trade-off performance. This metric is included because to ensure overall fairness we will have to balance user-side preference and provider-side fairness. A truly fair recommendation system must have preferences on the part of users, fairness on the part of providers, and trade-offs between them. For provider fairness:

$$MMF@K = \min_{p \in \mathcal{P}} \left\{ \sum_{t=1}^T \sum_{i \in L_K^F(u_t)} \mathbb{I}(i \in I_p) / \gamma_p \right\}$$

Where, $\mathbb{I}(\cdot)$ denotes the indicator function, $L_K^F(u_t)$ represents re-ranking list. The chronological interactions in the test data were divided into interaction sequences with a specified horizon length T . \mathcal{P} is the sets of providers. Now we will go through trade-off performance:

$$W_\lambda@K = \frac{1}{T} \sum_{t=1}^T \left(\sum_{i \in L_K^F(u_t)} s_{u_t,i} \right) + \lambda \cdot MMF@K$$

Where $\lambda \geq 0$ is a trade-off performance.

2.4.5 MDG

MDG is Mean Discounted Gain for an item i , it is a measure to calculate the true positive rate of an item. Fomula of MDG:

$$MDG_i = \frac{1}{|u_i^+|} \sum_{u \in u_i^+} \frac{\delta(\hat{z}_{u,i} \leq k)}{\log(1 + \hat{z}_{u,i})}$$

Where \mathcal{U}_i^+ is a set of matched users for the new item i in the test set. $\hat{z}_{u,i}$ is the ranking position of i for user u by a cold start recommendation. $\delta(x)$ return 1 if x is true, otherwise 0. Mean that we only take the discounted gain into account $\frac{1}{\log(1+\hat{z}_{u,i})}$ when ranking position within the top-K recommendations. $MDG_i = 0$ denote the item i is never recommended. $MDG_i = 1$ denote the item i is ranked at the top position to match all user.

Rawlsian Max-Min fairness requires the model necessitates maximizing the minimum utility for the group to ensure that no object is overlooked or undervalued.. Unlike fairness concepts [28 - 35] these eliminate differences between groups but ignore disincentives for those who are better served. Max-min fairness does not require reducing the utility of the objects served but on the contrary it accepts inequity. Max-min fairness is favored in scenarios that do not require absolute equality, like recommendation tasks, as it can also help maintain the overall utility of our study's model. We are quite suitable for its application.

By apply Rawlsian Max-Min fairness to the MDG metric. We can determine the average MDG of $t\%$ worst-off items, which includes the $t\%$ item with lowest MDG from testing and the highest MDG for best-served items. Our report denoted as MDG-min 10%, MDG-min 20% and MDG-max 10%.

Chapter 3

PROJECT MANAGEMENT PLAN

Project Timeline

Throughout the 15-week process we divided the work into three phases:

Preparation: We determine project goals, scope, resources, management methods and division of project tasks. We first determine what our research question is? What results do we want to achieve? Next we read research papers on the issues that we have oriented based on the research question. Once we identify those critical issues, we can easily learn and identify the relevant models and datasets that will be used throughout the project without deviating from our goals. Setting the foundation for the entire project and ensuring alignment among all team members are crucial during this phase.

Development: We choose the proposed models based on the fact that they must be able to run the cold start problem with items based on the articles we have researched. Next focus on running those base models. In parallel with running those base models, we also research our new method with the input of the method being the output of the base model recommender. The purpose of those methods is to increase fairness for users. We will not mention it in this entry. Once satisfied with the model's performance, we deployed it to the web API we developed to share our work with the world and demonstrate its performance.

Wrapping up: We focus on completing all remaining documents, project timelines and deliverables. We also prepared a final presentation of our work, which showcased the achievements and values of the project.

Going deeper into our weekly project planning, important tasks and milestones have been broken down into smaller, more manageable chunks to ensure things get done consistently and smoothly. During the first week, we focused on defining the project's goals by formulating research questions, identifying relevant models and data sets, and conducting a review of the research literature. . We also develop a basic project plan, identify the management tools that will be used, and establish a timeline for the project. During the second week, we continued our literature review and began reading research papers in parallel with searching for data sets. This gave us a deeper understanding of the techniques used by people in their research papers which enabled us to inform our approach to the project. During the third week, we configured a suitable environment to train the model while continuing to explore the research on the technique used. This involves setting up the necessary hardware and software tools, as well as developing a plan for how to deploy and optimize the model. By the end of the third week, we had a solid foundation for the project, including a clear understanding of the goals, a clear plan, and the right environment to develop the model. From weeks 4 to 12, we enter a continuous training and improvement loop in which we continuously train our method on models and data sets and collect results output to improve our methods so that they are most effective. This process allows us to refine our method and ensure that it increases the effectiveness of the user recommendation process without any unfairness. During weeks 8 to 10, while the training round is still going on, we focus on learning about API deployment tools, which allow us to deploy our model as a web service and deploy it as an e-commerce platform. Finally, in weeks 10 to 12, we tested the model on the website we developed, using what we considered the best performing model to test the site. Is the website operating stably or not? This gives us the opportunity to see how well our method works in a real-world situation and identify any issues that require further improvement. At week 13, we finished coding and began preparing the final report. We reviewed the progress of the project, identified areas for improvement, and gathered all the necessary documents for the report. We also submitted all relevant resources after ensuring that our work was formatted correctly. In week 14, we practice our final report, practice our presentation, and make sure we can communicate our findings and results effectively. Finally, in week 15, we demonstrated our method via the website, discussed the results, and answered questions from the audience. A summary of the weekly plan can be seen in Table 3.1 and specific work of each member in Table 3.2.

Table 3.1 Weekly Project Plan.

Week	Objectives
1	Determine goals by asking research questions, searching for related research on Risk-awareness and recommender systems. Develop basic project plans.
2	Read research papers in parallel with searching for relevant datasets.
3	Configure the environment suitable for model training while looking for more research on inference techniques.
4 - 12	Training and improvement loop in which we continuously train our method on models and data sets and collect results output to improve our methods so that they are most effective.
8 - 10	Learning about API deployment tools.
10 - 12	Tested the model on the website developed.
13	Finished coding and began preparing the final report.
14	Practice present for final report.
15	Present our work.

Table 3.2 Weekly Task Assignment.

Week	Nguyen Trong Quoc Dat	Nguyen Nhat Minh	Tran Le Anh Tuan
1	Determine the objectives, and find related research.	Determine the objectives, and find related research.	Determine the objectives, and find related research.
2	Read research papers.	Read research papers.	Find relevant datasets.
3	Research re ranking method. Find new method for reranking. Configure the environment suitable for model training. Prepare slide for review 1.	Research metric for fairness problem. Configure the environment suitable for model training. Prepare slide for review 1.	Research what is cold start, warm start problem. Configure the environment suitable for model training. Prepare slide for review 1.
4	Run base model Heater.	Run base model LightFM.	Run base model GoRec.
5	Run method re-ranking on base model Heater.	Run method re-ranking on base model LightFM.	Run base model GoRec apply amazon datasets
6	Suggest idea for Inverse Propensity Weight method	Run base model lighFM with method Inverse propensity weight	Write Acknowledgements
7	Relax Linear Program with Python-MIP for optimization	Learn and run first step of front-end	Learn and run first step of back-end
8	Run Amazon datasets for base model, and create new method for re ranking.	Build interface for demo, and help Dat run datasets amazon.	Build SQL database for demo.
9	Experiment with contrastive learning. Prepare slide for review 2.	Write Introduction, Related work, Project management plan in report. Prepare slide for review 2.	Combine front-end with back-end. Prepare slide for review 2.
10	Run base model CLCRec. Format data	Write Materials and Methods.	Create function front-end: add, delete item, rating item.

11	Tesing new method, metric. Prepare slide for review 3.	Run base model Vae CF. Prepare slide for review 3.	Write Deployment: Interface, database, API. Prepare slide for review 3.
12	Run base model CCFCRec. Write Result, Appendix.	Run Heater model after format tensorflow 2. Write Discussion, Reference.	Run web demo. Write Conclusion.
13	Finish project, prepare slide for final report.	Finish project, prepare slide for final report.	Finish project, prepare slide for final report.
14	Practice present for final report.	Practice present for final report.	Practice present for final report.
15	Present our work.	Present our work.	Present our work.

Chapter 4

MATERIAL AND METHODS

4.1 Materials

4.1.1 Hardware

Google Colaboratory

Google Colaboratory, is a free, cloud-based platform provided by Google that allows users to write and execute Python code collaboratively in a Jupyter Notebook environment. One of the significant advantages of using Google Colab is that it provides free access to Graphics Processing Units (GPUs). This is particularly useful for machine learning tasks that involve heavy computational workloads, as these accelerators can significantly speed up the training of deep learning models. Also, Colab comes with several pre-installed libraries commonly used in data science and machine learning, such as TensorFlow, PyTorch, Keras, OpenCV, and more. This makes it convenient for us to start working on their projects without worrying about installing these libraries manually. However, we still face some inconvenience during the training process when using Colab. There is a limitation on GPUs and TPUs access for Colab's users. For Colab's free plan, we can only have access to the Tesla T4 GPU, which

still takes quite a long time for heavy computational workloads. Also, the provided VRAM for the Tesla T4 GPU is only 15GB, which causes us many inconveniences during the training process as for some models, they require more VRAM to train. Therefore, we have to reduce the batch size which will make the training process finish longer than we expected. Moreover, Colab for free-plan users only provides us with 6 hours of continuous usage which makes us have to come up with a solution to save our model's checkpoint so that we can continue the training process the next day (or with a different account). Despite those drawbacks we have mentioned, Google Colab is still a great, powerful and useful platform for our model training process.

4.1.2 Data Management

Google Drive

Google Drive, a cloud-based service developed by Google, serves as a platform for file storage and synchronization. Google Drive provides users with free cloud storage space to store files such as documents, images, videos, and more. The free storage limit is typically generous, and additional storage can be purchased if needed. We use Google Drive as a platform for storing our management plan, our documents, slides for reviewing, v.v.

4.1.3 Libraries And Frameworks

PyTorch

PyTorch, an open-source machine learning library, was created by Facebook's AI Research lab (FAIR). It is widely used for deep learning and artificial intelligence applications. PyTorch is known for its dynamic computational graph, which makes it particularly well-suited for research and experimentation. PyTorch offers various libraries and tools to support different aspects of machine learning and deep learning. We used some functions of Pytorch to perform many parts in both our training phase and deployment phase.

TensorFlow

TensorFlow is an end-to-end platform for machine learning. The ecosystem is built on the Core framework that streamlines model building, training, and export. TensorFlow supports distributed training, immediate model iteration, and easy debugging with Keras, etc.

FastAPI

FastAPI is a modern, fast (as the name implies), web framework for building APIs. FastAPI automatically generates interactive API documentation (Open API) based on the Python type hints used in the code. This helps developers understand the API's structure and use it effectively. Also, FastAPI is designed for performance and utilizes asynchronous programming to handle multiple requests efficiently. It is one of the fastest Python web frameworks, making it ideal for building high-traffic APIs. We used FastAPI to build some APIs for our deployment process.

4.2 Methods

In this section, we outline our objective to achieve fairness in recommendations. Fairness within recommendation systems encompasses various aspects, but we primarily focus on fairness for items in this work. Our previous analysis reveals that both the datasets and model results exhibit bias towards a small subset of popular items, neglecting the majority.

Therefore, our goal is to ensure that even the least popular items are recommended to users. To evaluate this fairness, we utilize the Mean Discount Gain (MDG) metric (discussed in Section 2.4).

However, relying solely on the MDG metric presents a challenge. It's possible that while addressing item fairness, the majority of items might be recommended only to a small group of users. This would create unfairness in terms of user exposure. To mitigate this, we introduce a new metric designed to measure the distribution of item recommendations across users, ensuring a more equitable exposure.

To address this challenge, we also propose new methods, first convert the problem to Mixed-Integer Linear Programming Problems. And then using two type of algorithm to solve these problem, one type of algorithm for the smaller dataset which offer more performance and one for the larger dataset but worse in performance.

4.2.1 Datasets

Amazon Review Datasets. Amazon Review contains the product reivews and metadata from Amazon, it includes 142.8 million reviews spanning May 1996 – July 2014. Here is some information about review data and metadata:

- Review data includes: reviewerID, asin, reviewerName, helpful, reviewText, overall, summary, unixReviewTime, reviewTime.
- Metadata includes: asin, title, price, imUrl, related(also_bought, also_viewed, bought_together), salesRank, brand, categories.

To make it easier to imagine, we will take specific examples inside datasets.

For reviews data:

- reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- asin - ID of the product, e.g. 0000013714
- reviewerName - name of the reviewer
- helpful - helpfulness rating of the review, e.g. 2/3
- reviewText - text of the review
- overall - rating of the product
- summary - summary of the review
- unixReviewTime - time of the review (unix time)
- reviewTime - time of the review (raw)

For metadata:

- asin - ID of the product, e.g. 0000013714
- title - name of the product
- price - price in US dollars (at time of crawl)
- imUrl - url of the product image

- related - related products (also bought, also viewed, bought together, buy after viewing)
- salesRank - sales rank information
- brand - brand name
- categories - list of categories the product belongs to

MovieLens 1M. MovieLens 1M movie ratings. Stable benchmark dataset. One million ratings from anonymous 6000 users who joined MovieLens in 2000 over 4000 movies. They separate them into three files: ratings.dat, users.dat, movies.dat, we will go deeper into each file:

For ratings file:

UserID::MovieID::Rating::Timestamp

- UserIDs range between 1 and 6040
- MovieIDs range between 1 and 3952
- Ratings are assigned on a 5-star scale, with only whole-star ratings allowed.
- Timestamp is represented in seconds since the epoch as returned by time(2)
- Each user has at least 20 ratings

For users file:

UserID::Gender::Age::Occupation::Zip-code

- Gender is denoted by a "M" for male and "F" for female
- Age is chosen from the following ranges:

- * 1: "Under 18"
- * 18: "18-24"
- * 25: "25-34"
- * 35: "35-44"
- * 45: "45-49"
- * 50: "50-55"
- * 56: "56+"

- Occupation is chosen from the following choices:

- * 0: "other" or not specified
- * 1: "academic/educator"
- * 2: "artist"
- * 3: "clerical/admin"
- * 4: "college/grad student"
- * 5: "customer service"
- * 6: "doctor/health care"
- * 7: "executive/managerial"
- * 8: "farmer"
- * 9: "homemaker"

- * 10: "K-12 student"
- * 11: "lawyer"
- * 12: "programmer"
- * 13: "retired"
- * 14: "sales/marketing"
- * 15: "scientist"
- * 16: "self-employed"
- * 17: "technician/engineer"
- * 18: "tradesman/craftsman"
- * 19: "unemployed"
- * 20: "writer"

For movies file:

MovieID::Title::Genres

- Titles match exactly with the titles listed on IMDB (including the release year).
- Genres are pipe-separated and are selected from the following genres:
 - * Action
 - * Adventure
 - * Animation
 - * Children's
 - * Comedy
 - * Crime
 - * Documentary
 - * Drama
 - * Fantasy
 - * Film-Noir
 - * Horror
 - * Musical
 - * Mystery
 - * Romance
 - * Sci-Fi
 - * Thriller
 - * War
 - * Western

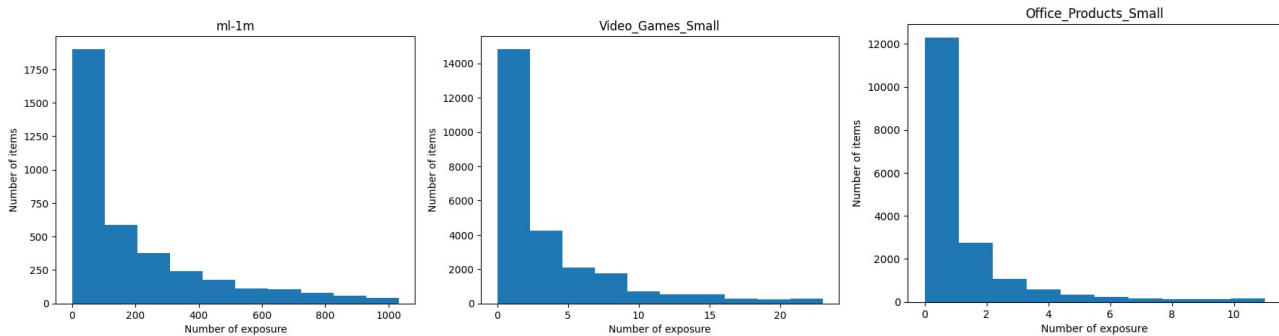
4.2.1.1 Statistic

In this article, we utilize three datasets: Movielens 1M, Amazon Video Games, and Amazon Office Products. For each Amazon dataset, we have two versions: a large version using the original Amazon reviews dataset and a small, filtered version. In the small version, we first remove users with fewer than 10 interactions and then remove any items with no interactions. Filtering by user in this way helps reduce noise from users with low interaction levels. Additionally, this process is necessary for Besides, this process is necessary to testing our methods within memory limitations. The detailed statistics of the dataset are shown in Table 4.1.

Table 4.1: Summary of the dataset [(S) is notation for small version]

<i>Dataset Name</i>	<i>Users</i>	<i>Items</i>	<i>Interactions</i>	<i>Density</i>
<i>Movielens</i>	6040	3883	1000209	4.2%
<i>Video Games (S)</i>	8057	26729	157494	0,07%
<i>Office Products (S)</i>	3714	18823	65582	0,09%
<i>Video Games</i>	31027	33899	300003	0.02%
<i>Office Products</i>	16772	37956	145141	0.02%

For these datasets, we analyze the fairness of each dataset based on item exposure, defined as the number of interactions relative to that item. The plotted distribution reveals that the original datasets exhibit unfairness. As shown in Figure 9 and Table 4.2, except for the Movielens dataset, which has a high density, more than 50% of items in all other datasets have fewer than three interactions. This indicates that the majority of items have very few interactions, which may bias the recommendation results.



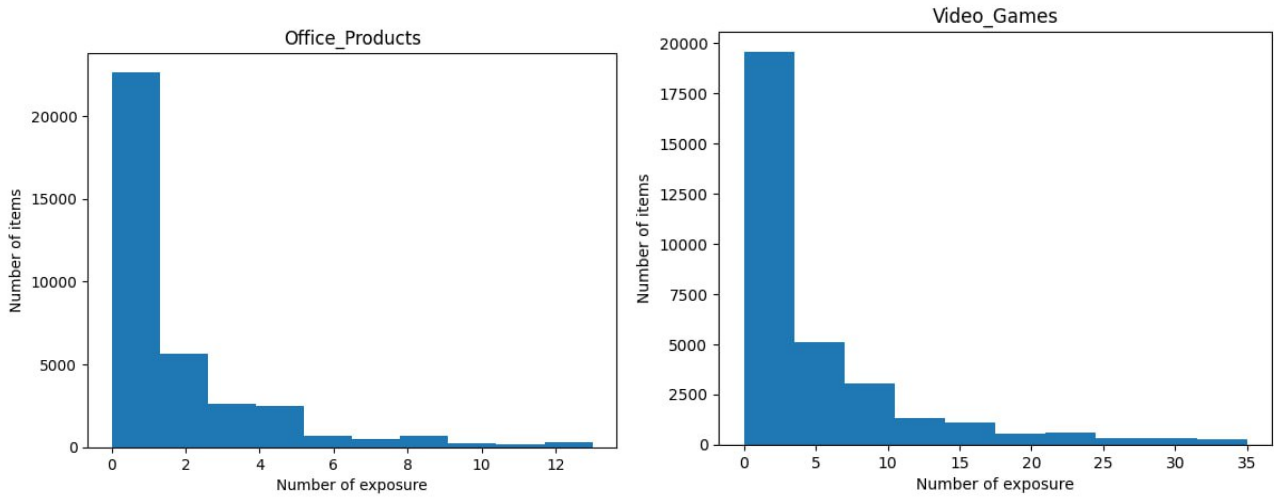


Fig. 9. The distribution of item exposure of each dataset

Table 4.2: Statistic about the exposure of item in the dataset

<i>Dataset Name</i>	<i>mean</i>	<i>std</i>	<i>min</i>	<i>25%</i>	<i>50%</i>	<i>75%</i>	<i>max</i>
<i>Movielens</i>	257.5	379.3	0	26	109	330	3428
<i>Video Games (S)</i>	5.89	12.6	1	1	2	5	323
<i>Office Products (S)</i>	3.48	9.9	1	1	1	2	263
<i>Video Games</i>	8.84	22.6	1	1	3	7	847
<i>Office Products</i>	3.82	10.5	1	1	1	3	318

4.2.1.2 Pre-processing

To prepare our datasets for a cold-start recommendation model that primarily relies on content-based collaborative filtering, we extract three types of features: text, image, and multi-hot.

For text feature vectors, we employ the MiniLM model [47] with the pretrained "all-MiniLM-L6-v2" from the Sentence Transformers library on Hugging Face. For each product and movie, we concatenate the title, description, and category into a single sentence, which is then input into the model to generate a 384-dimensional text feature vector.

For image feature vectors, we utilize a pretrained ResNet50 model [49] to obtain a 4096-dimensional vector representation.

For multi-hot features, we construct a vector based on the categories of each item. The dimension of this vector is equal to the total number of unique categories in the dataset. Each element in the vector corresponds to a specific category, with a value of 1 if the item belongs to that category and 0 otherwise.

Tabel 4.3: The dimension of features vectors in image, text, multi-hot

<i>Dataset Name</i>	<i>image</i>	<i>text</i>	<i>multi-hot</i>
<i>Movielens</i>	4096	384	18
<i>Video Games</i>	4096	384	413
<i>Office Products</i>	4096	384	793

For each dataset, we random sample some items as the cold-start items and split them into the cold validation and testing sets with ratio 70 : 15 :15, which means 70% items for training, 15% for validation and 15% for testings. The validation set is used to tune the hyper-parameters and the testing set is used to evaluate the performance in the experiments.

Table 4.4: Statistic for the experimental dataset

<i>Dataset Name</i>	<i>Train</i>		<i>Validation</i>		<i>Test</i>	
	Items	Interactions	Items	Interactions	Items	Interactions
<i>Movielens</i>	2718	641713	466	102753	699	184441
<i>Video Games (S)</i>	18710	97831	4009	23785	4010	25007
<i>Office Products (S)</i>	13176	41017	2823	9645	2824	10362
<i>Video Games</i>	27119	216727	3390	29726	3390	29469
<i>Office Products</i>	30364	104695	3796	5816	3796	5817

4.2.2 Algorithm

In previous work [27], they used machine learning to adjust the distribution of the worst-off items to match the average distribution of all items. This was achieved by optimizing a loss function:

$$\min_{\Psi} \mathcal{L}_{\mathcal{AE}} = \sum_{i \in \mathcal{I}_{\mathcal{W}}} (|\widehat{\mathbf{R}}_{:,i} - \mathbf{R}_{:,i}|_{\text{F}} + \alpha (\text{MMD}(\overline{\mathbf{R}}, \widehat{\mathbf{R}}_{\mathbf{u}^+,i}) \cdot \delta(i \in \mathcal{I}_{\mathcal{UE}}))) + \lambda \|\Psi\|_{\text{F}}$$

However, we argue that this approach is not effective because when the difference between the predicted and actual recommendations $\left\| \widehat{R}_{:,i} - R_{:,i} \right\|_F$ increase, the MMD term also increases. This can make the model hard to converge to an optimal solution.

So that we come up with another idea, by convert the problem into the resource allocation problem, which in the form of Mixed-integer Linear Programing Problem, and then apply Dual Simplex and the Primal-dual Hybrid Gradient Method (PDHG) to solve that problem.

Problem Formulation: Let U and I be the sets of users and items, respectively. In traditional recommendations, each user, $u \in U$, is provided a list of $|I|$ items, denote as $L_{|I|}(u)$, according to some relevance score, $s \in S^{|U| \times |I|}$. Normally, it will take the K highest relevance score items for this list. Our task is to define a new binary decision matrix $W = [W_{ui}]_{|U| \times |I|}$ with value 1 when the item i is recommended to user u in the new fairness recommendation list, denote as $L_K^F(u)$. Hence, the fair top- K recommendation list to user u can represented as $[W_{u1}, W_{u2}, \dots, W_{ui}]$ where $\sum_{i \in I} W_{ui} = K$.

To ensure that even the least popular item is recommended, we must ensure that every item is recommended at least once. However, we also need to maximize relevance scores while satisfying this constraint. We will construct a fairness-aware optimization problem for the binary matrix, W , as follows:

$$\begin{aligned} & \max_{W_{ui}} \sum_{u \in U} \sum_{i \in I} S_{ui} W_{ui} \\ & s. t. \sum_{i \in I} W_{ui} = K, W_{ui} \in 0,1 \\ & \sum_{u \in U} W_{ui} \geq \alpha \end{aligned}$$

The solution above will recommend exactly K items to each user while maximizing the total preference score and ensuring each item is recommended at least α times. The hyperparameter α will be chosen based on the specific application and will assign a weight to the fairness constraint relative to user preferences.

To solve this problem, we use two well-known algorithms for solving Mixed-Integer Programming (MIP) problems:

Dual Simplex. The Simplex algorithm is widely used in Linear Programming, optimizing by traversing feasible region vertices for the best solution. Meanwhile, the Dual Simplex algorithm is efficient for fixed doubly feasible regions and commonly used in MIP solvers. Why do we use dual simplex here because it works very well in practice. It's easier to find initial dual feasible solutions and the dual LP is often less degenerate.

PDLP. a practical first-order method for linear programming this method uses the primal-dual hybrid gradient method (PDHG) as the base algorithm. PDLP addresses a more general form of LP:

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & c^T x \\ \text{s.t.} \quad & Gx \geq h \\ & Ax = b \\ & l \leq x \leq u \end{aligned}$$

Where $G \in \mathbb{R}^{m_1 \times n}$, $A \in \mathbb{R}^{m_2 \times n}$, $c \in \mathbb{R}^n$, $h \in \mathbb{R}^{m_1}$, $b \in \mathbb{R}^{m_2}$, $l \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{\infty\})^n$. The problem of primal dual becomes:

$$\min_{x \in X} \max_{y \in Y} L(x, y) := c^T x - y^T Kx + q^T y$$

Where $K^T = (G^T, A^T)$ and $q^T := (h^T, b^T)$, $X := \{x \in \mathbb{R}^N : l \leq x \leq u\}$ and $Y := \{y \in \mathbb{R}^{m_1 \times m_2} : y_{1:m_1} \geq 0\}$. In PDLP, the primal and the dual are reparameterized as

$$\eta = \frac{s}{\omega}, \quad \sigma = s\omega \quad \text{with } s, \omega > 0$$

Where s controls the scale of the step-size, and ω is a primal weight controls the balance between the primal and the dual vairables.

4.2.2.1 U-MMF Method

The problem with solving a Mixed-Integer Programming (MIP) problem, as formulated above, is that it requires significant time and memory resources, making it impractical for large datasets. This limitation motivates our development of a second, more scalable method. The second method builds upon the approach in [48], which aimed to improve fairness for item providers.

Online propose of P-MMF.

In real-time recommendations, most users will approach the system in a sequential manner, such that if user u_t comes into contact with the system at time t , the system will have to consider the recommendation for the user consumed during the entire period of use by this person $t = 0$ to T . Applying provider equity, it can be thought of as the total number of the supplier's items exposed to consumption during the period time from 0 to T . This can be seen as a trade-off between user preference and group fairness, which is expressed through the formula:

$$\begin{aligned} \max_{L_K^F} \quad & \frac{1}{T} \sum_{t=1}^T f(L_K^F(u_t) + \lambda r(e)) \\ \text{s.t.} \quad & e \leq \gamma \end{aligned}$$

Where $L_K^F(u) \in I^K$ is a new fair list, $f(L_K^F(u))$ donates a balance of user advantage, $\gamma \in \mathbb{R}^{|\mathcal{P}|}$ represents the weight of different group such as PF or MMF [46] and metric of provider fairness e . Base on PF and MMF [46] $r(e)$ has two type:

- Proportion Fairness (PF): $r(e) = \sum_{p \in \mathcal{P}} \log [1 + e_p / \gamma_p]$
- Max-min Fairness (MMF): $r(e) = \min_{p \in \mathcal{P}} [e_p / \gamma_p]$

Returning to the fair recommendation issue, it can be formulated as a linear programming problem:

$$\begin{aligned} \max_{x_t} \quad & \frac{1}{T} \sum_{t=1}^T g(x_t) + \lambda r(e) \\ \text{s.t.} \quad & \sum_{i \in I} x_{ti} = K, \quad \forall t \in [1, 2, \dots, T] \end{aligned}$$

$$e_p = \sum_{t=1}^T \sum_{i \in I} x_{ti}, \quad \forall p \in \mathcal{P}$$

$$g(x_t) = \sum_{i \in I} x_{ti} s_{u_t, i}, \quad \forall t \in [1, 2, \dots, T]$$

$$e \leq \gamma, x_{ti} \in \{0, 1\}, \quad \forall i \in I, \quad \forall t \in [1, 2, \dots, T]$$

Where $x_t \in \{0, 1\}^{|I|}$ represents the decision vector of user u_t . $g(\cdot)$ is the function of user-site utility. For each item i , x_{ti} equals 1 when it included in the fair ranking list, otherwise it equals to 0.

The linear programming solution mentioned earlier is primarily suitable for resolving small-scale problems offline. However, in online systems, when a user interacts with the model, there is an urgent requirement to swiftly create a detailed ranking list from a vast array of items. This means that additional data may not be accessible after a specific period of user interaction. To address this, our approach involves merging the dual problem with the momentum gradient descent algorithm for efficient online learning.

Max-min fairness in online scenarios.

$$W = \frac{1}{T} \sum_{t=1}^T g(x_t) + \lambda \min[e/\gamma]$$

Where $\min[e/\gamma]$ relates to the fairness regularizer that maximizes the minimum. To achieve minimal regret, algorithm h should perform well. Define W_{OPT} as the optimal value, which assesses regret by comparing the expected performance W_{OPT} of the offline issue with that of the online algorithm W under user distributions \mathcal{U} :

$$Regret(h) = \mathbb{E}_{u_t \sim \mathcal{U}}[W_{OPT} - W]$$

According to the issue of LP formula. When we combine dual problems can reduce its computational cost. Dual problem of Equation:

$$W_{OPT} \leq W_{Dual} = \min_{\mu \in \mathcal{D}} [g^*(A\mu) + \lambda r^*(-\mu)]$$

Where $A \in \mathbb{R}^{|I| \times |P|}$ represents the item-provider adjacent matrix, $A_{ip} = 1$ indicates item $i \in I_p$, otherwise is 0. $g^*(\cdot)$, $r^*(\cdot)$ are conjugate functions:

$$g^*(c) = \max_{x_t \in X} \sum_{t=1}^T \left[\frac{g(x_t)}{T} - c^T x_t \right], \quad r^*(-\mu) = \max_{e \leq \gamma} [r(e) + \mu^T e / \lambda],$$

And $\mathcal{D} = \{\mu | r^*(-\mu) < \infty\}$ is the feasible region of dual variable μ . The theorem proof in Appendix A.1 of paper P-MMF [48]

The P-MMF algorithm operates by computing the proposed variable x_t whenever a user visits, considering the remaining resources and the dual variable. The bivariate is then determined as the average bivariate for each time t : $\mu = \sum_{t=1}^T \mu_t / T$. For μ_t where the dual variable value is elevated, the algorithm will recommend fewer items from the associated supplier. We have online algorithm down here.

Algorithm: Online learning of P-MMF

Input: User arriving order $\{u_i\}_{i=1}^N$, time-separate size T , ranking size K , user-item preference score $s_{u,i}$, $\forall u, i$, item-provider adjacent matrix A , maximum resources γ and the trade-off coefficient λ .

Output: The decision variables $\{x_i, i = 1, 2, \dots, N\}$

1. **for** $n = 1, \dots, N/T$ **do**
2. Initialize dual solution $\mu_1 = 0$, remaining resources $\beta_1 = \gamma$, and momentum gradient $g_0 = 0$
3. **for** $t = 1, \dots, T$ **do**
4. User u_{nT+t} arrives
5. $m_p = \begin{cases} 0, & \beta_{tp} > 0 \\ \infty, & \text{otherwise} \end{cases}$
6. //make the recommendation:
7. $x_t = \operatorname{argmax}_{x_t \in X} [g(x_t)/T - (A(\mu_t + m))^T x_t]$
8. $\beta_{t+1} = \beta_t - A^T x_t$
9. $e_t = \operatorname{argmax}_{e_t \leq \beta_t} r * (-\mu_t)$
10. $\tilde{g}_t = -A^T x_t + e_t$
11. $g_t = \alpha \tilde{g}_t + (1 - \alpha) g_{t-1}$
12. $\mu_{t+1} = \operatorname{argmax}_{\mu \in \mathcal{D}} [\langle g_t, \mu \rangle + \eta \|\mu - \mu_t\|_{y^2}^2]$
13. **end for**
14. **end for**

Assume that the function $\|\cdot\|_{y^2}^2$ is σ -strong convex and there exists a constraint $G \in \mathbb{R}^+$ and $H > 0$ such that $\|\tilde{g}_t\| < G$, $\|\mu_t - \mu_0\| \leq H$. Next, the regret can be limited as shown below:

$$\text{Regret}(h) \leq \frac{K(1 + \lambda\bar{r} + \bar{r})}{\min_p \gamma_p} + \frac{H}{\eta} + \frac{G^2}{(1 - \alpha)\sigma} \eta(T - 1) + \frac{G^2}{2(1 - \alpha)^2 \sigma \eta},$$

Where \bar{r} is the upper bound of MMF regularzier, and practice, $\bar{r} \leq 1$. Setting the learning rate as $\eta = O(T^{-1/2})$, It is possible to achieve a sublinear regret upper bound of approximately $O\left(T^{\frac{1}{2}}\right)$.

Out online propose of U-MMF.

Utilize the effective of the above method, to make it suitable for multiple dataset even the dataset that don't have provider. Instead of dividing items into groups based on providers, we divide them based on their exposure, similar to the new GIRS metric introduced in Section 4.2.3.

Specifically, we sort items by their exposure (how often they have been recommended) and then divide them into groups based on a predetermined proportion. This approach allows us to address fairness

concerns more directly by focusing on items with varying levels of visibility, regardless of their provider. The small proportion will result better fairness across all items.

4.2.3 Grouped Item Recommendation Spread Metric

The goal of this metric, termed Grouped Item Recommendation Spread, is to quantify the diversity of users exposed to item recommendations. This assessment is crucial for evaluating the fairness of a recommender system, as a fair system should distribute recommendations across a broad user base, not just a select few. To calculate GIRS, we first divide items into groups based on their exposure (i.e., how often they've been recommended), sorted from least to most exposed. Each group contains a predefined proportion $x\%$ of the total items.

Next, we calculate the Potential User Reach (PUR) for each item group. PUR represents the maximum number of users who could potentially be recommended items from that group, considering the total number of users and the maximum number of recommendations per user. The PUR is calculated as:

$$PUR = k|I| \frac{x}{100}$$

Where:

- $|I|$ is the number of items
- k is the maximum number of items each user can be recommended
- x the percentage of items in the group

We then determine the Actual User Reach (ACR) for each group, which is the number of unique users who received recommendations from that group.

$$ACR_p = \sum_{u \in U} \mathcal{F} \left(\sum_{i \in i_p} W_{ui} \right)$$

Where:

- $|U_p|$ is the number of actual users that group p can reach.
- U is the set of all users.
- W_{ui} is indicate whether item i is recommended to user u .
- $\mathcal{F}(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$

Finally, we compute the User Reach Proportion (URP) for each group, which is the ratio of AUR to PUR. This metric indicates how well the group's recommendations are spread across users relative to its potential reach.

$$URP_p = \frac{AUR_p}{PUR}$$

To aggregate the URP values across all groups, we use two methods:

- **Mean Proportion Fairness (MPF):** $\frac{1}{|P|} \sum_{p \in P} URP_p$
- **Max-Min Fairness (MMF):** $\min_{p \in P} URP_p$

Chapter 5

RESULT

5.1 Result of Base Model

In this section, we present the baseline model results and analyze their fairness implications.

Table 5.1: Performance comparation between CLCRec and CCFCRec

<i>Dataset Name</i>	<i>Precision</i>		<i>Recall</i>		<i>NDCG</i>	
	CLC	CCFC	CLC	CCFC	CLC	CCFC
<i>Movielens</i>	0.247	0.102	0.193	0.07	0.25	0.08
<i>Video Games (S)</i>	0.0669	0.0352	0.0669	0.0351	0.0259	0.0137
<i>Office Products (S)</i>	0.0969	0.0335	0.0969	0.0335	0.0377	0.0124
<i>Video Games</i>	0.0592	0.0529	0.0592	0.0529	0.0114	0.0099
<i>Office Products</i>	0.0494	0.0463	0.0494	0.0464	0.0099	0.0087

5.1.1 CLCRec

Similar to Section 4.2.1.1, we also use item exposure to assess fairness after applying recommendations.

As shown in Figure 10 and Table 5.1, the recommendations tend to concentrate on a small subset of items, leaving the majority with very few recommendations. Notably, over 25% of items consistently receive fewer than five recommendations, and a significant imbalance between the most and least recommended items.

The Movielens dataset demonstrates less bias in recommendation outcomes compared to the other datasets. This is because, as discussed in Section 4.2.1.1, the inherent distribution of item exposure in this dataset is more equitable. However, in real-world e-commerce scenarios, datasets often resemble the Amazon dataset more closely than the Movielens dataset, where item exposure tends to be less evenly distributed.

This underscores the challenge of ensuring fair recommendations in practical applications, as the underlying data may naturally favor certain items, leading to potential bias in the recommendation outcomes.

Table 5.1.1: Statistics on item exposure resulting from the CLCRec model in each dataset.

Dataset Name	mean	std	min	25%	50%	75%	max
Movielens	267.8	1005	0	9	18	41	5735
Video Games (S)	62.2	284.9	0	2	8	24	4764
Office Products (S)	40.7	237.8	0	0	3	11	2590
Video Games	283.7	1308.0	0	8	40	102	18744
Office Products	136.9	636.4	0	6	24	74	9496

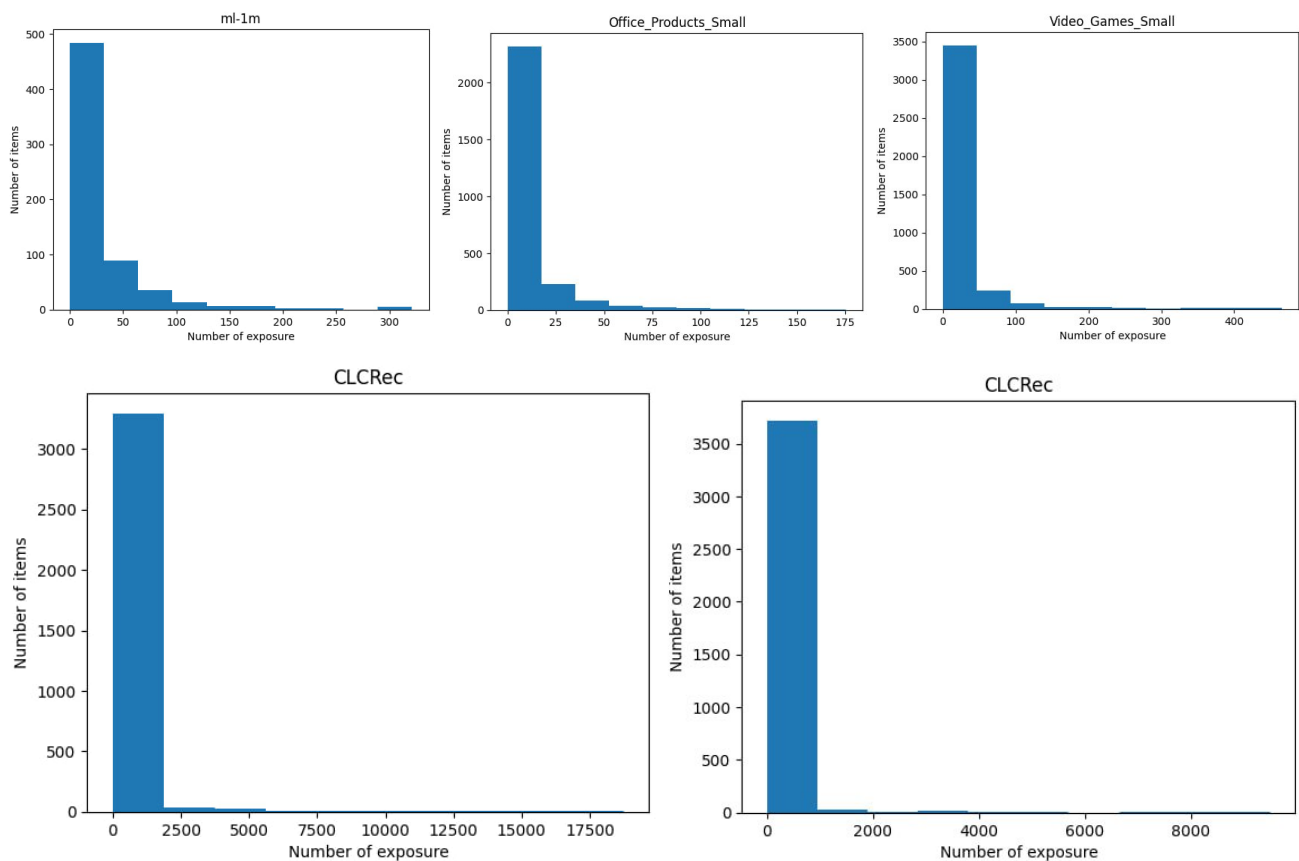


Fig. 10. The distribution of item exposure from the CLCRec result

5.1.2 CCFCRec

The CCFCRec model is unique in that it recommends users for each item, rather than the standard approach of recommending items to users. While this approach offers an advantage in terms of fairness by ensuring each item receives at least a few recommendations, it is less practical in real-world applications. Additionally, its performance, as indicated in Table 5.2, is inferior to that of the CLCRec model.

The impracticality stems from the fact that most recommendation systems are designed to suggest items to users based on their preferences and behavior. Recommending users for items doesn't align with this common use case and might not be helpful for businesses or users.

Table 5.1.2: Statistics on item exposure resulting from the CCFCRec model in each dataset.

Dataset Name	mean	std	min	25%	50%	75%	max
Movielens	267.8	352.4	7	51	118	347	1704
Video Games (S)	62.2	43.9	0	29	52	84	299
Office Products (S)	40.7	20.7	4	24	37	54	131
Video Games	283.7	183.9	8	143	241	371	1411
Office Products	136.9	89.7	6	71	117	180	813

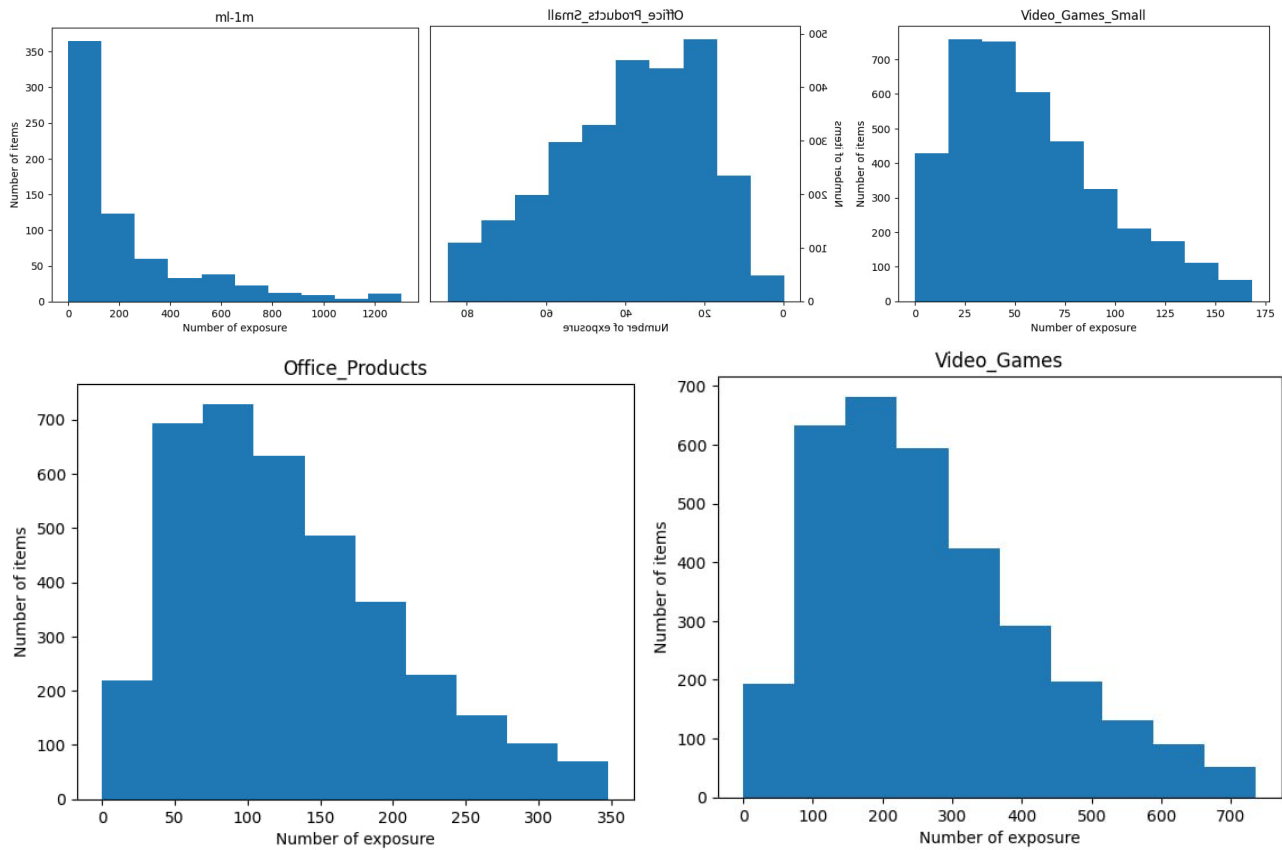


Fig. 11. The distribution of item exposure from the CCFCRec result

5.2 Experiment

We evaluate two baseline models, CLCRec and CCFCRec, on five datasets: Movielens, Amazon Video Games (small and full), and Amazon Office Products (small and full). We utilize the default hyperparameters outlined in the original papers for each model.

For CLCRec, we set the learning rate to 0.00002. We use 128 negative samples, lambda to 0.5 for movielens datasets and use 512 negative samples, lambda to 0.7 for amazons datasets. We incorporate both multi-hot and text feature vectors. The model is trained for a maximum of 100 epochs per dataset, with early stopping implemented if no improvement is observed on the validation set after 10 epochs. For CCFCRec, we employ a learning rate of 0.0005, tau of 0.1, lambda of 0.7 for amazons and 0.5 for movielens similar to CLCRec, 40 negative samples, and an attribute dimension of 256. In CCFCRec, we choose positive 5 for amazons because the density too low, we don't have enough user interactions to item and 10 for movielens because of the density. Training is performed for 50 epochs on each dataset.

For the Dual Simplex (MIP) and PDLP algorithms, we utilize the Google OR-Tools library. For the U-MMF method, we leverage the CVXPY library to solve the sub-linear problem. The U-MMF parameters are set as follows: lambda = 0.1, alpha = 0.1, eta = 0.1, and the group proportion (p) is set to 0.05.

Table 5.2: The hyperparameters for training model.

<i>Base model</i>	<i>Datasets</i>	<i>learning rate</i>	<i>negative sample</i>	<i>positive sample</i>	<i>lamda</i>	<i>epochs</i>
<i>CLCRec</i>	Amazon	0.00002	512	1	0.7	100
	Movielens	0.00002	128	1	0.5	100
<i>CCFCRec</i>	Amazon	0.00005	40	5	0.7	50
	Movielens	0.00005	40	10	0.5	50

5.3 Result Of Method

At Table 5.3, the time for each method and algorithm shows that the CCFC model nearly optimizes fairness, resulting in faster processing times for all methods compared to the CLC model. This observation is similar to the MovieLens dataset, which also demonstrates greater fairness than other datasets.

In Table 5.4, showing the performance of each method in every dataset, we see that the Dual Simplex (MIP) retains the highest performance but takes the slowest runtime. In contrast, the two methods, PDLP and UMMF, offer better runtimes but are less efficient in retaining the performance of the base model.

Contrasting with performance, PDLP shows that it has the best performance in both MDG, MMF, and MPF metrics, which is easy to understand because it sacrifices the most performance. As for UMMF, although it can work on large datasets, it is only effective with MMF and MPF metrics, which is also easy to understand because those are the main targets of the method. With MIP, although it runs slower and also can't run on large datasets, it improves fairness performance in all metrics and also preserves the performance of the base model.

Table 5.3: The time consuming of each algorithm for re-ranking methods.

Model Name	Time (second)				
	MovieLens	Video Games (S)	Office Products (S)	Videos Games	Office Products
CLCRec + PDLP	116	1031	309	x	x
CLCRec + MIP	459	68383	7786	x	x
CLCRec + UMMF	292	401	185	1518	827
CCFCRec + PDLP	110	1004	600	x	x
CCFCRec + MIP	152	1275	368	x	x
CCFCRec + UMMF	292	405	183	1526	827

Table 5.4a: The recommendations performance of all base model and methods in dataset MovieLens

Model Name	Precision@30	Recall@30	NDCG@30
CLCRec	0.247	0.193	0.203
CLCRec + PDLP	0.060	0.042	0.051
CLCRec + MIP	0.246	0.193	0.203
CLCRec + UMMF	0.142	0.101	0.125
CCFCRec	0.102	0.073	0.088
CCFCRec + PDLP	0.060	0.043	0.052
CCFCRec + MIP	0.102	0.073	0.088
CCFCRec + UMMF	0.111	0.080	0.096

Table 5.4b: The recommendations performance of all base model and methods in dataset Video Games

Model Name	Precision@30	Recall@30	NDCG@30
CLCRec	0.059	0.059	0.011

CLCRec + UMMF	0.021	0.021	0.004
CCFCRec	0.053	0.053	0.010
CCFCRec + UMMF	0.050	0.050	0.009

Table 5.4c: The recommendations performance of all base model and methods in dataset Video Games Small

Model Name	Precision@30	Recall@30	NDCG@30
CLCRec	0.066	0.066	0.025
CLCRec + PDLP	0.007	0.007	0.002
CLCRec + MIP	0.066	0.066	0.025
CLCRec + UMMF	0.048	0.048	0.019
CCFCRec	0.035	0.035	0.013
CCFCRec + PDLP	0.007	0.007	0.003
CCFCRec + MIP	0.035	0.035	0.013
CCFCRec + UMMF	0.033	0.033	0.013

Table 5.4d: The recommendations performance of all base model and methods in dataset Office Products

Model Name	Precision@30	Recall@30	NDCG@30
CLCRec	0.049	0.049	0.009
CLCRec + UMMF	0.017	0.017	0.003
CCFCRec	0.046	0.046	0.008
CCFCRec + UMMF	0.043	0.043	0.008

Table 5.4e: The recommendations performance of all base model and methods in dataset Office Products Small

Model Name	Precision@30	Recall@30	NDCG@30
CLCRec	0.096	0.096	0.037
CLCRec + PDLP	0.010	0.010	0.004

CLCRec + MIP	0.095	0.095	0.037
CLCRec + UMMF	0.049	0.049	0.019
CCFCRec	0.033	0.033	0.012
CCFCRec + PDLP	0.013	0.013	0.005
CCFCRec + MIP	0.033	0.033	0.012
CCFCRec + UMMF	0.033	0.033	0.012

Table 5.5a: The recommendations performance of all base model and methods in dataset MovieLens

Model Name	MDG _{min@10}	MDG _{min@20}	MDG _{max@10}	MDG _{max@20}
CLCRec	0.00009	0.0001	0.1228	0.0629
CLCRec + PDLP	0.0310	0.0368	0.0429	0.0428
CLCRec + MIP	0.0007	0.0007	0.1224	0.1224
CLCRec + UMMF	0.0045	0.0007	0.1336	0.1336
CCFCRec	0.0007	0.0010	0.0626	0.0626
CCFCRec + PDLP	0.0375	0.0388	0.0427	0.0427
CCFCRec + MIP	0.0009	0.0011	0.0631	0.0631
CCFCRec + UMMF	0.00003	0.0004	0.0782	0.0782

Table 5.5b: The recommendations performance of all base model and methods in dataset Video Games

Model Name	MDG _{min@10}	MDG _{min@20}	MDG _{max@10}	MDG _{max@20}
CLCRec	0.0017	0.0424	0.022	0.012
CLCRec + UMMF	0.0005	0.0014	0.021	0.017
CCFCRec	0.0005	0.0007	0.007	0.006
CCFCRec + UMMF	0.0005	0.0007	0.009	0.008

Table 5.5c: The recommendations performance of all base model and methods in dataset Video Games Small

Model Name	MDG _{min@10}	MDG _{min@20}	MDG _{max@10}	MDG _{max@20}
CLCRec	0	0.0003	0.019	0.01
CLCRec + PDLP	0.007	0.007	0.007	0.007
CLCRec + MIP	0.0005	0.0006	0.018	0.0099

CLCRec + UMMF	0.0003	0.0006	0.257	0.017
CCFCRec	0.0002	0.0004	0.006	0.005
CCFCRec + PDLF	0.007	0.0073	0.007	0.007
CCFCRec + MIP	0.0005	0.0007	0.006	0.005
CCFCRec + UMMF	0.0004	0.0006	0.008	0.006

Table 5.5d: The recommendations performance of all base model and methods in dataset Office Products

Model Name	MDG _{min@10}	MDG _{min@20}	MDG _{max@10}	MDG _{max@20}
CLCRec	0.0003	0.0018	0.019	0.01
CLCRec + UMMF	0.0005	0.0011	0.021	0.016
CCFCRec	0.0004	0.0006	0.006	0.005
CCFCRec + UMMF	0.0005	0.0007	0.008	0.006

Table 5.5e: The recommendations performance of all base model and methods in dataset Office Products Small

Model Name	MDG _{min@10}	MDG _{min@20}	MDG _{max@10}	MDG _{max@20}
CLCRec	0	0	0.029	0.015
CLCRec + PDLF	0.009	0.01	0.01	0.01
CLCRec + MIP	0.001	0.001	0.024	0.014
CLCRec + UMMF	0.0008	0.001	0.037	0.024
CCFCRec	0.0008	0.0011	0.006	0.006
CCFCRec + PDLF	0.01	0.01	0.01	0.010
CCFCRec + MIP	0.001	0.0014	0.007	0.006
CCFCRec + UMMF	0.0009	0.0014	0.009	0.008

Table 5.6a: The recommendations performance of all base model and methods in dataset Movielens

Model Name	MMF@0.05	MFF@0.10	MFP@0.05	MFP@0.1
CLCRec	0.039	0.059	0.959	0.723
CLCRec + PDLF	5.760	2.880	5.760	2.880
CLCRec + MIP	0.286	0.208	0.993	0.749
CLCRec + UMMF	0.123	0.256	2.532	1.588

CCFCRec	0.210	0.235	2.279	1.358
CCFCRec + PDLP	5.755	2.877	5.756	2.878
CCFCRec + MIP	0.226	0.239	2.282	1.355
CCFCRec + UMMF	0	0.015	2.523	1.498

Table 5.6b: The recommendations performance of all base model and methods in dataset Video Games

Model Name	MMF@0.05	MFF@0.10	MFP@0.05	MFP@0.1
CLCRec	0.027	0.066	1.537	0.942
CLCRec + UMMF	0.205	0.388	3.560	2.279
CCFCRec	0.944	1.088	3.861	2.459
CCFCRec + UMMF	0.895	0.998	3.871	2.465

Table 5.6c: The recommendations performance of all base model and methods in dataset Video Games Small

Model Name	MMF@0.05	MFF@0.10	MFP@0.05	MFP@0.1
CLCRec	0	0	0.286	0.165
CLCRec + PDLP	1.339	0.669	1.339	0.669
CLCRec + MIP	0.269	0.186	0.385	0.243
CLCRec + UMMF	0.029	0.045	0.520	0.322
CCFCRec	0.153	0.188	0.825	0.524
CCFCRec + PDLP	1.336	0.668	1.337	0.668
CCFCRec + MIP	0.273	0.218	0.823	0.524
CCFCRec + UMMF	0.175	0.201	0.826	0.527

Table 5.6d: The recommendations performance of all base model and methods in dataset Office Product

Model Name	MMF@0.05	MFF@0.10	MFP@0.05	MFP@0.1
CLCRec	0	0.008	0.8228	0.544
CLCRec + UMMF	0.153	0.248	1.72	1.031
CCFCRec	0.494	0.235	1.808	1.065
CCFCRec + UMMF	0.388	0.224	1.829	1.078

Table 5.6e: The recommendations performance of all base model and methods in dataset Office Product Small

Model Name	MMF@0.05	MFF@0.10	MFP@0.05	MFP@0.1
CLCRec	0	0	0.152	0.108
CLCRec + PDLP	0.876	0.438	0.876	0.438
CLCRec + MIP	0.279	0.196	0.326	0.226
CLCRec + UMMF	0.029	0.056	0.328	0.206
CCFCRec	0.094	0.047	0.539	0.316
CCFCRec + PDLP	0.873	0.4366	0.873	0.436
CCFCRec + MIP	0.094	0.0472	0.537	0.315
CCFCRec + UMMF	0.097	0.048	0.548	0.323

Chapter 6

DISCUSSION

Based on the results presented in the table on Chapter 5, we will go through each datasets.

6.1 Movielens 1M

In Table 5.4a, if we compare the two models CLCRec and CCFCRec, it is clear that the CLCRec model is superior in terms of Precision when recommending related items. Furthermore, it exhibits a better Recall value, allowing users to find related items while maintaining a high NDCG value, ensuring the ranking order of items matches the user's preferences. In this Movielens dataset we will take CLCRec as a benchmark against our hybrid model, the CLCRec model shows higher stability. Among the three methods PDLP, MIP, UMMF, MIP maintains recommendation quality similar to the original model but lacks a fair penalty mechanism. Turning to Table 5.5a, PDLP significantly outperforms the original model and the other two methods. Minimum MDGs represent the least recommended items. Furthermore, an important aspect explored in this study is the accessibility of recommended items to all users. High MFF and MFP values indicate strong performance of the PDLP method on the Movielens 1M dataset, balancing model performance with fairness in item recommendation.

6.2 Video Games

Next we will access the Amazon Video Games dataset. First, due to memory limitations, we can currently only run the CLCRec, CCFCRec models along with the UMMF method combination model. In terms of Precision, Recall, it can be seen that CLCRec still performs better than CCFCRec. Looking at the MDG values next, it seems that our UMMF method performs worse than the original model when less recommended items are not promoted. But when it comes to MFF and MFP, UMMF ensures that all items are recommended equally. But we will have to trade off a few items at the bottom of the list.

6.3 Video Games Small

Due to capacity issues, we have re-filtered the video game data so it can now be run with PDLF and MIP methods. After re-filtering the data, it can be seen that CLCRec's Precision and Recall still achieve high efficiency and the NDCG value of the model is increased. That means the performance of the system has been increased and in parallel, CLCRec with the MIP method also achieved the same effect. But when looking at the MDG value, items that are little known to users are not recommended by CLCRec. On the other hand, PDLF works very well when it can recommend to users items that are less recommended and have MMF values. and PDLF's MFP also performed very well.

6.4 Office Product and Office Product Small

Similar to video games so we won't discuss it much.

Chapter 7

DEPLOYMENT

7.1 Server

For the demo deployment, we use Django, a Python web framework, to showcase our application on a local server. This allows us to easily demonstrate and test the functionality of the systems.

7.2 Database

In our Django e-commerce website, we use SQLite as our database management system. SQLite stores data in a structured format, organized into tables, rows, and columns. Each table represents a specific entity in our website, such as products, interactions, and users. We establish relationships between entities using foreign key constraints.

The database stores essential information about products, users, and interactions. For example, the Products table contains details such as product name, category, price, description, and image path. The Interactions panel links users to the products they interacted with along with timestamps.

SQLite's convenience makes it portable and easy to manage, while Django's ORM simplifies database interactions by abstracting away the underlying operations. Overall, our database supports efficient data storage, retrieval and manipulation, ensuring the smooth operation of our e-commerce platform.

7.3 Platform Flowchart

In the workflow of our e-commerce website, there is a continuous process from the initial user interaction until they receive the final result. Users start by logging in to the website, browsing products, or taking actions like adding items to the cart or placing an order. The front-end server handles these requests by providing corresponding HTML templates. Upon receiving a request, the backend server interacts with the database to retrieve product information or update the user's shopping cart. The backend server then executes business logic, such as pricing, inventory management, and user authentication. The response from the backend server contains data that is returned to the frontend server, where the data is rendered and displayed on the user's browser. This process continues as the user continues to interact with the website, creating a continuous and repetitive workflow.

7.4 UI

7.4.1 Homepage

Our home page is shown in Figure 12. In this page, we will introduce about products and the details for each of them. User can click on “View” of each products to see more.

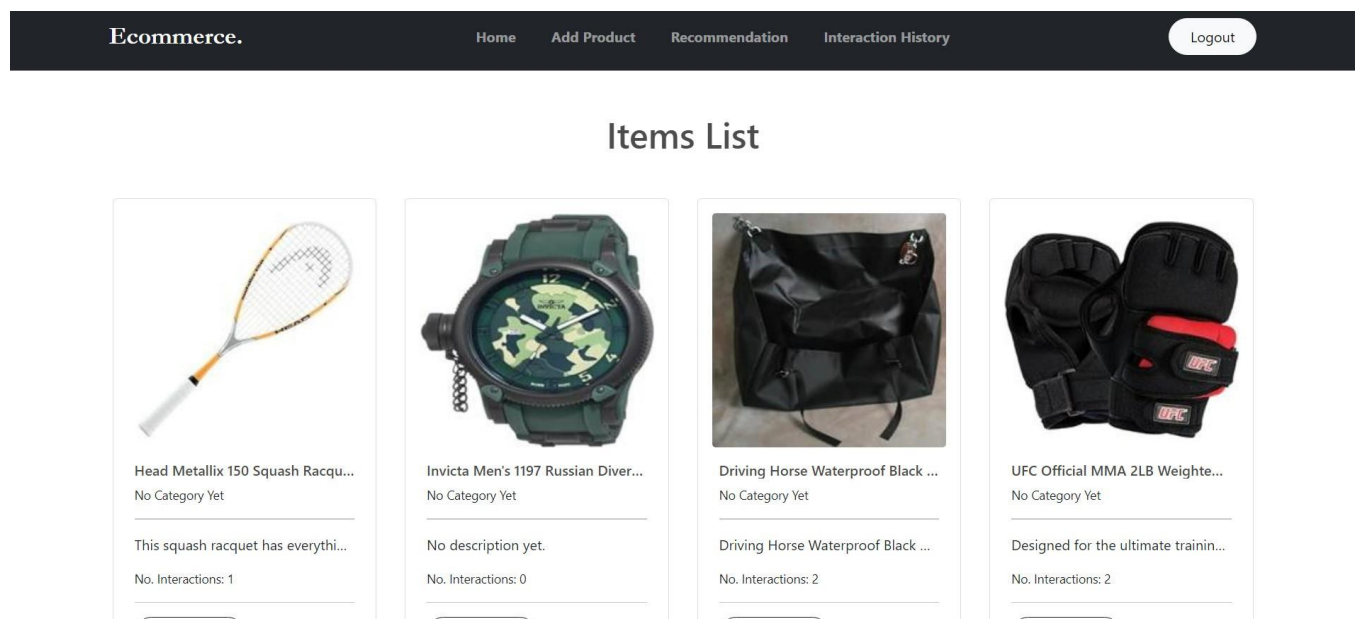
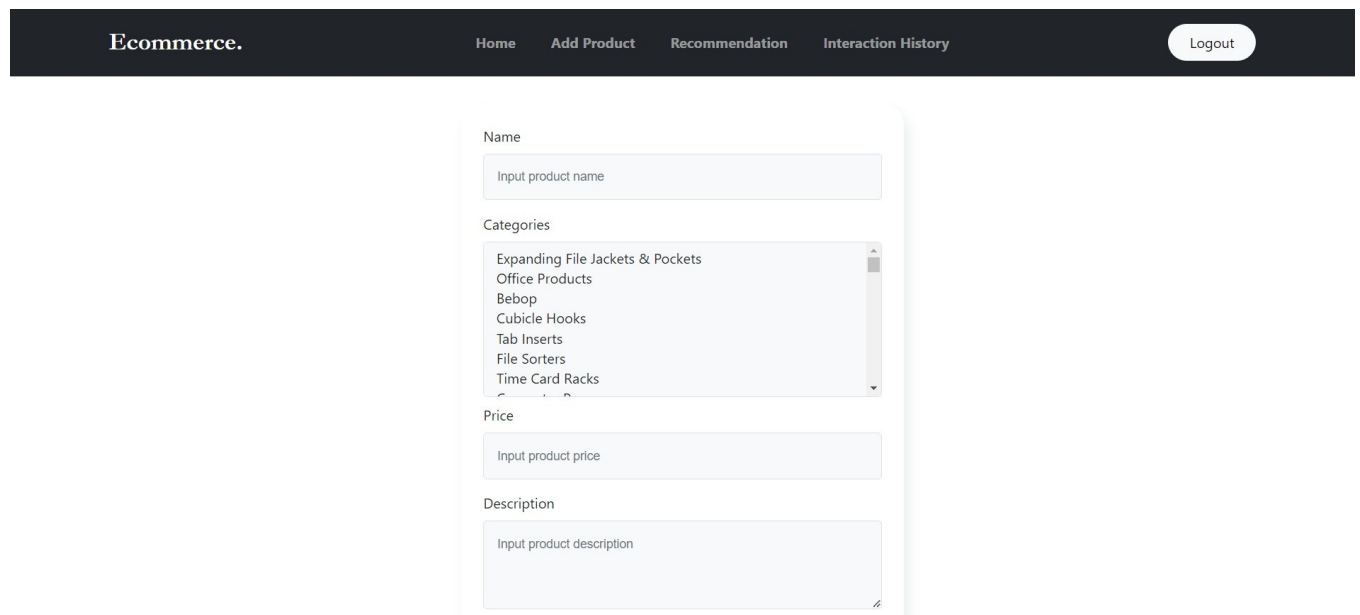


Fig. 12. Homepage of our E-commerce website

7.4.2 Add Products

On this page, providers can upload products. The providers will have to provide product Name, Categories, Price, Description. Add products page shown in Figure 13



The screenshot displays the 'Add Products' page within an Ecommerce application. The page features a dark header with the 'Ecommerce.' logo on the left and navigation links for 'Home', 'Add Product', 'Recommendation', and 'Interaction History' in the center. A 'Logout' button is positioned on the right side of the header. The main content area is a light gray box containing several input fields: a 'Name' field with the placeholder 'Input product name', a 'Categories' field with a list of options including 'Expanding File Jackets & Pockets', 'Office Products', 'Bebop', 'Cubicle Hooks', 'Tab Inserts', 'File Sorters', and 'Time Card Racks', a 'Price' field with the placeholder 'Input product price', and a 'Description' field with the placeholder 'Input product description'.

Fig. 13. Add Products page

7.4.3 Recommendation page

In this page, user can adjust the number of suggested products and the minimum interaction required for each of them. When user clicks on “Recommend” button, our model will generate on the back-end then output the results. Besides, our system will also provide a comparison between Recommendation with and without fairness. The Recommendation page will be shown in Figure 14.

Ecommerce.

[Home](#) [Add Product](#) [Recommendation](#) [Interaction History](#)

Logout


Fairness recommendations System

No. Recommend Items:


Min Exposure:

Recommend


Recommendation without Fairness




Charles Riv...
No Category ...



Civil War H...
No Category ...




UFC Officia...
No Category ...




Roxy Wom...
No Category ...


Recommendation with Fairness




Civil War H...
No Category ...



Charles Riv...
No Category ...



UFC Officia...
No Category ...



Speedo W...
No Category ...

Fig. 14. Recommendation page

7.4.4 Interaction history page

This page will display a history of products the user has interacted with. This page will be displayed in Figure 15.

Ecommerce.

[Home](#) [Add Product](#) [Recommendation](#) [Interaction History](#)

Logout

Fairness recommendations System

No. Recommend Items:

Min Exposure:

Recommend

Interaction History

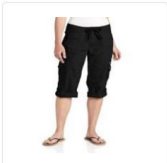





Fig. 15. Interaction history page

CONCLUSION

Through our research and experimentation, we have successfully developed a novel solution to address the fairness item problem in recommender systems. Our approach, based on a multi-index linear programming problem and two distinct algorithms, offers promising results for improving fairness while maintaining recommendation quality.

8.1 Limitations

Despite these advancements, our current solution has limitations. The most effective algorithm, while maintaining high performance, is not scalable to large datasets. Conversely, the algorithm designed for large datasets does not excel in all fairness metrics and may not fully preserve the base model's performance.

8.2 Future Developments

Future research will focus on enhancing the U-MMF method to improve its performance on the MDG metric and ensure it better preserves the base model's performance. We also aim to develop additional techniques to further enhance fairness across all metrics, particularly for large-scale recommender systems. This could involve exploring alternative optimization strategies, incorporating additional fairness constraints, or investigating new ways to balance fairness and accuracy.

Appendix A

Failed Experiments

We tried a few other approaches in addition to the primary one that was previously suggested, but they did not work for various reasons, such as not enough memory or worse outcomes. The first method yields a worse outcome by using the DPF metric rather than the additional limitation. Secondly, we employ the direct metric MDG constraint; however, its implementation is rather intricate, and our system lacks the memory to employ that technique.

We also attempted to run other baseline models, but most of them either lacked transparency in their data preprocessing steps or utilized outdated frameworks that were difficult to re-implement and reuse in our current environment.

Appendix B

Source Code And Datasets

Table B.1 Source

<i>Source</i>	
<i>All Sources</i>	https://drive.google.com/drive/folders/1MumAOuO6em8QS4aWj_N9ZsDmJXvm7C5u?usp=drive_link
<i>Process datasets</i>	https://colab.research.google.com/drive/1LPHkPIgGrNPGIV78hODz2GFkOjRh4J?usp=drive_link
<i>Statistic</i>	https://colab.research.google.com/drive/1xbBLHnH_pLSBaL5JMFaVic8vrNyuJ0?usp=drive_link
<i>CLCRec base model</i>	https://github.com/datn2107/CLCRec
<i>Run CLCRec</i>	ml_1m: https://colab.research.google.com/drive/1LGxk69XH2QCt6mILSm6-pNRsqA0KMiMw?usp=drive_link amazon: https://colab.research.google.com/drive/1UdBkOSWVvghM0OgncoZuDTdlYwtRe-Az?usp=drive_link
<i>CCFCRec base model</i>	https://github.com/datn2107/CCFCRec
<i>Run CCFCRec</i>	ml_1m: https://colab.research.google.com/drive/1IFZ0zQprXca2ZyNDs_9kKJZ8wvhuYw8m?usp=drive_link amazon: https://colab.research.google.com/drive/1SySVXwqbwt3P2bWZ6XhHNVDfuNsew0__?usp=drive_link
<i>Method</i>	https://github.com/datn2107/fairness-recommendation-system

Run Methodhttps://colab.research.google.com/drive/1rnq8fasTNSuieRooqdzQlzThoimZ7h9q?usp=drive_link***Evaluation***<https://colab.research.google.com/drive/1hCsaKX9a5tw4-HLqkPR3-bMEAtovWg5Q>

Appendix C

Import Base Model

In the process of searching for base models of the recommendation system, we acknowledged that using 5 base models, however, most of these models have trouble with pre-processing issues. We will show it in table C.1. With “-“ is don't have, “√“ is have. From best of our knowledge, only CLCRec and CCFCRec suitable for our project.

Table C.1 Errors during import of base model.

Base model	Dataset	Pre-processing	TensorFlow	User based	Item based
CLCRec	Can use Amazon and Movielens Datasets	Have pre-processing for data	Tensorflow 2	-	✓
CCFCRec	Can use Amazon and Movielens Datasets	Have pre-processing for data	Tensorflow 2	-	✓
LightFM	-	Have pre-processing for data	Tensorflow 2	✓	-
Heater	-	-	Using tensorflow 1 hard to format and save weight	-	-
GoRec	-	Have pre-processing for data	Tensorflow 2	✓	-
Vae	Requires use on large datasets	Have pre-processing for data	Tensorflow 2	✓	-

REFERENCES

1. Loren Terveen and Will Hill. Beyond Recommender Systems: Helping People Help Each Other. In *HCI In The New Millennium*, Jack Carroll, ed., Addison-Wesley, 2011.
2. Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl. Item-based collaborative filtering recommendation algorithms. *WWW '01: Proceedings of the 10th international conference on World Wide Web*. 2001. Pages 285-295.
3. G. Linden, B. Smith, J. York. Amazon.com recommendations: item-to-item collaborative filtering. In *IEEE Internet Computing*. Jan-Feb. 2003 Pages 76-80.
4. Djallel Bouneffouf. DRARS, A Dynamic Risk-Aware Recommender System. *Computation and Language [cs.CL]*. Institut National des Télécommunications, 2013. English. NNT: . tel-01026136.
5. Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H Chi, et al. 2019. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2212–2220.
6. Robin Burke. 2017. Multi Sided fairness for recommendation. *arXiv preprint arXiv:1707.00093* (2017).
7. Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2221–2231.
8. Ziwei Zhu, Xia Hu, and James Caverlee. 2018. Fairness-aware tensor-based recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1153–1162.
9. Ziwei Zhu, Jianling Wang, and James Caverlee. 2020. Measuring and Mitigating Item Under-Recommendation Bias in Personalized Ranking Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 449–458.
10. Iman Barjasteh, Rana Forsati, Farzan Masrour, Abdol-Hossein Esfahanian, and Hayder Radha. 2015. Cold-start item and user recommendation with decoupled completion and transduction. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 91–98.
11. Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. 2010. Learning Attribute-to-Feature Mappings for Cold-Start Recommendations.. In *ICDM*, Vol. 10. Citeseer, 176–185.
12. Martin Saveski and Amin Mantrach. 2014. Item cold-start recommendations: learning local collective embeddings. In *Proceedings of the 8th ACM Conference on Recommender systems*. ACM, 89–96.
13. Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 650–658.

14. Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.
15. Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From Zero-Shot Learning to Cold-Start Recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
16. Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, Lexing Xie, and Darius Braziunas. 2017. Low-rank linear cold-start recommendation from social data. In *Thirty-First AAAI Conference on Artificial Intelligence*.
17. Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*. 4957–4966.
18. Zhihui Zhou, Lilin Zhang, and Ning Yang. 2023. Contrastive Collaborative Filtering for Cold Start item Recommendation. DOI: <https://doi.org/10.1145/3543507.3583286>.
19. Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and TatSeng Chua. 2018. Contrastive Learning for Cold-Start Recommendation. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/1122445.1122456>.
20. Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2018. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2564–2572.
21. Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. 2019. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 100–109.
22. Asia J. Biega, Krishna P. Gummadi, and Gerhard Weikum. 2018. Equity of attention: Amortizing individual fairness in rankings. *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'18)*. Association for Computing Machinery, New York, NY, 405–414. DOI: <https://doi.org/10.1145/3209978.3210063>
23. Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS'12)*. Association for Computing Machinery, New York, NY, 214–226. DOI: <https://doi.org/10.1145/2090236.2090255>
24. Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. 2023. A Survey on the Fairness of Recommender Systems. *ACM Trans. Inf. Syst.* 41, 3, Article 52 (February 2023), 43 pages. <https://doi.org/10.1145/3547333>.
25. Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 452–461.
26. Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Tie-Yan Liu, Wei Chen. A Theoretical Analysis of NDCG Type Ranking Measures. DOI: <https://doi.org/10.48550/arXiv.1304.6480>.
27. Ziwei Zhu, Jingu Kim, Trung Nguyen, Aish Fenton, and James Caverlee. 2021. Fairness among New Items in Cold Start Recommender Systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*, July 11–15, 2021, Virtual Event, Canada. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3404835.3462948>.
28. Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H Chi, et al. 2019. Fairness in recommendation ranking through pairwise comparisons. In

- Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2212–2220.
29. Sahin Cem Geyik, Stuart Ambler, and Krishnaram Kenthapadi. 2019. Fairness-aware ranking in search & recommendation systems with application to linkedin talent search. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2221–2231.
 30. Toshihiro Kamishima and Shotaro Akaho. 2017. Considerations on Recommendation Independence for a Find-Good-Items Task. (2017).
 31. T Kamishima, S Akaho, H Asoh, and J Sakuma. 2013. Efficiency Improvement of Neutrality-Enhanced Recommendation.. In Decisions@RecSys.
 32. Toshihiro Kamishima, S Akaho, H Asoh, and J Sakuma. 2018. Recommendation Independence. In Conference on Fairness, Accountability and Transparency.
 33. T Kamishima, S Akaho, H Asoh, and I Sato. [n.d.]. Model-based approaches for independence-enhanced recommendation. In 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW).
 34. Flavien Prost, Hai Qian, Qiuwen Chen, Ed H Chi, Jilin Chen, and Alex Beutel. 2019. Toward a better trade-off between performance and fairness with kernel-based distribution matching. arXiv preprint arXiv:1910.11779 (2019).
 35. Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In Advances in neural information processing systems. 1257–1264.
 36. Mohammadmehdi Naghiaei, Hossein A. Rahmani, and Yashar Deldjoo. 2022. CPFair: Personalized Consumer and Producer Fairness Re-ranking for Recommender Systems. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), July 11–15, 2022, Madrid, Spain. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3477495.3531959>.
 37. Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).
 38. Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
 39. Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In Advances in neural information processing systems. 1257–1264.
 40. Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 452–461.
 41. Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of International Conference on World Wide Web*. 173–182.
 42. Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. *Proceedings of International ACM SIGIR Conference on Research and Development in Information Retrieval*, 639–648.
 43. Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the International ACM SIGIR conference on Research and Development in Information Retrieval*. 165–174.
 44. Yinwei Wei, Xiang Wang, Xiangnan He, Liqiang Nie, Yong Rui, and Tat-Seng Chua. 2021. Hierarchical User Intent Graph Network for Multimedia Recommendation. *IEEE Transactions on Multimedia* (2021).

-
45. Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. *Proceedings of International Conference on Machine Learning*.
 46. Dimitris Bertsimas, Vivek F Farias, and Nikolaos Trichakis. 2011. The price of fairness. *Operations research* 59, 1 (2011), 17–31.
 47. Wenwan, Fuwei, Lidong1, T-habao, Nanya, Mingzhou. 2020. MINILM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers.
 48. Chen Xu, Sirui Chen, Jun Xu, Weiran Shen, Xiao Zhang, and Gang Wang, Zhenghua Dong. 2023. P-MMF: Provider Max-min Fairness Re-ranking in Recommender System. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*, May 1–5, 2023, Austin, TX, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3543507.3583296>.
 49. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. 2016. Deep Residual Learning for Image Recognition. Submissions to ILSVRC & COCO 2015 competitions