2-15-2024

Mining User Facebook Post Likes for Cross Domain Product Recommendations across E- commerce platforms

Emmanuel Jojoe Ainoo
University of Windsor

Mining User Facebook Post Likes for Cross Domain Product Recommendations across E- commerce platforms

By

Emmanuel Ainoo

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfilment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2024

Mining User Facebook Post Likes for Cross Domain Product Recommendations across E-commerce platforms

by

Emmanuel Jojoe Ainoo

APPROVED BY:

January 15th, 2024

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Ecommerce recommendation accuracy can be improved by mining patterns from other domains such as social media (Facebook), to predict purchase behaviours. The "cross-site cold start problem" arises when traditional recommender systems, relying on e-commerce purchase history, face platforms with no user history. Existing systems such as the GaoLinRec23 (2023) system that employs (CMF) Collective Matrix Factorization to jointly factorize user-item interaction matrices from both domains, WangZhaoRec21 (2021), GaoRec20 (2021), WangHeNeiChuaRec17 (2017), which also incorporate user attribute and social connections from the social media domain have attempted to bridge the gap, however the assumption that specific item embeddings which are the specific product details are shared between these domains does not align with the real-world scenario. Major e-commerce and social media platforms, including Amazon and Facebook, typically do not exchange granular product information. This disconnect poses a critical obstacle for existing recommendation systems in providing accurate suggestions for users starting with no observable e-commerce activity.

This thesis proposes Facebook Data Cross Recommendation '2023 (FD-CDR '23) system, which uses the proposed MLTU (Mine Likes and Transactions per User) algorithm to extract likes and purchase history of users from both domains, transforming then into itemsets. A modified association rule mining is applied uncover patterns of frequent co-occurrence between user Facebook post likes and e-commerce transactions as rules. It then uses the proposed HARR (Hybrid Association Rule Recommendation) algorithm to match new user facebook likes to, generated rules such as "Users who typically like cooking posts, buy cooking recipes" without needing to share item embeddings across platforms and still solve the cross-site cold start problem. Experimental results with precision and recall show that the proposed FD-CDR'23 system provides more accurate recommendations than the mentioned existing systems.

Keywords: Cross-domain recommendations, social media, e-commerce, recommender systems, data mining, association rule mining, collaborative filtering, social media activity

Dedication

I would like to dedicate this thesis to my parents, Mr and Mrs Ainoo, my supervisor, Dr Christie Ezeife and internal and external readers; Dr Jeffrey Rau and Dr Shafaq Khan, and my friends who have helped and supported to complete my graduate study at the University of Windsor.

Acknowledgements

I extend my heartfelt gratitude to my parents, whose unwavering support and encouraging words fuelled my determination to complete this endeavour.

My profound appreciation goes to Dr. Christie Ezeife for her invaluable guidance and supervision. Her constructive criticism and advice were instrumental in driving the successful completion of this work. I am also thankful to Dr. C. I. Ezeife for continuous research assistantship positions from NSERC.

Special acknowledgment is extended to my external reader, Dr. Jeffrey Rau, my internal reader, Dr. Shafaq Khan, and the chair, Dr. Majid Afshar, for graciously agreeing to be part of my thesis committee. I highly appreciate your decision, despite your busy schedules, to read the thesis and provide valuable input to shape the contributions of this thesis into research.

Lastly, I want to express my sincere thanks to friends and colleagues at the University of Windsor for their advice and unwavering support throughout the duration of this work.

Table of Contents

List of Tables

List of Equations

List of Figures

List of Algorithms

List of Abbreviations

CDRS	Cross-Domain Recommendation Systems
RS	Recommender Systems
CF	Content Filtering
CBF	Collaborative Filtering
HF	Hybrid Filtering
SPM	Sequential Patterning Mining
GSP	Generalized Sequential Pattern
CMF	Collective Matrix Factorisation
AR	Association Rules
MF	Matrix Factorisation
MLTU	Mine Likes and Transactions per User
HARR	Hybrid Association Rule Recommendation
FD-CDR	Facebook Data Cross Domain Recommender
ID	Identity Document
API	Application Programming Interface
JSON	JavaScript Object Notation
RFM	Recency, Frequency, Monetary
MPS	Minimum Profit Support
SHOD	Sequential Historical Periodic Database
MFUA	Matrix factorization with User Attributes
LDA	Latent Dirichlet Allocation
CSR	Coordinate System Transfer
SVD	Singular Value Decomposition
CD-SPM	Cross-Domain Sequential Pattern Mining
PBMF	Preference Biased Matrix Factorization
SNSs	Social networking services
NCSR	Neural Social Collaborative Ranking
NCF	Neural collaborative filtering
SOCIALMF	Matrix Factorization with Social Regularization
SID	Sequence Identifier
NSCR	Neural Social Collaborative Ranking
UUID	Universally Unique Identifier.
TP	True Positives
FP	False Positives
TN	True Negatives
FN	False Negatives

CHAPTER I: INTRODUCTION

In recent years, the lines separating e-commerce and social media networking have become less distinct. E-commerce platforms like Amazon **(Waters, 2024)** now incorporate several features commonly associated with social media networks, including real-time status updates and interactions between buyers and seller's media **(Safko & Brake, 2009).** Additionally, some e-commerce websites offer social login functionality, enabling new users to sign in with their credentials from popular social media platforms like Facebook **(Hall, 2024),** Instagram **(Eldridge, 2024),** or Twitter **(Britannica, 2024)** using tools like "facebook connect" **(Zhang & Pennacchiotti, 2015).**

E-commerce systems encompass a distinct component known as recommender systems, aimed at predicting users' preferences and boosting sales **(Fuchs, 2014).** These components employ advanced techniques like collaborative filtering and content filtering to anticipate users' potential purchases by using user history **(Aggarwal, 2016).** Typically reliant on user history, these systems face a significant challenge when attempting predictions in the absence of any activity.

Addressing this challenge, known as the "**cross-site cold start**," involves the transfer of activity or historical data from an alternate domain, such as social media **(Zhao et al., 2016).** This inter-domain transfer becomes instrumental in overcoming the hurdle of predicting user preferences when there is no observed activity in the e-commerce domain. To tackle the cross-site cold start problem, recommender systems necessitate the utilization of cross-domain techniques, including transfer learning or joint matrix factorization **(Cantador et al., 2015).** These techniques enable the integration of user preferences from both e-commerce and social media domains, enhancing the accuracy of predictions regarding purchase behaviour.

While online cross-domain product recommendation has been extensively studied before **(Gao et al., 2023), (Gao et al., 2019), (Lin et al., 2013), (Zhao et al., 2016),** most of these studies focus on constructing solutions for specific e-commerce websites that have a database integration to allow shared user, products and transactional records or rely on transaction history confined to both domains. In most scenarios, the product data which could include product details such as product names, IDs, categories, and other schema related fields, are not shared across the two domains. The products themselves are not shared across the domains. As an illustration of this problem, consider the two scenarios of data here:

First Scenario (GaoLinRec23)

	Users	**Items**
Platform A (Beidan)	1	1
	2	2
	3	3
	4	4

Table 1.1 Sample Users and Items from Beidan

	Users	**Items**
Platform B (WeChat)	1	1
	2	2
	3	3
	4	4

Table 1.2 Sample Users and Items from WeChat

In the first scenario 1, we have items and users in platform A shown in **Table 1.1,** and in **Table 1.2**, we have items and users in platform B. If user1 likes item1 (showing interest in that item1) on WeChat in **Table 1.2** , we can simply recommend that same item1 on Beidan in **Table 1.1**.

Second Scenario (eBay-Facebook)

	Users	Products
Platform A (eBay)	1	1
	2	2
	3	3
	4	4

Table 1.3 Sample Users and Products from eBay

	Users	Posts
Platform B (Facebook)	1	"water"
	2	"I love cars"
	3	"Airpods yes"
	4	"Wow shoes"

Table 1.4 Sample Users and Posts from Facebook

In the second scenario, what do we recommend when user2 posts "I love cars" on Facebook in **Table 1.4**, product 2? or 3? We are not sure what to recommend here from **Table 1.3** because There is no straightforward way to extract what specific product interests from user2's posts. Amazon may have a product schema in their database while Facebook may have an entirely different structure for modelling users and their networks.

Item embeddings refer to the products and its schema details across each platform. It then becomes a challenging task to apply existing systems to domains that do not that share products, and it is even more challenging to extract and transform user's interests from the social media activity into features that could be used effectively for product recommendation.

In this thesis we aim to answer two research questions :

1. How do we recommend items to users with no historic activity, using historic activity from another domain?
2. How do we identify user item preferences across domains without platforms sharing any form of item data or schema structure?

To answer both questions, we propose utilizing "**facebook connect**" to connect users (Zhang & Pennacchiotti, 2013) who are common across both social media platforms and e-commerce platforms. These are individuals who have logged in into e-commerce platforms with their social media accounts as a bridge to link user's interests from social media activity to their interests on e-commerce platform. We then propose a procedure to creating user item interests from older users' social media platform liked posts and ecommerce transactions by counting how frequently combinations of liked posts and purchased products occur as patterns. We believe a pattern where multiple users who liked the same posts purchased the same product can be useful in predicting what item a user may be interested in based on a specific liked post. For instance, if a user frequently posts and likes content about cooking on Facebook, they may be interested in recipes or cooking utensils from Amazon.

What exactly we are adding on top of (Gao et al., 2023), and other existing systems is a way to represent user item interests from extracted social media activity logs for product recommendation on the e-commerce platform by discovering associations between purchased products and interacted social posts for platforms do not share items or any product information.

The proposed solution is essential for start-up e-commerce platforms that have very low patronage or user activity but allow users to connect social accounts. The value here is to boost sales within the e-commerce platform by discovering interests with users connected social accounts and recommending products based on their interests. In the context of this research implementation and experiments we would use historical purchased products as ecommerce data and Facebook likes as the social media activity used for the system

Let us understand what social media activity, which is derived from the social networking domains, encompasses, and how this form of data is used within the context of this research.

1.1 Social Media

In the digital age, social media platforms have emerged as pervasive tools for communication, interaction, and content sharing **(Boyd & Ellison, 2007).** These platforms, such as Facebook which creates a social network of users to converse and create media posts on their interests **(Hewitt & Forte, 2006),** Twitter which is an open public platform for users to share random thoughts, Instagram which allows users to upload photos of their everyday lifestyle and TikTok which similarly uses short videos to create exciting

content to be shared, have created an interconnected global network that significantly influences individual behaviour and societal dynamics **(Kaplan & Haenlein, 2010).**
The widespread adoption of social media has transformed how people connect, share information, and express themselves and their preferences openly. However, let us understand why we would prefer the social media domain to other alternatives like movies or e-sports.

According to a study by Smith and Anderson, nearly 72% of adults use social media, showcasing the widespread adoption of these platforms **(Smith & Anderson, 2018)**. We currently live in an age where almost everyone uses social media platforms as means to express opinions, engage in activities, network with new people, and make friendship connections, and social media adoption has increased vastly in recent years.

1. With a global user base of 4.33 billion, social media accounts for approximately 55% of the world's population as of January 2021 **(DataReportal, 2021, Simon Kemp)**
2. Individuals spend an average of 2 hours and 29 minutes per day on social media platforms, highlighting its pervasive presence **(DataReportal, 2021, Simon Kemp)**

The massive number of users and average hours spent on these social network platforms provide a rich history of activity to extract user interests from. Users are more likely to spend some more time typically just browsing on a social media platform than an ecommerce platform, providing rich history to discover preferences from.

Let us understand what historic data we can distract to discover some insights into user preferences and behaviour.

1.1.1 Social Media Activity

Social media activity refers to the diverse range of interactions and actions that users engage in on social media platforms **(Hoffman & Fodor, 2010).** These activities encompass various forms of engagement, such as creating posts which are published text short messages **(Kaplan and Haenlein, 2010),** liking posts, commenting on posts, engaging in conversations with other users, searching for specific content, and measuring the amount of time spent on the platform **(Dwivedi et al., 2023).**

Engagement Form	Example	Data Type
Posts	"I love cars"	Text
Likes	Like or Dislike (True or False)	Boolean
Comments	"Beautiful car, where did you get this"?	Text
Search History	"Best recipes for homemade pizza"	Text
Time Spent on Platform	45	Numeric (mins)

Table 1.5 Various Forms of Engagement as Social Media Activity

In **Table 1.5,** we can observe the different forms of data we can derive as social media activity from users' engagement on social media platforms. Typically, "likes" simply implies a user's interest in the specific post by tapping a like button attached to a post **(Evans et al., 2017).** For example, let us see a summary Melanie's page likes from Facebook in **Figure 1.1**.

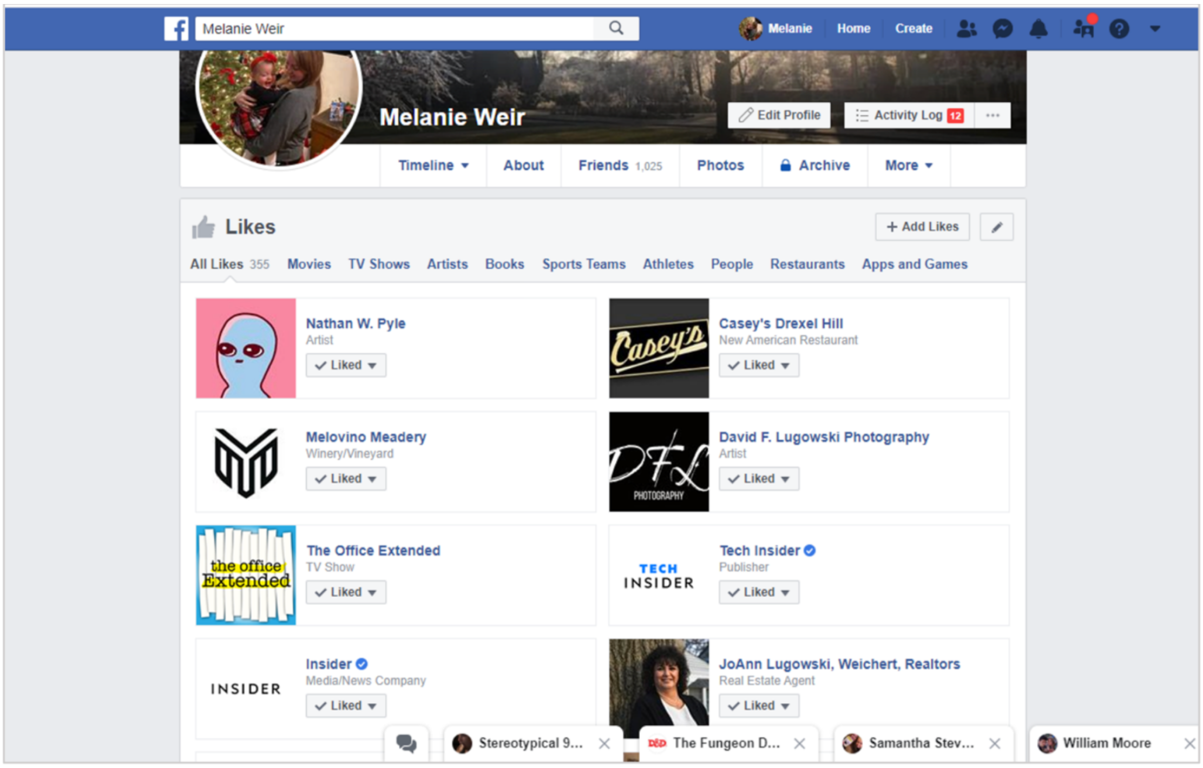


Figure 1.1 Sample User (Melanie) Facebook Likes

From **Figure 1.1**, we can derive that Melanie is interested in Nathan Pyle's Art **(Nield, 2019).** We believe such information on her activity log can provide rich insights for what she may want to buy and her preferences. The activities create a trail of digital data that can be analysed to uncover patterns and associations. For example, a user extremely interested in cooking, may like posts on cooking utensils, comment on nicely packaged food and post cooking recipes and lot more activity **(Fuchs, 2014)** . Likewise, that same user may mostly likely be interested in purchasing cooking recipes, cooking utensils and we can only justify this by analysing their social media activity and e-commerce activity to discover a correlation.

In the context of this research, we are interested in the data of Facebook activity likes that show user actions of liking a specific post extracted from Facebooks Graph API **(Facebook, 2024)** as their Social Media Activity. The Facebook Graph API is a programming interface provided by Facebook that allows developers to interact with and retrieve data from the Facebook platform. It exposes a structured representation of the Facebook social graph, including objects such as users, pages, photos, events, as a way for developers to access and manipulate data within the Facebook ecosystem **(Weaver & Tarjan, 2013).** It outputs data in the form of JSON (JavaScript Object Notation) object which is a collection of key-value pairs where each key is a string, and the values can be strings, numbers, arrays, objects, booleans, or null. In the context of the Facebook Graph API, when you make a request to retrieve data, the response is often in JSON format **(Schafer, 2023).** For example, if you request information about a user activity on a specific post, the response is shown in **Figure 1.2**

```
{
    "id": "123456789",
     "message": "Enjoying a great meal with @Friend1 and @Friend2!",
     "created_time": "2023-01-01T12:34:56+0000",
     "likes": {
         "data": [
             {
                "id": "1041851b-87db-4054-b56a-d1e4c280f39b",
                 "name": "User 1"
             },
             {
                "id": "0844682d-6e20-4f92-86e9-a0c60ee81eb2",
                "name": "User 2"
             }
         ]
     },
}
```

Figure 1.2 Sample Activity Log Data from Facebook Graph API

Data from **Figure 1.2** contains user actions on post including likes of users which is an example of the social media activity we are interested in for this research. It is crucial to note that user actions on social media can be unpredictable and may not exhibit a distinct pattern **(Waheed et al., 2017).** For instance, users might unintentionally like a post or engage with content randomly without any specific significance, making it challenging to interpret such actions as preferences. Nevertheless, consistently liking similar posts or content within the same category can be construed as a preference for that specific category.

Now that we have a grasp of what social media and social media activity entail, let us understand how we intend to mine such data.

1.2 Data Mining

Data mining is a methodology designed to derive valuable insights from extensive datasets aligned with specific business objectives **(Han et al., 2022).** Given the notion of being "rich in data but lacking in information," data mining has garnered significant attention due to its pivotal role in transforming large datasets into meaningful information and knowledge **(Tan, Steinbach & Kumar, 2006).**

The conventional data mining methodologies that aid in predictive analysis include Association Rule Mining which identifies patterns where the occurrence of one set of items implies the occurrence of another set of items **(Agrawal & Srikant, 1994),** Classification which involves the process of categorizing data into predefined classes or groups based on identified patterns and features **(Jurafsky & Martin, 2014).** Lastly, Clustering involves grouping similar observations points into segments based on their inherent similarities **(Dunham, 2006).** For the proposed solution, we would dive into an example of how Association Rule Mining works.

1.2.1 Association Rule Mining

Association rule mining is about finding interesting connections between items in a dataset. Imagine you have a list of items, and you have a bunch of transactions where each transaction is a collection of items **(Agrawal & Srikant, 1994)**. For example, you might have a list of things people buy, and each purchase is a transaction.

Now, association rule mining looks for patterns like "if someone buys item A, they are likely to buy item B." These patterns are called association rules. An association rule looks like "A ⇒ B," where A and B are sets of items. It suggests that if someone has item A in their transaction, they likely have item B as well. The "support" of a rule tells us how often that rule appears in the dataset. For example, if 30% of all transactions have both items A and B, the support for the rule "A ⇒ B" is 30%. So, association rule mining helps us discover connections between items in our data, showing us which items tend to be bought together. Association rule mining looks for interesting relationships between objects in each data set.

Let $I = \{i_1, i_2, \ldots, i_m\}$ represent the universal set containing all possible items in the dataset. Let D be a set of database transactions where each transaction T is a subset of I such that $T \subseteq I$. Each transaction is associated with an identifier, called TID. Now let A be a specific subset of I representing a set of items. A transaction T is said to contain A if and only if $A \subseteq T$. In simpler terms, if all the items in set A are present in a transaction T, then T contains A. An association rule is an implication of the form $A \Rightarrow B$ where A and B are both subsets I specifically $A \subset I$, $B \subset I$. Additionally, $A \cap B = \emptyset$ indicating that the sets A and B are disjoint meaning they have no items in common The rule $A \Rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D, that contain the union of A and B, $A \cup B$.

$$Support\ (A \Rightarrow B) = \frac{Number\ of\ Transactions\ containing\ I}{Number\ of\ Transactions}$$

Equation 1.1 Support for Association Rule Mining

The rule $A \Rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A which includes B. That is,

$$Confidence\ (A \Rightarrow B) = \frac{Support\ (A\ \cup B)}{Support\ (A)}$$

Equation 1.2 Confidence for Association Rule Mining

Rules are considered strong that meet both a minimum support threshold (min-sup) and a minimum confidence threshold (min-conf) **(Han et al., 2022)**

Apriori algorithm is a classic algorithm that helps mine frequent itemset and relevant association rules. It has got this odd name because it uses 'prior' knowledge of frequent itemset properties **(Agarwal & Srikanth, 1994)**.

To understand Apriori better, let us look at an example. In this example we would apply it on set of items of user liked posts and purchase products The itemset collection would be I = {postA, productB, postC, productD, postE}

The problem to be solved here is to identify which list of items co-occur most frequently together

UserID	List of Items
U1	postA, productB, postC
U2	postB, postC, productD
U3	productD, postE
U4	postA, productB, postC
U5	postA, productB, postC, postE
U6	postA, productB, postC, productD

Table 1.6 Association Rule Mining Sample Data

The Apriori makes the following assumptions:

1. All subsets of the frequent itemset should be frequent.
2. Similarly, the subsets of an infrequent itemset should be infrequent.
3. Set a threshold support level. In our case min_support=2
4. Set minimum confidence to be 50%.

Now let us follow the steps from Apriori to mine frequent patterns from the data in **Table 1.6**

Step 1: Generate Candidate 1-itemsets.

Count the occurrence of each item in the transactions and generate candidate 1-itemsets which are all possible combinations of single items:

Item counts = {postA: 4, productB: 5, postC: 6, productD: 3, postE :2 }

Frequent 1-itemsets: {postA}, {productB}, {postC}, {productD}, {postE}

Step 3: Generate Candidate k-itemsets (k=2) and Prune Infrequent Itemsets

Generate candidate 2-itemsets which are combinations of items with k elements. In this step, we focus on 2-itemsets. and prune those that are infrequent by Removing candidate itemsets that do not meet the minimum support threshold=2.

Candidate 2-itemsets: {postA, productB}, {postA, postC}, {postA, productD}, {postA, postE}, {productB, postC}, {productB, productD}, {productB, postE}, {postC, productD}, {postC, postE}, {productD, postE}

Count the occurrence of each candidate 2-itemset: {postA, productB} :4, {postA, postC} :4, {postA, productD}: 2, {postA, postE} : 2, {productB, postC} : 5, {productB, productD}: 3, {productB, postE} :2, {postC, productD} :3, {postC, postE}: 2, {productD, postE}: 2.

Frequent 2-itemsets after pruning that do not meet minimum support of 2.

{postA, productB}, {postA, postC}, {productB, postC}, {productB, productD}, {postC, productD}

Step 4: Repeat the Process for k=3 and k=4

Repeat the process for k=3 and k=4, generating candidate k-itemsets, counting their occurrences, and pruning infrequent itemsets. In this case, we won't find any frequent 3 or 4-itemsets.

Step 5: Generate Association Rules

Now, use the frequent itemsets to generate association rules based on a specified confidence threshold. The association rules have two parts: the antecedent (if) and the consequent (then). For example, {postA, productB} => {postC}. Prune rules that do not meet the minimum confidence threshold. For each frequent k-itemset, consider all possible combinations of antecedent and consequent.

Association Rules:

{postA, productB} => {postC} (Support: 4, Confidence: 100%)

{postA, postC} => {productB} (Support: 4, Confidence: 100%)

{productB, postC} => {postA} (Support: 4, Confidence: 80%)

{postA} => {productB, postC} (Support: 4, Confidence: 100%)

{productB} => {postA, postC} (Support: 4, Confidence: 80%)

{postC} => {postA, productB} (Support: 4, Confidence: 66.67%)

{productB} => {productD} (Support: 3, Confidence: 60%)

{productD} => {productB} (Support: 3, Confidence: 100%)

{postC} => {productD} (Support: 3, Confidence: 50%)

{productD} => {postC} (Support: 3, Confidence: 100%)

From these generated rules we have an example as {postA} => {productB, postC} with a confidence of 100% indicating that 100% of the time liking postA is followed by purchase of productB and like of postC

1.3 Ecommerce Recommendation Systems (RS)

Ecommerce systems use digital technologies to facilitate online transactions, enabling businesses to reach a global customer base and consumers to conveniently purchase products and services from the comfort of their homes **(Watson et al., 2008).** The popular systems we all mostly use every day include Amazon, eBay, Walmart, and many others **(Laudon & Traver, 2017).** These systems contain an important component called recommender systems, necessary for increasing user engagement and predicting sales, recommendation by suggesting items that users may like to buy **(Gao et al., 2019).**

Recommender systems simply make suggestions of items for a particular user **(Aggarwal, 2016).** They typically rely on two types of data: user-item interactions and attribute information about users and items. There are three popular techniques for recommendation: collaborative filtering recommends items to a user based on the preferences and behaviours of similar users. The underlying idea is that users who have agreed in the past tend to agree in the future. It can be further divided into user-based and item-based collaborative filtering content-based filtering, and hybrid filtering **(Koren et al., 2009).** Secondly there is content-based filtering recommends items to a user based on the characteristics of the items and a profile of the user's preferences. It involves analysing the content or attributes of items and matching them to a user's preferences **(Pazzani et al., 2007).** Lastly, Hybrid filtering combines collaborative and content-based filtering to overcome the limitations of each approach. By integrating both techniques, hybrid systems aim to provide more accurate and diverse recommendations, taking advantage of the strengths of both collaborative and content-based methods **(Burke, 2002)**

1.3.1 Techniques for Recommender Systems

The three (3) popular RS collaborative filtering, content-based systems, and hybrid have been briefly described in section above. We would look at an example of collaborative filtering to generally understand how recommendation works.

1. **Collaborative filtering (CF)** is a widely used approach in recommender systems that uses user-item interaction data to make recommendations **(Aggarwal, 2016).** The underlying principle is that users who have similar preferences or behaviours in the past are likely to have similar preferences in the future. For example, If User A and User B have similar preferences and both liked Movie X, Collaborative Filtering would recommend Movie X to User B if User A has already watched and liked it. The formula to calculate the predicted rating r_ui is given as :

$$r\_ui = \mu + \frac{\sum(sim(u,\ v)\ *\ (r\_vi\ -\ \mu))}{\sum(|sim(u,v)|)}$$

Equation 1.3 Predicted Rating for Collaborative Filtering

From **Equation 1.3**, r_ui represents the predicted rating of user u for item i, μ represents the overall average rating, $sim(u,v)$ represents the similarity between users u and v, and r_vi represents the rating of user v for item i **(Aggarwal, 2016).** For similarity we can any similarity function such as cosine similarity **(Salton & McGill, 1983)** or Pearson correlation **(Resnick et al., 1994)**

Let us see an example of applying collaborative filtering steps on a sample movie recommendation dataset Table to make recommendation in

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5
User 1	5	3	4	4	-
User 2	4	-	5	5	2
User 3	-	1	2	4	5

Table 1.7 Sample Movie Dataset for Collaborative Filtering Example

From the sample dataset in **Table 1.2**, the goal is to predict for User 1 and Movie 5 using collaborative filtering. Let us follow the algorithm step by step:

Step 1: Calculate Similarity

We need to compute the similarity between User 1 and other users (User 2 and User 3). Let us use the cosine similarity measure **(Salton & McGill, 1983) :**

Similarity(User 1, User 2):

sim(1, 2) = (5-3.6)*(4-3.2) + (3-3.6)*(-) + (4-3.6)*(5-3.2) + (4-3.6)*(5-3.2) / sqrt((5-3.6)^2 + (3-3.6)^2 + (4-3.6)^2 + (4-3.6)^2) * sqrt((4-3.2)^2 + (-)^2 + (5-3.2)^2 + (5-3.2)^2)

sim(1, 2) ≈ 0.49

Similarity(User 1, User 3)

sim(1, 3) = (5-3.6)*(-) + (3-3.6)*(1-2.8) + (4-3.6)*(2-2.8) + (4-3.6)*(4-2.8) + (-) / sqrt((5-3.6)^2 + (3-3.6)^2 + (4-3.6)^2 + (4-3.6)^2) * sqrt((-)^2 + (1-2.8)^2 + (2-2.8)^2 + (4-2.8)^2 + (5-2.8)^2)

sim(1, 3) ≈ -0.88

Similarity(User 2, User 3)

sim(2, 3) = (4-4.2)*(-) + (-) + (5-4.2)*(2-2.8) + (5-4.2)*(4-2.8) + (2-2.8) + (-) / sqrt((4-4.2)^2 + (-)^2 + (5-4.2)^2 + (5-4.2)^2 + (-)^2) * sqrt((-)^2 + (1-2.8)^2 + (2-2.8)^2 + (4-2.8)^2 + (5-2.8)^2)

sim(2, 3) ≈ -0.63

Step 2: Select Neighbours

We select k=2, which is the top 2 users that are like user 1 based on the similarity calculations step 1.

Neighbours for User 1: {User 2, User 3}

Step 3: Predicted Ratings

Predict the missing rating for Movie 5 for User 1 using the ratings of the selected neighbours using Equation 1.3

For User 2: r_{u1} += sim(1, 2) * (2-3.2)

For User 3: r_{u1} += sim(1, 3) * (5-2.8)

r_{u1} ≈ 3.9

So, the predicted rating for Movie 5 for User 1 is approximately 3.9.

2. **Content-based filtering (CBF)** relies on the attributes or characteristics of items to make recommendations. It analyses the content of items and compares it to the user's preferences or profile to identify items that are likely to be of interest. The system creates a profile for each user based on their preferences and a profile for each item based on its attributes **(Aggarwal, 2016)**. In a movie recommendation system, if a user has previously liked action movies, Content-Based Filtering might recommend new action movies to that user based on shared characteristics such as genre, actors, or director. The formula to calculate the predicted rating is given as :

$$Predicted\ Rating\ (u,\ i) = \sum_{f} (w_{uf} \cdot f_{if})$$

Equation 1.4 Predicted Rating for Content Based Filtering

Where w_{uf} denotes the weight associated with the user u for feature f. It signifies the importance or strength of the relationship between the user's preferences and the specific feature. f_{if} represents the value of feature f for item i. It indicates the degree to which item I possesses the feature f.

3. **Hybrid recommender (HF)** systems combine collaborative filtering and content-based filtering methods to provide more accurate and diverse recommendations. These systems aim to overcome the limitations of individual approaches by leveraging their complementary strengths **(Tarus et al., 2018).** There are various ways to integrate collaborative filtering and content-based filtering, including weighted hybrid, switching hybrid, and mixed hybrid approaches.
   For example, in In a weighted hybrid recommender system, the predictions from collaborative filtering and content-based filtering are combined using weights. In the first step we calculate the Collaborative Filtering Prediction (CF) to capture collaborative patterns among users, then we calculate the Content-Based Filtering Prediction (CBF) to capture the content characteristics of items. Now we do a combination of CF and CBF predictions for a Hybrid Prediction by adding a weight factor α ($0 \leq \alpha \leq 1$) determining the influence of CF. Adjusting α allows customization of the hybrid model's emphasis on CF or CBF.

.

Why the need to cross to another domain for recommendation? --- The need to venture into cross-domain recommendation arises from the constraints posed by data scarcity, user inactivity, and the formidable cold-start issue, which is particularly pronounced for newcomers or users with limited engagement within a particular domain. Existing systems have tackled the mentioned problems with very complex solutions with the aim of improving recommendation accuracy, however these problems become practically easier to solve when you incorporate data from another domain.

Extensive research, such as that presented by Gao and his team **(Gao et al., 2023).**, underscores the significance of this approach. By harnessing insights from diverse domains like social media, this paradigm not only enriches personalization but also taps into the wealth of user profiles, ultimately leading to more precise and varied recommendations. This strategic shift not only enhances user experiences but also confers a competitive edge in the realm of recommendation systems, as elucidated in works like Zang and his colleagues **(Zang et al, 2022)** paper on "A survey of Cross Domain Recommendations".

Let us understand how cross domain recommender systems make recommendation from multiple domains.

1.4 Cross Domain Recommendation Systems (CDRS)

Cross-Domain Recommender Systems (CDRS) emerge as a compelling solution to bridge the gap between multiple domains, enabling the extraction of valuable insights from diverse user activities. These systems navigate the intricate web of user preferences across different domains, such as social media and e-commerce, to provide comprehensive and personalized recommendations. In most situations, it becomes necessary to group or merge domains within CDRS to improve recommendation accuracy and address challenges such as data sparsity **(Anwar & Uma, 2020).** Grouping domains involves combining related categories that share common characteristics or target similar user preferences.

Merging domains, on the other hand, involves combining different domains to utilize shared knowledge or user preferences across multiple categories. This grouping or merging process enables CDRS to enhance recommendations and provide a more holistic understanding of users' preferences and interests **(Zang et al, 2022).** For example, consider a scenario where a user frequently purchases women's fashion items from the fashion domain and shows a strong interest in beauty and skincare products from the beauty domain. By merging these domains, the recommendation system can use user's preferences and behaviours across both categories to provide more accurate and comprehensive recommendations. This approach helps overcome data sparsity issues and ensures that users receive recommendations that align with their evolving interests and preferences.

1.4.1 Definition of Domains

In the context of CDRS, the notion of "domains" plays a crucial role. A domain represents a distinct category of products or content within the e-commerce ecosystem. Each domain focuses on a specific set of items or services, such as electronics, fashion, books, or home appliances **(Zang et al, 2022).** CDRS uses information from multiple domains to provide personalized recommendations to users, considering their preferences and interests across different categories on different levels such as system level or item level **(Cantador et al., 2015)**.

Cross-domain recommendations inherently involve multiple domains, each characterized by distinct user behaviours and preferences. In our context, the domains encompass social media and e-commerce. There is usually the **target domain**, or which we want to make recommendations and the **source domain**, from which we draw data or insights to inform recommendations in the target domain **(Cantador et al., 2015).** Let us look at **Figure 1.3** to see what domains are discussed in the context of this research.

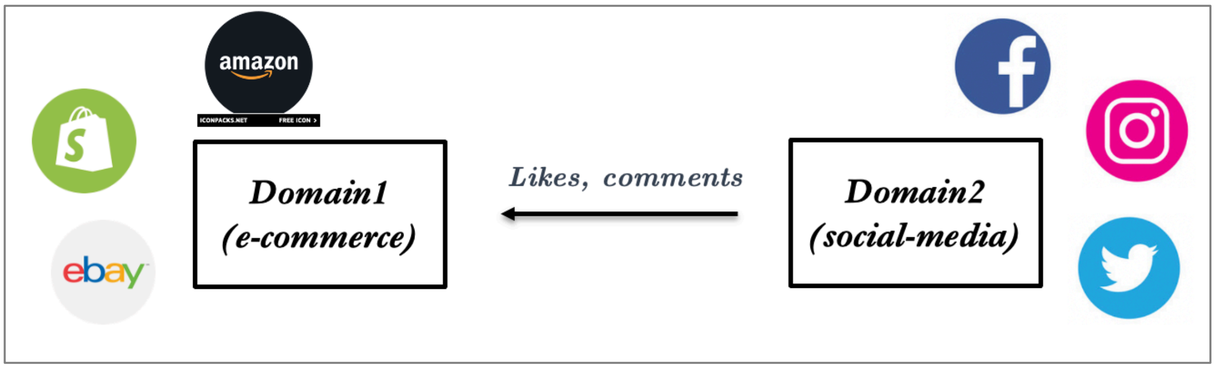


Figure 1.3 Sample Domain Examples

In **Figure 1.3** our target domain here as the arrow points to, is the e-commerce domains such as eBay and Amazon, while the goal is to extract likes or comments from the source domain, social media such as Facebook and Instagram. The intersection of these domains as shown in **Figure 1.3** forms the foundation of cross- domain recommendations. By connecting the dots between a user's likes on social media and their purchasing behaviour on e-commerce platforms, CDRS unearths patterns that might remain hidden within individual domains.

In this thesis we focus only on purchases and likes. We focus on purchases from the ecommerce domain and likes from the social media domain as shown in **Table 1.8**

Domain	Engagement Form	Datatype
Social Media	Likes	List of Liked Post Objects
Ecommerce	Purchases	List of Purchased Item Objects

Table 1.8 Domains used for this Thesis.

1.4.2 Cross Domain Techniques

A spectrum of techniques empowers CDRS to make meaningful recommendations. These methods can be categorized into knowledge-based, collaborative filtering, content-based, and hybrid approaches. Knowledge-based methods draw on explicit rules or domain knowledge to drive recommendations. The typical flow of cross domain is to create a user preference aggregation from combining data from both domains and sending it into recommendation system for recommendations on the target platform as shown in **Figure 1.4**

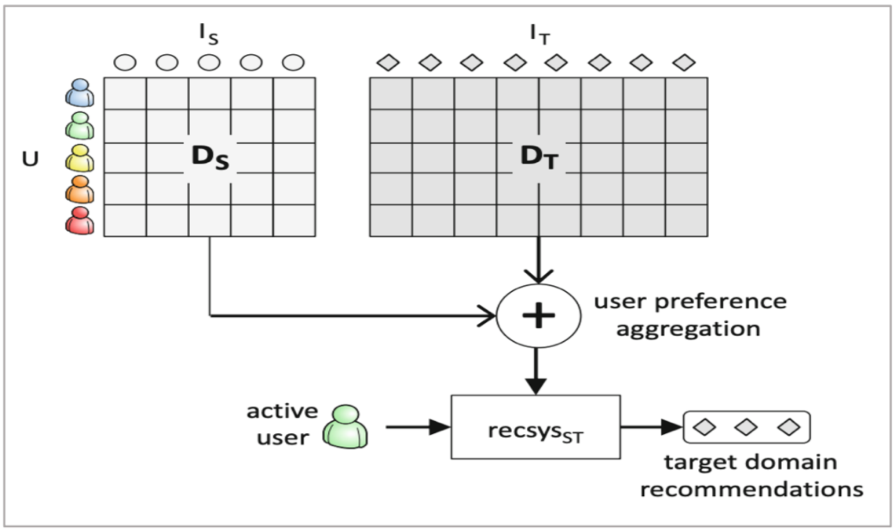


Figure 1.4 Sample CDRS System Architecture

As shown in **Figure 1.4** from Cantador's extensive book on cross-domain recommendation **(Cantador et al., 2015),** user's behaviours and preferences are aggregated from both domains (Ds and Dt) to provide personalized recommendations for the target domain. Data from both domains based on the user profile are fed into the CDRS to make recommendations. By utilizing knowledge representation techniques, domain characteristics can be hierarchically represented, facilitating concept categorization, and enabling more accurate mapping. As a result, recommendations become more effective and accurate **(Anwar & Uma, 2020).**

In the realm of e-commerce, cross-domain recommendation systems have gained significant attention due to their ability to provide personalized product recommendations across different domains or categories **(Bahaweres & Almujaddidi, 2022).** There are several different algorithms that can be used to build cross-domain e-commerce recommendation systems. The two general approaches are Transfer Learning which focus on transferring knowledge from a source domain to a target domain **(Pan et al., 2010)** and Joint Matrix Factorisation Simultaneously which factorizes matrices from multiple domains to capture shared latent features, **(Ma et al., 2011).**

1. **Transfer Learning** involves transferring knowledge gained from one domain to enhance the recommendation performance in another domain with limited data. for example, if a recommender system excels in movie recommendations but lacks data for books, transfer learning enables the model to transfer knowledge from the movie domain to enhance book recommendations **(Pan et al., 2010).** To attain the features of the target domain f_{target} from source domain is given as:

$$f_{target} = Transfer(f_{source}, D_{source}\, f_{target})$$

Equation 1.5 Transfer Learning in CDRS

Here, the goal is to enhance the performance of a target function f_{target} by incorporating knowledge or features from a related source domain D_{source} through the transfer learning process.

2. **Joint Matrix Factorisation** By jointly factorizing matrices from different domains, shared latent factors are identified, enabling the model to make recommendations that generalize across domains, even when data is sparse in individual domains **(Gao et al., 2019)**. Kuang and his colleagues (2021), discusses how matrix factorization technique can be used to factorize a user-item matrix into two lower-dimensional matrices representing user and item latent features, respectively. The latent features can then be used to predict the rating that a user would give to an item **(Kuang et al., 2021).**

In addition to the above approaches, there are several other algorithms that can be used to build cross-domain e-commerce recommendation systems. The choice of algorithm will depend on the specific dataset and the desired accuracy of the recommendations.

As an illustration, let us use the sample dataset from two domains: books and movies showing **Table 1.9** to make cross domain recommendations.

User	Harry Potter Book Rating	Harry Potter Movie Rating
User1	5	3
User2	4	1
User3	3	2
User4	-	4

Table 1.9 Sample Movie and Book Dataset for CDRS Example

The data in **Table 1.9** shows User's rating of Harry Potter across two domains Books and Movies. The goal here is to leverage on the ratings of similar users' ratings for harry potter movie to make recommendations for missing user 4's rating for harry potter book. This is shown in the steps below:

Step 1: Calculate the cosine similarity between User4 and all other users in the dataset.

Sim(User4, User1) = (4 x 5) / sqrt ((4^2) x (5^2)) = 0.96

Sim(User4, User2) = (4 x 4) / sqrt ((4^2) x (5^2)) = 0.8

Sim(User4, User3) = (4 x 3) / sqrt ((4^2) x (5^2)) = 0.72

Step 2: Select Nearest Neighbour, User with highest Similarity with User4

Let us assume we want to consider the top 2 most similar users (k = 2). So, the top 2 similar users to User4 are, User1 and User2

Step 3: Calculate Predicted Rating

Calculate predicated rating for missing User4 rating taking weighted average of top k users' book ratings.

Predicted rating for User4 (Book X) = ((0.96 x 5) + (0.8 x 4)) / 0.96 + 0.8 = 4

Existing systems **(Gao et al., 2023), (Gao et al., 2019) (Zhao et al., 2016),** utilize this approach differently across their proposed solutions as they have both domains share the necessary data through an integration making it easy to for instance, develop item representations for recommendation. It is easier to solve the problems CDRS try to address when the same item purchased on the ecommerce platform is also the same item liked on social media **platform. But then what happens when the two domains do not have any integration to share product details, how then do we discover their interests in products from their social media activity?** – This is the gap this thesis aims to fill.

1.4.3 Limitations of applying existing CDRS to platforms who do not share item embeddings.

1. **Inability to Handle Item Cold Start:**

   Existing systems heavily relies on shared product details between platforms to make recommendations. In cases where new items are introduced to one platform without shared embeddings, the system cannot effectively provide recommendations for these "cold" items, leading to a poor user experience.

2. **Limited Support for User Cold Start:**

   When a new user joins a platform where (Gao et al., 2023) operates and there is no historical data for that user on both platforms, the system struggles to understand their preferences and provide relevant recommendations. This results in a suboptimal recommendation quality for new users.

3. **Dependency on Cross-Platform Product Sharing:**
   GaoLinRecSys assumes the availability of shared item embeddings, making it dependent on the willingness and capability of platforms to share such data. In real-world scenarios, many platforms may not be willing to share sensitive item details, rendering the system ineffective for those cases.
4. **Lack of Scalability for Diverse Product Catalogs:**
   GaoLinRecSys may not scale well for e-commerce platforms with diverse and extensive product catalogs. Without shared item embeddings, it becomes challenging to efficiently capture user preferences for a wide range of products, potentially resulting in suboptimal recommendations.

By considering the temporal dynamics of user interactions from integrating social media insights, we aim to address the mentioned limitations above by not requiring any item embeddings to be shared across platforms. We propose a unique way to extract user item preferences by transforming their social media activity to make recommendations for users who did not make any transactions yet. This is done by deriving user interests from their social media activity and using those interests to make product recommendations especially for cold users (new users on the e-commerce platform) using our proposed algorithm FD-CDRS'23.

For example, if we establish a rule that users who like post A on Facebook, typically purchase item B on e-commerce platform. Once a new user joins the e-commerce platform with a Facebook account as soon as they like post A, we can recommend item B for purchase with no user purchase history yet at all and with not having to connect the two domains through a strenuous integration as they belong to separate companies practically.

1.5 Problem Formulation

Given historic e-commerce purchases, facebook likes and a set of users, the goal is to discover all prevalent associations between the new user like and purchases, sorting them based on confidence, and ultimately recommending items by prioritizing those associated with the highest-confidence rules.

Let us define some terms:

L be the set of user likes (from Facebook activity log).

P be the set of e-commerce purchases.

U be the set of users.

R be the set of association rules, where each rule is of the form $L_i \rightarrow P_j$ indicating a combination of likes leading to a purchase

Also $Conf(L_i \rightarrow P_j)$ be the confidence level for the association rule $L_i \rightarrow P_j$

Now, the formula for the confidence level $Conf$ for an association rule $L_i \rightarrow P_j$ can be calculated as

$$Conf(L_i \rightarrow P_j) = \frac{Support(L_i \cup P_j)}{Support(L_i)}$$

Equation 1.6 Confidence for FD-CDR'23

Where:

$Support(L_i \cup P_j)$ is the number of occurrences of the combination of likes L_i and purchases P_j in the dataset.

$Support(L_i)$ the number of occurrences of the combination of likes L_i in the dataset.

To generate personalized recommendations for a new user, we would follow these steps:

1. For the new user's like L_{new}, find all association rules $L_i \rightarrow P_j$ that contain L_{new}
2. Calculate the confidence level $Conf$ for each of these rules.
3. Rank the rules by confidence level in descending order.
4. Return the purchased items P_j from the top-ranked rule as the recommended items for the new user.

The formula to generate the top personalized recommendation for a new user is given as :

$$R_{new} = Top_{conf}\ (\{P_j\}\ for\ L_i \rightarrow P_j\ where\ L_{new} \in L_i)$$

Equation 1.7 Predicted Rating for New User for FD-CDR'23

R_{new} is the top rule based on confidence level for the new user's like and extracting the associated purchased items.

$Top_{conf}\ (\{P_j\}\ for\ L_i \rightarrow P_j)$ selects the top-ranked items based on confidence from the association rules that involve the new user's like L_{new}

$\{P_j\}$ This denotes the set of purchased items P_j

$L_i \rightarrow P_j$ represents association rules where L_i includes new user's like L_{new}

1.6 Thesis Contributions

The main limitation of existing systems such as **(Gao et al., 2023)**, **(Gao et al., 2019)** is that they attempted to solve the cross-site cold start problem on the ecommerce domain by using the same shared item embeddings across the domains. Thus, in this thesis we propose these contributions:

1. Mining the frequent combinations of user purchases and likes to extract user item preferences for recommendation without sharing item embeddings across domains.
2. We provide extensive analysis of user purchasing patterns and Facebook activity log over 13,000 users and show a subset of Facebook features correlates with purchase behaviours.
3. Improve recommendation accuracy by predicting purchase behaviours of users from their Facebook like activity.
4. Address the cold start problem by building the proposed FD-CDRS'23 recommender system for ecommerce start-ups with low user patronage and no activity.

1.6.1 Thesis Feature Contributions

1. **Using user facebook likes activity to predict purchase behaviours of users.** Users who spend a lot of time on Facebook provide a rich pool of activity log that could be used to represent their preferences. In this thesis, we extract intricate patterns from user Facebook likes over a substantial period and transform them into meaningful predictors of potential ecommerce.
2. **Discovering associations between user Facebook likes and their corresponding ecommerce purchases to represent user preference of item embeddings.** In situations where products are not shared across platforms, it becomes challenging to extract and transform user's interests from the social media activity into features that could be used effectively for product recommendation.

   In this thesis, we understand the associations between user interests on both platforms, by mining the frequents combinations of likes and purchases with high confidence. We believe that it is a pattern when multiple users who liked a specific post A, always buy a specific product B. Once that pattern is discovered, new users who like that specific post A on Facebook can be recommended product B.
3. **Improve recommendation accuracy in addressing cross-site cold start problem**. We address the problem of making recommendations to users who have no history at all on the ecommerce platform by using their facebook activity to make product recommendations more accurately, and experimental results discussed in **Chapter 4** show that our system FD-CDRS'23 preforms better than related existing systems.

1.6.2 Thesis Procedural Contributions

To make the listed feature contributions, this thesis proposes the **FD-CDR system** (Algorithm) which follows these main steps:

a. Apply the MLTU (Mine Likes and Transactions per User) algorithm present in **(Algorithm 3.2)** which.

    a. Extracts and clean (remove missing values, unused data rows and inconsistencies) ecommerce purchases from ecommerce domain.
    b. Extracts and clean (remove missing values, unused data rows and inconsistencies) user likes on posts from facebook activity logs of users from Facebook Graph API
    c. Combine ecommerce purchases and facebook likes by a chosen unique user identifier.
    d. Convert combined data into itemsets per user by creating a list of each user and their combined activity.

b. Apply Modified Apriori algorithm present in **(Algorithm 3.3)** to discover frequency of occurrences of Likes leading to a purchase $L_i \rightarrow P_j$ shown in **Section 1.4**
    a. This first considers the datatype of each item; Posts or Products to generate rules prioritizing rules where posts imply products to cater for limited activity on ecommerce platform.
    b. It discovers rules by counting the frequent combinations of itemsets {posts, products} retrieved from step 1d, which indicates a set of liked posts and purchased products using a minimum support.

c. Now given a new a user, Apply HARR (Hybrid Association Rule Recommendation) Algorithm present in **(Algorithm 3.4)** which:
    a. For a new user like, find all the association rules that contain that like
    b. Retrieve the confidence level of each of these rules.
    c. Rank the rules by confidence level in descending order.
    d. Return the purchased items from the top-ranked rule as the recommended items to the new user.

d. Repeat step 3a to 3b for each new user activity to acquire recommendations.

CHAPTER 2: RELATED WORKS

2.1 E-commerce Recommender Systems

In this section, we present the existing recommender systems that have contributed overtime to e-commerce recommendation including **(Liu, Lai & Lee, 2009)**, **(Li et al., 2011)**, **(Dai & Zeng, 2016)**, **(Bhatta, Ezeife & Butt, 2019)**, **(Yang et al., 2020).** These systems incorporate some of the procedures such as association rule mining and collaborative filtering discussed in this thesis.

2.1.1 Liu, Y., Lai, C., & Lee, C. (2009). A hybrid of sequential rules and collaborative filtering for product recommendation

The LiuRec09 paper addresses the limitation of sequential rule methods by proposing a hybrid approach that incorporates Recency, Frequency, Monetary (RFM) segmentation and transaction matrix clustering. The goal is to enhance product recommendations by considering both historical purchase sequences and current user behaviour. The proposed method involves customer clustering based on RFM values, creating a binary transaction matrix, and leveraging association rule mining for personalized recommendations.
As an illustrative example, consider the sample e-commerce data provided in **Table 2.1**

TransactionID	CustomerID	Timestamp	Product	Quantity	Price($)
1	C001	Jan 5, 2018	Perfumes	2	50
2	C001	Feb 10, 2018	Dresses	1	75
3	C002	Jan 8, 2018	Shoes	3	120
4	C002	Feb 15, 2018	Perfumes	1	40
5	C003	Mar 20, 2018	Skincare	2	30
6	C003	Jan 10, 2018	Dresses	2	90
7	C004	Feb 18, 2018	Shoes	1	60
8	C004	Mar 25, 2018	Skincare	3	75
9	C005	Apr 30, 2018	Perfumes	2	80
10	C005	Apr 14, 2018	Dresses	1	50

Table 2.1 Sample Ecommerce Data for LiuRec09 Example

Step 1 : Customer Clustering using RFM.

In this step, customers are grouped based on Recency, Frequency, and Monetary values, assigning each customer an RFM score to quantify their purchasing behaviours as shown in **Table 2.2**

Customer ID	Recency (R)	Frequency (F)	Monetary (M)	R_ Quartile	F_ Quartile	M_ Quartile	RFM Score
C001	15	2	125	2	2	2	222
C002	92	2	160	4	2	3	423
C003	83	2	120	4	2	2	422
C004	76	2	135	3	2	2	322
C005	79	2	130	3	2	2	322

Table 2.2 RFM Clusters (LiuRec09)

Step 2 : Create Transaction Matrix

In this step Once RFM clusters of users are created, users' transaction (binary) matrix is created by analysing the list of items purchased by users, where 1 represents purchased items and 0 represents not purchased items by a user. The output transaction matrix is shown in **Table 2.3**

Customer ID	Perfumes	Dresses	Shoes	Skincare
C001	1	1	0	0
C002	1	0	1	0
C003	0	1	0	1
C004	0	0	1	1
C005	1	1	0	0

Table 2.3 Transaction Matric (LiuRec09)

Step 3 : Transaction Matrix Clustering

The transaction clustering helps to locate the customer past transaction and present transaction. Furthermore, transaction cluster represents a group of transactions with a similar item purchased by users. Utilizing the K-means clustering method, customers with similar purchasing patterns are grouped into clusters, helping identify segments with shared preferences. Using the transaction sequences from **Table 2.3** we have transaction clusters shown in **Table 2.4**

Transaction Sequence	Skincare
1100	C1
1010	C2
0101	C3
0011	C4
1100	C1

Table 2.4 Transaction Matrix Clusters (LiuRec09)

Step 4: Mining Customer Behaviour from Transaction Clusters

Sequential pattern mining is applied to extract frequent patterns from the clusters in **Table 2.4,** revealing the evolving purchasing behaviours of customers over time shown in **Table 2.5** below:

Customer ID	Locus in 1st Trans (T-2)	Locus in 1st Trans (T-1)	Behaviour Clusters in Trans (T)
C001	1	1	1
C002	1	0	1
C003	2	1	2
C004	3	1	3
C005	1	1	1

Table 2.5 Customer Behaviour Clusters (LiuRec09)

Step 5: Determine and Match Cluster Sequences of Target Customers

The cluster sequence of a target customer is compared with association rules derived from other customers, determining the best-matching locus, and calculating the fitness measure.

Customer ID	Matching Locus	Support	Confidence	Fitness Measure
C001	1	0.4	1	0.4
C002	1	0.4	1	0.4
C003	2	0.2	1	0.2
C004	3	0.2	1	0.2
C005	1	0.4	1	0.4

Table 2.6 Matched Cluster Sequences of Customers (LiuRec09)

Step 6: Recommendation

Based on the customer's cluster and frequent purchase patterns, a recommendation list is generated, suggesting the most relevant products for the target customer. The goal here is to rank the most frequently purchased products in the cluster:

1. M(r1): Perfumes
2. M(r2): Dresses
3. M(r3): Shoes

The recommendation list for the target customer is Perfumes, Dresses, Shoes.

2.1.2 Li, R., Niu, J., Chen, C., & Zhang, Y. (2011). Combining collaborative filtering and sequential pattern mining for recommendation

Traditional collaborative filtering techniques focus on identifying similarities between users and items, predominantly relying on nearest neighbours for recommendations. However, these methods often overlook the dynamic nature of user preferences over time. Recognizing the limitation of collaborative filtering in capturing evolving purchasing habits, Li, Niu, Chen, & Zhang **(2011)** introduced an innovative approach that combines collaborative filtering with sequential pattern mining for enhanced recommendation systems. To address the temporal dynamics of user purchase choices, the proposed method integrates collaborative filtering with sequential pattern mining. Beginning with a user-item rating matrix as input, the algorithm follows a multi-step process. First, it computes item-item similarity and predicts the Top-K items based on collaborative filtering. Subsequently, the Top-K items are arranged in descending order, forming a sequence representing the items purchased by users. Finally, a sequential pattern mining algorithm, such as GSP (Generalized Sequential Pattern), is applied to uncover frequent sequential patterns in the user's purchase history.

As an illustrative example, consider the user-item rating matrix provided in Table 2.7 Following the proposed methodology:

	User 1	User 2	User 3	User 4	User 5
Item 1	1	-	1	1	1
Item 2	1	2	-	-	2
Item 3	3	-	3	-	3
Item 4	-	4	-	1	-
Item 5	-	5	1	2	-

Table 2.7 User-item rating matrix (LiuRec11)

Step 1: Compute Mean-Centered User-item Matrix

The process of mean centering a user-item matrix diminishes the influence of users with numerous ratings on individual ratings. Thus, the mean centering of the user- item rating matrix (depicted in **Table 2.8**) involves subtracting each user's specific item ratings from their corresponding mean rating, as illustrated in Table 2.7

	User 1	User 2	User 3	User 4	User 5	Mean Rating
Item 1	0	0.6	0	1	1	0.8
Item 2	0.5	1.5	-	-	2	0.88
Item 3	-0.33	-	-0.33	-	3	1.66
Item 4	-	-1.5	-	1	-	0.5
Item 5	-	0.5	-1.5	2	-	-0.5

Step 2: Find Item-Item Similarity

After the computation of the mean-centered user-item rating matrix, the next step involves determining the similarities between items. This is achieved through the application of similarity functions such as Cosine similarity or the Pearson Correlation

Coefficient. The resulting item-item similarity, computed using the Pearson Correlation Coefficient from the user-item rating matrix (**Table 2.2**), is presented in **Table 2.4.** This table illustrates the degree of similarity between different items in the mean- centered rating matrix.

	User 1	User 2	User 3	User 4	User 5
Item 1	1.00	-0.81	0.14	-0.25	0.12
Item 2	-0.81	1.00	-0.16	-0.67	-0.12
Item 3	0.14	-0.61	1.00	-0.25	-0.67
Item 4	-0.25	-0.67	-0.25	1.00	-0.50
Item 5	0.12	-0.12	-0.67	-0.50	1.00

Table 2.8 Item-item similarity of mean centered matrix (LiuRec11)

Step 3: Select Items with Highest Similarities.

Choose items with the greatest similarities to the current item. During this stage, we pick the most similar items to the current one. In our example, observing that the rating for Item1 is 1.00, we identify other similar items, namely Item3 and Item5, as indicated in Table 2.9

	Closest Items
Item 1	Item3, Item5
Item 2	Item3, Item5
Item 3	Item1, Item2, Item5
Item 4	Item1, Item2, Item3, Item5
Item 5	Item1, Item2, Item3, Item4

Table 2.9 Closest Items to each Item (LiuRec11)

Step 4: Compute Predicted Ratings

Predicted ratings for each user-item pair are computed using the Pearson Correlation Coefficient resulting in the predicted user-item rating matrix (Table 2.10)

	User 1	User 2	User 3	User 4	User 5
Item 1	0.6	0.12	0.8	0.5	1.14
Item 2	1.18	2.5	0.88	1.5	2.5
Item 3	1.33	0.16	1.33	0.33	0.66
Item 4	0.5	4.5	-0.5	0.5	2.55
Item 5	-0.5	5.5	1.5	1.5	-0.5

Table 2.10 Predicted User-Item Matrix (LiuRec11)

Step 5: Create Sequence Database

Items are arranged in descending order based on their predicted ratings to create a sequence database (Table 2.11)

	User 1	User 2	User 3	User 4	User 5
Predicted Items	<I3, I1>	<I3, I4>	<I1, I3>	<I2, I4>	<I2, I4>

Table 2.11 Sequence database created by using rating value in descending order (LiuRec11)

Step 6: Apply GSP Algorithm

The GSP algorithm is then applied to the sequence database, resulting in the identification of frequent sequential patterns. A minimum support of 2 and a candidate set of I1, I2, I3, I4} is used.

The GSP follows these steps:

1. Identify 1-frequent sequences (L1) by retaining only those sequences with an occurrence or support count in the database greater than or equal to the minimum support count of 2. For instance, L1={<I1>:2, <I2>:2, <I3>:3, <I4>:3}.
2. Generate candidate sequences (Ck=2) using L(1) through the GSP join operation.
3. Evaluate and prune the candidate set C (K=2) by examining the minimum support and eliminating infrequent items.
4. Repeat the iterative process of candidate generation and pruning until the resulting candidate set (Ck) and pruned set (Lk) yield an empty set, indicating the completion of finding frequent sequences.
5. Output the frequent sequences as the union of L1, L2, and so forth, up to Ln.

1-Sequence	<I3>: 2, < I1>: 1, <I4>: 1, <I2>: 2
2-Sequence	<I3, I4>: 1, <I2, I4>: 2

Table 2.12 Frequent Sequences (LiuRec11)

From the 2-sequence <I2, I4>, it can be inferred that the recommended sequence for User2 is to first adopt Item2 and then Item4. The GSP algorithm helps uncover frequent sequential patterns in the user's predicted purchase history.

2.1.3 Dai, J., & Zeng, B. (2016). An association rule algorithm for online e-commerce recommendation service.

Dai and Zeng propose a novel association rule algorithm tailored for online e- Commerce recommendation services, aiming to overcome the limitations of traditional methods. The problem addressed involves the inefficiency of existing algorithms that focus solely on sales volume without considering item profitability. To tackle this, the authors introduce the profit-support association rule algorithm. This approach integrates profit considerations into the association rule mining process, providing online shop managers with a tool to enhance profits and offer personalized product recommendations based on both sales volume and profitability.

As an illustrative example, consider the transaction database provided in **Table 2.13** Following the proposed methodology:

	Transactions
T1	{Item1, Item2, Item3}
T2	{Item2, Item4, Item5}
T3	{Item1, Item3, Item5}
T4	{Item2, Item3, Item4}

Table 2.13 Transaction Database (JiabeiRec16)

Step 1: Generation of 1-Itemset

Given the transaction data and an appointed profit of $100, the algorithm calculates the total profit for each item and sets a minimum total profit of $50 shown in Table 2.9. In this step, Item4 is excluded from further consideration as show in Table2.10

	Amount	Unit Profit	Total Profit	Minimum Total Profit
Item 1	2	$30	$60	$50
Item 2	3	$20	$60	$50
Item 3	3	$15	$45	$50
Item 4	1	$10	$10	$50
Item 5	2	$25	$50	$50

Table 2.14 1-Itemset (JiabeiRec16)

Step 2: Minimum Support for Each Item Using Profit

The algorithm calculates the Minimum Profit Support (MPS) for each item based on the appointed profit.

	MPS
Item 1	1.66
Item 2	2.5
Item 3	3
Item 5	2

Table 2.15 Minimum Profit Support (MPS) for each Item (JiabeiRec16)

Step 3: Association Rule Mining

Using the calculated MPS values from Table 2.10, the algorithm conducts association rule mining, generating large itemsets with multiple minimal supports.

It generates large itemsets with multiple minimal supports, ensuring that the association rules reflect both the profit considerations and the support thresholds for each item.

The algorithm derives from the Apriori algorithm, but with a focus on profit as a criterion for minimum support.

Association rules mined based on the provided example could include:

1. Rule A: {Item1, Item2} ➔ {Item3} (Support: 2, Confidence: 0.67)
2. Rule B: {Item2, Item5} ➔ {Item4} (Support: 1, Confidence: 1.0)

These rules represent associations between items with their corresponding support (frequency) and confidence values.

Step 4: Recommendation

The association rules generated in step 3 is used as input for the recommendation step. For each user's transaction history, the algorithm identifies applicable association rules that match the items in the user's cart or recent purchases. The confidence values help determine the strength of the association, guiding the recommendation engine to suggest additional items.

1. Given a user has purchased {Item1, Item2}, the recommendation system might suggest adding {Item3} to the cart based on Rule A.
2. Given user has purchased {Item2, Item5}, Rule B indicates a high-confidence association with {Item4}, leading to a recommendation for {Item4}.

These recommendations consider both the historical purchase patterns and profitability, aiming to enhance the user experience and increase profits for the e- commerce platform.

2.1.4 Bhatta, R., Ezeife, C. I., & Butt, Z. A. (2019). Mining the sequential pattern based on daily purchase history for the collaborative filtering

The primary objective of the Historical Sequential Recommendation (HSPRec19), proposed by Bhatta, Ezeife, and Butt in 2019, is to extract frequent sequential patterns from E-commerce historical data. The aim is to enhance a user-item rating matrix through the utilization of discovered patterns. The authors introduced two algorithms, HSPRec19 (Historical Sequential Recommendation), and SHOD (Sequential Historical Periodic Database) System. These algorithms contribute to creating a daily purchase sequence database based on the consequential bond between click and purchase databases.

As an illustrative example, consider a user-item purchase frequency matrix (Table 2.20) containing historical purchase information, the goal is to identify sequential patterns from the consequential bond of click and purchase datasets. This will enrich the user-item matrix using HSPRec19 and SHOD algorithms.

Step 1: Create a user-item frequency matrix from historical purchase.

This step involves constructing a matrix that represents the historical purchases made by users. Each cell in the matrix indicates the frequency of a specific item purchased by a particular user.

	Milk	Bread	Cream	Cheese	Honey	Butter
User 1	1	0	2	2	1	3
User 2	0	1	1	1	0	2
User 3	2	0	1	0	1	2

Table 2.16 Sample User-item matrix (HSPRec19)

Step 2: Convert historical purchase to the sequential database using SHOD on daily purchases.

The historical purchase information is transformed into a daily purchase sequential database using the SHOD algorithm. This process organizes the purchase data into a sequence based on daily transactions.

	Purchase Sequence
Day 1	Milk, Bread, Cream, Cheese, Honey
Day 2	Butter, Cheese, Milk
Day 3	Cream, Cheese, Milk
Day 4	Honey, Butter, Cream

Table 2.17 Daily Purchase Sequence using SHOD (HSPRec19)

Step 3: Create frequent sequential purchase patterns from daily sequential database using GSP algorithm.

Utilizing the GSP (Generalized Sequential Pattern) algorithm, frequent sequential purchase patterns are extracted from the daily sequential database. These patterns represent frequently occurring sequences of purchases.

Sequential Patterns	**Support**
{Milk}	2
{Cheese, Milk}	2
{Cream, Cheese, Milk}	2
{Butter, Milk}	2
{Honey, Cheese}	1

Table 2.18 Sequential Patterns with Support (HSPRec19)

Step 4: Fill purchase information in user-item frequency matrix using sequential purchase rule.

The user-item frequency matrix is updated by incorporating information from the sequential purchase rules obtained in the previous step. This enriches the matrix with additional purchase data inferred from the discovered sequential patterns.

	Milk	Bread	Cream	Cheese	Honey	Butter
User 1	1	0	3	3	1	4
User 2	0	1	1	2	0	3
User 3	3	0	2	0	1	3

Table 2.19 Updated User-item matrix from step 3 (HSPRec19)

Step 5: Create click sequential database from the consequential bond.

A daily sequential database is created for user clicks based on the consequential bond between clicks and purchases. This database captures the sequential relationships between clicked items.

Sequential Patterns = {Milk, Cheese, Butter, Cream, Cheese} { Honey, Cream, Butter}

Step 6: Use SPM algorithm on user click sequence.

Applying the Sequential Pattern Mining (SPM) algorithm, n-frequent click sequential patterns are identified. These patterns reveal the frequently occurring sequences of items that users click on.

Sequential Patterns	Support
{Milk}	1
{Cheese}	2
{Butter}	2
{Cream}	1
{Honey}	1
{Milk, Cheese}	1
{Butter, Cheese}	1
{Honey, Butter}	1
{Cream, Cheese, Milk}	1
{Cheese, Cream, Milk}	1

Table 2.20 Click Sequential Patterns and Support (HSPRec19)

Step 7: Create sequential rules from frequent click sequential pattern.

Sequential rules are generated from the n-frequent click sequential patterns. These rules highlight associations and dependencies between clicked items, providing insights into user behaviours.

Rules	Confidence
{Milk} -> {Cheese}	1
{Cheese} -> {Butter}	0.5
{Butter} -> {Cheese}	0.5
{Cream} -> {Cheese}	1
{Honey} -> {Butter}	1
{Milk, Cheese} -> {Butter}	1
{Butter, Cheese} -> {Milk}	1
{Honey, Butter} -> {Cream}	1
{Cream, Cheese, Milk} -> {Butter}	1
{Cheese, Cream, Milk} -> {Butter}	1

Table 2.21 Sequential Rules (HSPRec19)

Step 8: Recommend item for click when purchase is not happened.

Items are recommended for users based on their click sequences, particularly when a user clicks on items but does not make a purchase. These recommendations aim to enhance the user experience and potentially increase conversion.

The recommended item for the click sequence for {Cheese, Milk} is Cream.

Step 9: Compute Click Purchase Pattern (CPS) similarity.

CPS similarity is calculated by considering the frequency and sequence of click and purchase patterns. This step assesses the similarity between the items clicked and purchased, incorporating parameters α and β.

LCSR (X, Y) = |common(X, Y)| / max(|X|, |Y|) = 0.55

FS(X, Y) = cosine({2, 1, 1, 1}, {1, 0, 2, 2, 1, 3}) = 0.97

CPS-Sim (X, Y) = 0.8 * 0.55 + 0.2 * 0.97 = 0.634

Step 10: Assign Click Purchase (CPS) similarity value to the purchase patterns.

The computed CPS similarity values are assigned to the purchase patterns, providing a measure of similarity between clicks and purchases. This information is then used to weight the purchase patterns. Purchase Pattern {Cream, Butter, Milk} has CPS similarity 0.634 and purchase pattern {Honey, Butter} has CPS similarity 0.634

Step 11: Assign weighted purchase patterns to Weighted Frequent Purchase Pattern Miner.

Weighted purchase patterns are assigned to the Weighted Frequent Purchase Pattern Miner module, and item weights are computed based on the formula presented. This step involves considering the support of each item in the patterns.

Support (Milk) = 3, Support (Cream) = 3, Support (Cheese) = 3, Support (Honey) = 4, Support (Butter) = 3

R_Milk = (0.634 + 0.516 + 0.516) / 3 = 0.55

R_Cream = (0.634 + 0.516 + 0.198) / 3 = 0.44

R_Cheese = (0.516 + 0.562 + 0.562) / 3 = 0.54

R_Honey = (0.634 + 0.516 + 0.516 + 0.198) / 4 = 0.46

R_Butter = (0.634 + 0.634 + 0.562) / 3 = 0.6

Step 12: Use the weight of items to make the user-item matrix rich.

The weights obtained for each item are used to enrich the user-item matrix. This step aims to provide a more nuanced representation of user preferences by considering the weighted influence of items.

Step 13: Normalize rich user-item purchase frequency matrix.

The enriched user-item purchase frequency matrix is normalized to ensure that values are scaled appropriately. This normalization step facilitates meaningful comparisons and analyses of user-item interactions.

	Milk	Bread	Cream	Cheese	Honey	Butter
User 1	0.2	0	0.6	0.6	0.2	0.8
User 2	0	0.3333	0.333	0.667	0	1
User 3	0.6	0	0.4	0	0.2	0.6

Table 2.22 User-item frequency matrix (HSPRec19)

2.1.5 Yang, J., Chen, H., Xiong, S., Yang, Z., & Jiang, Y. (2020). Custom Data Mining Association Rules of Nixing Pottery Product Recommendation System

The paper introduces a recommendation system for Nixing Pottery products using custom data mining association rules. Focusing on user interests and buying behaviours, the system employs a hardware topology for efficient handling of consumer demand data. The software design encompasses modules for product management, product data pre-processing, association rule base generation, and product recommendation. The study indicates a high recommendation hit rate, understanding user interests effectively and enhancing product recommendation rates.

As an illustrative example, consider three products: NP101, NP102, NP103 and the following steps below:

Step 1: Product Management Module Design

This module handles user-related operations, shopping cart functions, order processing, and refund procedures. Example operations:

1. User Registration: Assume a new user, "PotteryEnthusiast," completes registration.
2. Shopping Cart: PotteryEnthusiast adds three Nixing Pottery items (ProductID: NP101, NP102, NP103) to the shopping cart.
3. Order Placement: PotteryEnthusiast places an order, and the system checks for account balance.
4. Refund: The user initiates a refund for a purchased pottery item before shipment

Step 2: Product Data Pre-processing Module

Involves data preparation, classification, and pre-processing, ensuring efficient storage and modification of goods.

1. Data Preparation: Assume original data on Nixing Pottery transactions and product information are entered into an Excel file.
2. Data Classification: The data is classified based on Nixing pottery information hierarchy as shown in **Table 2.23**
3. Data Pre-processing: Shelf status changes and other treatments are applied to ensure data primality.

ProductID	Category	Shelf Status
NP101	Teapots	On Shelf
NP102	Bowls	Off Shelf
NP103	Sculptures	On Shelf
NP104	Dinnerware	On Shelf

Table 2.23 Product Shelf Status (YangRec20)

Step 3: Association Rule Library Generation Module

Utilizes association algorithms to mine frequent patterns and generate association rules, maintaining a dynamic rule base. Frequent patterns are converted into association rules based on credibility and sorted in descending order.

Step 4: User Interest Analysis Module

Detects and analyses user interests, creating an interest library based on browsing behaviours and association algorithm calculations. For example:

1. Interest Calculation: For a new user, "PotteryFanatic," browsing behaviours on Nixing Pottery pages is analysed.
2. Interest Library: PotteryFanatic's interest values for specific pottery items are calculated and stored in the interest library.

User ID	Product ID	Interest
User 3	NP101	0.65
User 3	NP104	0.42

Table 2.24 User Interest Analysis (YangRec20)

Step 5: Recommendation

Essential interface for the recommendation system, providing lists of recommended products based on user interest values. Recommendation List: Based on PotteryFanatic's interest values, a recommendation list is generated, showing top Nixing Pottery products.

Product ID	Interest	Recommendation
NP101	0.65	High
NP104	0.42	Medium
NP102	0.21	Low

Table 2.25 Recommended Products (YangRec20)

The goal here is to sort pottery items based on interest values and recommend the top ones.

2.2 Recommender Systems Addressing Cold Start Problem

Earlier in **Chapter 1**, we discussed the cross-site cold start problem that challenges recommender systems in providing accurate recommendations for users with limited or no history (Panteli & Boutsinas, 2023) and in this section, we discuss some of those systems (Seroussi, Bohnert & Zukerman ,2011), (Lin et al., 2013) that tried to close this gap and how.

2.2.1 Seroussi, Y., Bohnert, F., & Zukerman, I. (2011). Personalised rating prediction for new users using latent factor models.

The paper titled "Personalised Rating Prediction for New Users Using Latent Factor Models" by Y. Seroussi, F. Bohnert, and I. Zukerman (2011) addresses the challenge of inaccurate rating predictions for new users in the context of personalized recommendations. The authors propose extensions to the basic matrix factorization (MF) algorithm, introducing matrix factorization with user attributes **(MFUA).**

They consider both demographic attributes provided by users and attributes inferred from user-generated texts to improve the accuracy of predictions for users with limited ratings. The primary goal is to enhance the accuracy of rating predictions for new users who have submitted only a few ratings. The authors aim to achieve this by extending the MF algorithm to incorporate user attributes, considering both explicit demographic information and implicit attributes inferred from user-generated texts.

As an illustrative example let us consider the following dataset:

User	Age	Gender
User 1	25	Male
User 2	30	Female
User 3	22	Male

Table 2.26 Sample User Demographics (Seroussi11)

User	Item A	Item B	Item C
User 1	4	3	-
User 2	5	-	2
User 3	-	4	-

Table 2.27 Sample User-Item Matrix (Seroussi11)

Step 1: Matrix Factorization (MF)

Input:

Rating matrix R (user-item interactions)

Regularization parameters λ1, λ2, λ3, λ4

Learning rate parameters

Algorithm Minimizing Equation by Stochastic Gradient Descent

Input: R, D, λ1, λ2, λ3, λ4, γ0, λγ, numSteps

Output: P, Q, b(U), b(I)

Initialization:

Random values are assigned to the factor matrices P(user latent factors) and Q(item latent factors), as well as bias vectors b(U)(user biases) and b(I)(item biases).

Learning rate parameters (γ, λ values) are set.

Calculate the mean rating (μ) of all known ratings in the training set.

Stochastic Gradient Descent (SGD) Loop: The algorithm iterates over a specified number of steps (numSteps) to minimize the prediction error through Stochastic Gradient Descent.

- For each known rating rui in the training set:
- Calculate the prediction error eui as the actual rating minus the predicted rating.
- Update the user bias b(U)u. and item bias b(I)i based on the error and regularization terms.
- Update the user latent factors pu and item latent factors qi for each dimension d based on the error and regularization terms.
- Adjust the learning rate (γ) using a decay factor (λγ).

Output:

Learned factor matrices P and Q, and bias vectors b(U) and b(I).

Step 2: Matrix Factorization with User Attributes (MFUA)

Input:

Rating matrix R (user-item interactions)

Learned factor matrices P and Q from Step 1

Regularization parameters λ5, λ6

Learning rate parameters

Algorithm Minimizing Equation by Stochastic Gradient Descent

Input: R, D, λ1, λ2, λ3, λ4, λ5, λ6, γ0, λγ, numSteps, Q, b(I), P(a|u) for all $1 \leq a \leq L$ and $1 \leq u \leq N$

Output: Y, b(A)

Initialization: Random values are assigned to the latent factor matrix Y(capturing user attribute factors) and the bias vector b(A) (user attribute biases).

Learning rate parameters (γ, λ values) are set.

Calculate the mean rating (μ) of all known ratings in the training set.

Stochastic Gradient Descent (SGD) Loop : The algorithm iterates over a specified number of steps (numSteps) to minimize the prediction error through Stochastic Gradient Descent.

- For each known rating rui in the training set:
- Calculate the prediction error eui as the actual rating minus the predicted rating.
- Update the user attribute bias b(A)a for each attribute a based on the error and regularization terms.
- Update the latent factors ya for each dimension d and each attribute a based on the error and regularization terms.
- Adjust the learning rate (γ) using a decay factor (λγ).

Output:

Learned factor matrix Y and bias vector b(A).

Step 3: Prediction for User 3 on Item A

Input: Learned factor matrix Y and bias vector b(A).

Now, let us use the learned models to predict the rating for User 3 on Item A:

Predicted Rating(User 3, Item A) = μ + b(I)_A + P(Age|User 3) * y_A + P(Gender|User 3) * y_G

Here, we use the learned parameters and attributes for User 3.

2.2.2 Lin, J., Sugiyama, K., Kan, M., & Chua, T. (2013). Addressing cold start in app recommendation: Latent user models constructed from Twitter followers.

As the number of mobile applications (apps) continues to grow, users face challenges in discovering apps aligned with their interests. Traditional recommender systems, particularly collaborative filtering (CF), struggle with the "cold start" problem for new apps that lack sufficient ratings. In this paper, the authors propose a novel method to address the cold-start problem using information from Twitter. By leveraging Twitter followers, the proposed approach constructs latent user models to provide relevant recommendations for apps without prior ratings. The method outperforms other state-of-the-art recommendation techniques in cold-start situations.

As an illustrative example consider the sample data :

User	App1	App2	App3	App4	App5
User 1	5	0	4	-	-
User 2	0	3	-	4	5
User 3	-	5	3	-	1

Table 2.28 Sample User Data (LinRec13)

Step 1: Targeting the Cold-Start Problem

The authors distinguish between in-matrix prediction (items with at least one user rating) and out-of-matrix prediction (newly released items with no ratings). This work primarily

focuses on the more challenging out-of-matrix prediction, referred to as the cold-start problem.
In this step, Identify the cold-start problem, focusing on out-of-matrix prediction for newly released items with no prior ratings.

Step 2: Apps and their Twitter-Followers
The paper highlights the relationship between apps and their Twitter-followers. Using Twitter handles, the authors extract the IDs of app-associated followers. The method creates pseudo-documents containing follower IDs and applies latent Dirichlet allocation (LDA) to generate latent groups, representing combined interests.
In this step Consider the Twitter handles of apps to extract the IDs of their Twitter-followers. Create pseudo-documents containing follower IDs.

Step 3: Pseudo-Documents and Pseudo-Words
To estimate the probability of a user liking an app, the authors introduce the concepts of pseudo-documents and pseudo-words. Pseudo-documents represent users, and pseudo-words contain follower IDs and binary preference indicators. LDA is adapted for collaborative filtering using these constructs. In this step Convert user ratings into binary preferences and construct pseudo-documents and pseudo-words based on follower IDs.
Pseudo-Document for User1:
- Pseudo-Word: TwitterFollower1, Liked
- Pseudo-Word: TwitterFollower3, Liked

Pseudo-Document for User2:
- Pseudo-Word: TwitterFollower2, Disliked
- Pseudo-Word: TwitterFollower4, Liked
- Pseudo-Word: TwitterFollower5, Liked

Pseudo-Document for User3:
- Pseudo-Word: TwitterFollower2, Liked
- Pseudo-Word: TwitterFollower3, Liked

- Pseudo-Word: TwitterFollower5, Disliked

Step 4: Constructing Latent Groups

Given sets of pseudo-documents, LDA generates probability distributions over latent groups for each pseudo-document. These latent groups capture combined interests of Twitter-followers, forming a crucial component of the recommendation algorithm. In this step Apply LDA to generate latent groups representing combined interests of Twitter-followers.

Latent Group 1:

- TwitterFollower1, TwitterFollower3

Latent Group 2:

- TwitterFollower2, TwitterFollower4, TwitterFollower5

Step 5: Estimation of the Probability

To estimate the probability of a target user liking an app, the authors employ an averaging method. The probability is the expectation of how Twitter-followers collectively like the app, considering both follower IDs and binary preferences. In this step Use an averaging method to estimate the probability of a target user liking an app, considering both follower IDs and binary preferences.

Probability of Liking an App for User1:

- Latent Group 1: (0.5) * Probability of Liking TwitterFollower1 + (0.5) * Probability of Liking TwitterFollower3

Probability of Liking an App for User2:

- Latent Group 2: (0.33) * Probability of Liking TwitterFollower2 + (0.33) * Probability of Liking TwitterFollower4 + (0.33) * Probability of Liking TwitterFollower5

Probability of Liking an App for User3:

- Latent Group 1: (0.33) * Probability of Liking TwitterFollower2 + (0.33) * Probability of Liking TwitterFollower3 + (0.33) * Probability of Liking TwitterFollower5

Step 6: Pseudo-Words Estimation

Two frameworks for estimating pseudo-words are introduced: one assuming a uniform distribution over Twitter-followers and another giving more importance to influential followers using TwitterRank. In this step : Estimate pseudo-words using both frameworks: assuming a uniform distribution and giving more importance to influential Twitter-followers using TwitterRank.

Probability of Liking TwitterFollower1 = 1 / 2 = 0.5

Probability of Liking TwitterFollower2 = 1 / 3 = 0.33

Probability of Liking TwitterFollower3 = 1 / 2 = 0.5

Probability of Liking TwitterFollower4 = 1 / 3 = 0.33

Probability of Liking TwitterFollower5 = 1 / 3 = 0.33

2.3 Transfer Learning Cross Domain Recommender Systems

In this section we discuss the methodology of transfer learning in cross domain recommendation **(Pan et al., 2010)** and then some recommender systems such as **(Anwar & Uma, 2022)** that leverage this method in making cross domain recommendations.

2.3.1 Pan, W., Xiang, E. W., Liu, N. N., & Yang, Q. (2010). Transfer learning in collaborative filtering for sparsity reduction.

The paper addresses the data sparsity issue in collaborative filtering (CF) for recommender systems. It introduces a transfer learning approach called Coordinate System Transfer (CST) to alleviate data sparsity in the target domain by leveraging related auxiliary data from more mature application domains. The CST method integrates both user and item knowledge, considering heterogeneous user feedback types, through a matrix-based transfer learning framework. The solution involves discovering principal coordinates of users and items in the auxiliary domain and transferring them to the target domain, thus reducing the impact of data sparsity.

Let us run through an example to see how this works.

Target Domain:

Users [U1, U2, ..., Un]

Items [I1, I2, ..., Im]

Sparse Rating Matrix R (n x m)

Auxiliary Domain:

Users [U1, U2, ..., Un]

Items [I1, I2, ..., Im]

Auxiliary Data Matrices R(1), R(2) (n x m)

Step 1: Coordinate System Construction

SVD on Auxiliary Data:

Apply sparse Singular Value Decomposition (SVD) on auxiliary data R(1) and R(2) to obtain principal coordinate systems U0 and V0.

U0, V0 = SparseSVD(R(1)), SparseSVD(R(2))

Initialize Target Coordinate Systems:

Initialize target coordinate systems U and V with U = U0 and V = V0.

Step 2: Coordinate System Adaptation (CST)

Optimization Problem:

Formulate an optimization problem for CST, considering sparsity reduction and adaptation of coordinate systems.

$\min U, V, B \; \|Y \;\; (R - UBVT)\| + \rho u/2 \; \|U - U0\|2F + \rho v/2 \; \|V - V0\|2F$

s.t. $UT\,U = I$, $VT\,V = I$

Adaptation Process:

Use an alternative method (Algorithm 1) to iteratively update U, V, and B until convergence.

Fix U, V, and estimate B.

Fix B, and update U, V via alternative gradient descent on the Grassmann manifold.

Parameters:

Adjust trade-off parameters ρu and ρv for confidence in auxiliary data.

Iterations (Repeat):
Repeat the optimization process for a predefined number of iterations or until convergence.
Outcome: The result is the adapted coordinate systems U and V, which can be used to predict missing values in the target domain's rating matrix R, effectively reducing the impact of data sparsity.

This walkthrough outlines the Coordinate System Transfer (CST) algorithm, illustrating how it addresses the data sparsity problem in collaborative filtering by transferring knowledge from auxiliary domains to the target domain. The algorithm involves the construction and adaptation of coordinate systems, leveraging sparse SVD and an optimization process. Adjusting parameters allows fine-tuning the balance between target and auxiliary domain information.

2.3.2 Anwar, T., & Uma, V. (2022). CD-SPM: Cross-domain book recommendation using sequential pattern mining and rule mining

The paper introduces CD-SPM, a Cross-Domain Sequential Pattern Mining system for book recommendation in the e-commerce domain. The system incorporates techniques such as ontology, sequential pattern mining, collaborative filtering, and rule mining to generate accurate and diverse recommendations. The proposed approach aims to address the challenges of the new user problem, data sparsity, and provide cross-domain recommendations by leveraging sequential patterns and semantic similarity.

The authors aim to tackle the challenge of making effective book recommendations in the e-commerce domain by leveraging data from multiple domains, specifically movies and books. They aim to overcome the limitations of traditional recommender systems, such as the new user problem and data sparsity, and provide accurate and diverse recommendations based on cross-domain datasets.

The CD-SPM system utilizes several techniques to generate book recommendations. It incorporates ontology and the Wpath method to calculate semantic similarity between items from different domains. Collaborative filtering is used to filter similar items and predict missing user ratings using the item-rating matrix. Sequential pattern mining is performed using the PrefixSpan algorithm to retrieve frequent sequences of user preferences. Lastly, the Topseq Rules algorithm is applied to order the items in frequent sequences and generate the most preferred sequence of books as recommendations.

An example scenario could be a user who has a history of watching action movies and wants to receive book recommendations. The CD-SPM system leverages sequential pattern mining on the integrated dataset to discover frequent patterns such as "watching action movie A" followed by "reading book X." By applying collaborative filtering and considering the semantic similarity between movies and books using the Wpath method, the system generates recommendations for books that align with the user's preferences, such as action novels or books related to the themes of the movies they have watched.

Let us see an example of how their proposed system works:

User	Movies
User1	Action Movie A, Comedy Movie B
User2	Action Movie A, Romantic Movie C
User3	Comedy Movie B, Romantic Movie C
User4	Action Movie A, Comedy Movie B, Romantic Movie C

Table 2.29 Sample Movie Data (AnwarRec22)

User	Books
User1	Book X, Book Y
User2	Book X, Book Z
User3	Book Y, Book Z
User4	Book X, Book Y, Book Z

Table 2.30 Sample Book Data (AnwarRec22)

Step 1: Integration of datasets

The movie dataset and book dataset are combined into a single integrated dataset, considering user preferences across both domains.

User	Books
User1	Action Movie A, Comedy Movie B, Book X, Book Y
User2	Action Movie A, Romantic Movie C, Book X, Book Z
User3	Comedy Movie B, Romantic Movie C, Book Y, Book Z
User4	Action Movie A, Comedy Movie B, Romantic Movie C, Book X, Book Y, Book Z

Table 2.31 Integrated Data from Book and Movie Domain (AnwarRec22)

Step 2: Sequential Pattern Mining

The CD-SPM algorithm performs sequential pattern mining on the integrated dataset to discover frequent patterns of user preferences.

Frequent Patterns:
"Action Movie A" followed by "Book X"
"Comedy Movie B" followed by "Book Y"
"Romantic Movie C" followed by "Book Z"

Step 3: Collaborative Filtering

Collaborative filtering is applied to generate recommendations based on the frequent patterns discovered in the previous step.

- User1 Recommendations:

Since User1 watched "Action Movie A" and the frequent pattern is "Action Movie A" followed by "Book X," the algorithm recommends books related to action movies. For example, it may suggest an action novel, or a book based on the themes of "Action Movie A."

- User2 Recommendations:

As User2 watched "Action Movie A" and the frequent pattern is "Action Movie A" followed by "Book X," the algorithm recommends books related to action movies.

Similarly, it suggests books related to romantic movies based on the pattern "Romantic Movie C" followed by "Book Z."

- User3 Recommendations:

User3, who watched "Comedy Movie B," is recommended books related to comedy or humor based on the pattern "Comedy Movie B" followed by "Book Y." Additionally, User3 can receive recommendations for books related to romantic movies based on the pattern "Romantic Movie C" followed by "Book Z."

- User4 Recommendations:

Since User4 watched multiple movies and read multiple books, the algorithm considers all the frequent patterns found in the integrated dataset. For example, it can recommend books related to action movies based on the pattern "Action Movie A" followed by "Book X," as well as books related to comedy movies based on the pattern "Comedy Movie B" followed by "Book Y." Similarly, it suggests books related to romantic movies based on the pattern "Romantic Movie C" followed by "Book Z."

The CD-SPM algorithm generates personalized recommendations for each user based on their movie preferences and aligns those preferences with relevant books using sequential pattern mining and collaborative filtering techniques. The recommendations are tailored to the user's preferences and the patterns discovered in the integrated dataset. The CD-SPM system presented in the paper provides a comprehensive solution for cross-domain book recommendations in the e-commerce domain. It addresses the challenges of the new user problem and data sparsity by leveraging ontology, collaborative filtering, and sequential pattern mining. The integration of multiple techniques allows for accurate and diverse recommendations based on cross-domain datasets. However, in the context of the current thesis, which focuses on integrating social media activity and e-commerce data, the gap lies in exploring the incorporation of social media data as an additional domain and utilizing the generated patterns and rules to enhance recommendation accuracy and relevance specifically in the context big data e-commerce and social media era.

2.4 Joint Matrix Factorisation Cross Domain Recommender Systems

In this section we present the recommender systems (**Wang et al., 2015), (Gao et al., 2020)** that apply joint matrix factorisation as the second approach in making cross domain recommendations as well as the top closest existing system (**Gao et al., 2023)** to the proposed system in this thesis.

2.4.1 Wang, J., Zhao, W. X., He, Y., & Li, X. (2015). Leveraging Product Adopter Information from Online Reviews for Product Recommendation.

The paper addresses the challenge of deriving implicit demographic information of product adopters from online product reviews. It proposes a novel approach that extracts adopter mentions from reviews, categorizes them into demographic user groups, and leverages this information for product recommendation.

Step 1 : Extraction of Product Adopter Mentions:

a. Bootstrapping-based Extraction Method: Due to the volume of review data and informal writing styles, unsupervised methods are preferred. A bootstrapping approach iteratively learns linguistic patterns and extracts adopter mentions. Patterns like "buy somebody something" are learned and applied to extract adopter mentions.
b. Pattern Filtering: A pattern filtering step reduces spurious patterns using the Jaccard coefficient to measure similarity among extracted phrases. Without this step, extraction accuracy drops significantly.
c. Extraction Results: After the bootstrapping process, 45 patterns are derived. Human judges verify the extraction, achieving a precision of 88.5%. Ambiguous mentions are removed, resulting in 259 adopter mentions.
d. Statistics Analysis: 10.8% of 139 million reviews contain at least one adopter mention. Distribution analysis shows that 48.5% of products and 25.4% of users have at least 10 reviews with adopter mentions

Step 2: Grouping Adopter Mentions into Categories:

Adopter mentions are grouped into categories based on demographic features like age and sex.

Six user categories are identified: Children, Young Female, Old Female, Young Male, Old Male, and Colleagues.

Ambiguous mentions are removed for better categorization.

Step 3: Estimating User-Product Conditional Preference Probabilities:

a. Two-Step Generative Approach: The paper proposes a two-step approach to estimate user-product conditional preference probabilities. Users first select an adopter-related category and then a product within that category.
b. Preference Biased Matrix Factorization (PBMF): Matrix factorization is chosen as the baseline for recommendation. A preference biased matrix factorization (PBMF) is introduced to incorporate confidence levels into the model. Confidence levels are based on the user-product conditional probabilities.
c. Parameter Learning: The PBMF model involves learning parameters {xu} and {ye}. Analytic expressions for parameter updates are derived.

The proposed framework is shown to be feasible and effective through experiments on 15 million reviews from JINGDONG. The incorporation of adopter-related information into the recommendation framework improves the performance. This summary provides an overview of the paper's key contributions, methodologies, and findings. It condenses the content for a quick understanding of the paper. For in-depth details, It is recommended to refer to the complete paper.

2.4.2 Gao, C., Lin, T. -H., Li, N., Jin, D., & Li, Y. (2020). Item Recommendation for Online Social

Online platforms can be categorized into two distinct domains: information-oriented and social-oriented. The former, exemplified by platforms like Trip.com and Amazon, prioritize user-item interactions, while the latter, typified by social networking services (SNSs) like Facebook and Twitter, thrive on rich user-user connections. Despite their inherent differences, these two domains intersect through a subset of users known as bridge users. In this study, the authors tackle the novel challenge of cross-domain social recommendation, specifically, the task of suggesting pertinent items from information domains to potential users within social networks. Notably, this problem has received limited attention in prior research. This paper focuses on the challenge of cross-domain social recommendation, which involves recommending relevant items from information-oriented domains (e.g., e-commerce) to users in social-oriented domains (e.g., social media) where user-item interactions are sparse. The key issue is that very few users overlap between these domains, making traditional recommendation methods ineffective.

To address this problem, the paper introduces a novel approach called Neural Social Collaborative Ranking (NSCR). NSCR combines neural collaborative filtering (NCF) with graph regularization techniques to model the interactions between users, items, and their attributes in the information domain. It also leverages the embeddings of bridge users (those overlapping in both domains) to guide the embedding learning of social users. This way, it bridges the gap between the information and social domains, enabling effective cross-domain social recommendation.

In summary, the paper defines and tackles the novel challenge of cross-domain social recommendation, proposing NSCR as an innovative solution that combines deep neural networks and graph regularization to address this problem effectively.

The GaoRec20 system focuses on enhancing recommendations by incorporating user attributes and social connections. This slide highlights the initial steps of the system, including embedding users, items, and attributes into dense vectors and employing pairwise pooling to capture correlations. The goal is to create meaningful representations of users, items, and attributes in vector form and capture correlations between them through pairwise pooling. This process sets the foundation for subsequent stages of the recommendation system. In the embedding layer, users, items, and attributes are represented as dense vectors. For example, User1's initial embedding (pu_0) is [0.2, 0.5].

The pairwise pooling layer calculates correlations by squaring these vectors. For User1, pu becomes [0.04, 0.25] after applying the formula pu = pu * pu.

Let us try an example with a sample Input Data.

Users (U1): [User1 (ID: 1), User2 (ID: 2), User3 (ID: 3)
Items (I): [ItemA (ID: 10), ItemB (ID: 20)]
Attributes (A): [User1 Attributes: {Female, 25-30 years old}, User2 Attributes: {Male, 31-35 years old}
User3 Attributes: {Female, 20-25 years old}]
Social Connections (S): [User1 is friends with User2 (1 -> 2), User1 is friends with User3 (1 -> 3), User2 is friends with User3 (2 -> 3)}]

Let us assume we have initial embeddings (these would be learned during training, but we will use arbitrary values here for illustration):

Step 1: Embedding Layer
In the embedding layer, we represent users, items, and attributes as dense vectors.
Let us assume we have initial embeddings (these would be learned during training, but we will use arbitrary values here for illustration):

- User1's initial embedding (pu_0) = [0.2, 0.5]
- User2's initial embedding (pu_0) = [0.6, 0.7]
- User3's initial embedding (pu_0) = [0.3, 0.8]
- ItemA's initial embedding (qi_0) = [0.1, 0.3]
- ItemB's initial embedding (qi_0) = [0.4, 0.6]

Step 2: Pairwise Pooling Layer
We will use pairwise pooling to capture correlations. The formula for pairwise pooling is:
pu = pu * pu
For instance, for User1: pu = [0.2, 0.5] * [0.2, 0.5] = [0.04, 0.25]
Similarly, we calculate pairwise pooling for all users and items.

You can recommend items based on the highest predicted scores for each user. In a real-world scenario, the model would be trained on a larger dataset, and the recommendations would be more meaningful.

This example illustrates the steps of NSCR with simplified calculations. In practice, the model parameters would be learned through training on real data,

Step 3: Hidden Layers

In this simplified example, let us assume we have one hidden layer with weights and biases:

Weight matrix W1 (for hidden layer) = [[0.1, 0.2], [0.3, 0.2]]

Bias vector b1 (for hidden layer) = [0.1, 0.2]

We will use the ReLU activation function.

We calculate the output of the hidden layer (e1) for each user-item pair:

For User1 and ItemA: e1(User1, ItemA) = ReLU(W1 * pu * qi + b1) e1(User1, ItemA) = ReLU([[0.1, 0.2], [0.3, 0.2]] * [0.04, 0.25] + [0.1, 0.2]) e1(User1, ItemA) = ReLU([0.022, 0.055] + [0.1, 0.2]) e1(User1, ItemA) = ReLU([0.122, 0.255])

Similarly, we calculate e1 for all user-item pairs.

Step 4: Prediction Layer

Let us assume we have a weight vector w (for prediction layer) = [0.4, 0.3].

We calculate the predicted scores for each user-item pair:

For User1 and ItemA: y^(User1, ItemA) = w * e1(User1, ItemA) y^(User1, ItemA) = [0.4, 0.3] * [0.122, 0.255] y^(User1, ItemA) = 0.4 * 0.122 + 0.3 * 0.255 y^(User1, ItemA) = 0.0488 + 0.0765 y^(User1, ItemA) = 0.1253

Similarly, we calculate predictions for all user-item pairs.

Step 5: Recommendations

Now that we have the predicted scores for all user-item pairs, we can recommend items to social users. Let us focus on User1, User2, and User3:

```
Recommendations for User1:
Sort the items by predicted scores for User1.
Recommend the top items.

Recommendations for User2:
Sort the items by predicted scores for User2.
Recommend the top items.

Recommendations for User3:
Sort the items by predicted scores for User3.
Recommend the top items.
```

You can recommend items based on the highest predicted scores for each user. In a real-world scenario, the model would be trained on a larger dataset, and the recommendations would be more meaningful.

2.4.3 Gao, C., Lin, T. -H., Li, N., Jin, D., & Li, Y. (2023). Cross-Platform Item Recommendation for Online Social E-Commerce.

This is the top closest existing system to the proposed system in this thesis as the two approaches both attempt to solve the cross site cold start problem. However, GaoLin's system faces the issue of applying their proposed solution to domains who do not share item embeddings.

In this research, the authors tackled the challenge of cross-platform recommendation in the realm of social e-commerce. This involves the task of suggesting products to users while they engage in shopping activities through social media platforms. To begin, they conducted a comprehensive examination of the distinctive shopping behaviours exhibited on traditional e-commerce applications compared to those on social media. Drawing insights from this analysis, they introduce a novel recommendation framework named CROSS (Cross-platform Recommendation for Online Shopping in social media). This framework leverages not only user-item interaction data gathered from both e-commerce and social media platforms but also incorporates social relationship data from the realm of social media.

Additionally, they present two distinct variants of CROSS, namely CROSS-MF and CROSS-NCF. To evaluate the effectiveness of our approach, they conduct extensive experiments utilizing real-world social e-commerce datasets. The results from these experiments conclusively demonstrate that our CROSS framework significantly outperforms existing state-of-the-art methods in the domain of cross-platform product recommendations.

Let us run through an example of how this system works with a sample dataset.

GaoLinRec23	Dataset
Users	[1, 2, 3, 4, 5]
Items	[10, 20, 30, 40, 50]
Platforms	A, B

Table 2.32 Sample Dataset Summary (GaoLinRec23)

user_id	platform	item_id	rating	friend_ids
1	A	10	5	2, 3
2	A	20	4	1, 3
3	A	30	3	1, 2
4	B	40	2	5
5	B	50	1	4

Table 2.33 Sample Dataset for GaoLinRec23

Step 1: Create User-Item Matrices

In this step, user-item interaction data is organized into matrices for each platform, representing users' interactions with items in their respective domains.

Platform A Matrix (R(A)):

Rows: User IDs

Columns: Item IDs

Entries: Ratings or interactions with items (e.g., ratings or clicks)

User-Item Matrices: It shows examples of user-item interaction matrices for two platforms, R(A) and R(B), where the entries represent user ratings or interactions with items.

Latent User and Item Matrices: It introduces the concept of latent user and item matrices (P and Q) for both platforms. These matrices are used in matrix factorization techniques to capture hidden patterns in user-item interactions.

Initialization: It mentions that these latent matrices are initialized with random values, and their dimensions are specified (e.g., P(A) is 3xK for platform A).

We start by organizing user-item interaction data into matrices for each platform (A and B). These matrices represent users' interactions with items in their respective domains.

Platform A has a matrix (R(A)) representing user interactions with items {item_10, item_20, item_30}.

Platform B has a matrix (R(B)) representing interactions with items {item_40, item_50}.

Entries in the matrices show the level of interaction (e.g., ratings or clicks).

- **Platform R(A):** [user_id, item_10, item_20, item_30]

  [5, 0, 0]

  [0, 4, 0]

  [0, 0, 3]

- **Platform R(B):** [user_id, item_40, item_50]

  [2, 0]

  [0, 1]

This matrix shows the level of interaction for each user with the items on Platform A

Step 2: Collective Matrix Factorization (CMF)

- **Explanation**: CMF is a technique used to jointly factorize the user-item interaction matrices (R(A) and R(B)). This process helps us uncover latent user and item features shared across both platforms.

- **Example**: We create latent user matrices (P(A) and P(B)) and latent item matrices (Q(A) and Q(B)). Let us assume we use 2 latent features (K = 2) for simplicity.
    - **Platform A (CMF)**:
        - Latent User Matrix (P(A)):
          Each row of P(A) represents a user, and each column represents a latent feature.
          Entries are initialized with random values.
          [[0.1, 0.2], [0.3, 0.4], [0.5, 0.6]]
        - Latent Item Matrix (Q(A)): [[0.7, 0.8, 0.9], [0.1, 0.2, 0.3]]
          Each row of Q(A) represents a latent feature, and each column represents an item.
          Entries are initialized with random values.
    - **Platform B (CMF)**:
        - Latent User Matrix (P(B)): [[0.4, 0.5], [0.6, 0.7]]
        - Latent Item Matrix (Q(B)): [[0.2, 0.3], [0.8, 0.9]]
- These matrices capture hidden patterns and preferences among users and items in both platforms.
-

The idea is to learn these latent matrices (P(A), Q(A), P(B), Q(B)) such that their multiplication approximates the original user-item interaction matrices (R(A) and R(B)). The matrices are learned through an optimization process that minimizes the difference between the actual ratings in the user-item matrices and the predicted ratings obtained by multiplying the latent matrices.

Step 3: SocialMF (Matrix Factorization with Social Regularization)

In Step 3, the focus is on the SocialMF algorithm, which is used for one of the platforms (Platform B in this case). The goal of SocialMF is to factorize the user-item interaction matrix R(B) for Platform B while incorporating social regularization. Here are the key points:

- **Algorithm Used**: SocialMF.

- **Objective**: Factorize R(B) into two latent matrices, P(B) and Q(B), where P(B) represents user embeddings, and Q(B) represents item embeddings for Platform B. The objective is to minimize the prediction error for observed ratings on Platform B.
- **Regularization Term**: SocialMF adds a social regularization term to the loss function. This term encourages the learned user embeddings (P(B)) to be influenced by the user's friends' embeddings.
- **Step 3: Social Matrix Factorization (SocialMF)**
    - **Explanation**: In SocialMF, we introduce a social regularization term into the loss function. This modification allows us to calculate predicted ratings for items on Platform B while considering the influence of social connections.
    -

For Platform B, we have latent user matrix (P(B)) and latent item matrix (Q(B)), each with dimensions (2 x K) and (2 x N_B), respectively.
We calculate predicted ratings (R^(B)ui) for items on Platform B using element-wise multiplication between P(B)u and Q(B)i.
For instance, to find R^(B)4,40 (the predicted rating for User 4 and Item 40 on Platform B), we compute (0.4 * 0.2) + (0.5 * 0.3) = 0.08 + 0.15 = 0.23.

Step 4: Loss Function
The result of step 2 and 3 is the learned latent matrices (P(A), Q(A), P(B), and Q(B)) that will be used for making cross-platform item recommendations. In this step, we define and compute the loss functions for both Platform A and Platform B, which will be used to train the recommendation models and optimize the latent matrices. The loss functions are used to quantify the error between the predicted ratings (Rhat) and the actual ratings (R) from the dataset.

For Platform A (Loss):
- **Loss Function (L(A))**: The loss function L(A) for Platform A is used to measure the error between the predicted ratings (Rhat(A)) and the actual ratings (R(A)) for

Platform A. It quantifies how well the recommendation model captures user-item interactions on Platform A.

- **Prediction (Rhat(A))**: Rhat(A) represents the predicted ratings for user-item pairs on Platform A, and it is calculated based on the learned latent matrices P(A) and Q(A).
- **Calculations**: To calculate L(A), you typically use a loss function that measures the difference between R(A) (actual ratings) and Rhat(A) (predicted ratings) for the observed user-item interactions on Platform A. A common choice is the Mean Squared Error (MSE) loss, which is computed as the sum of squared differences between actual and predicted ratings, divided by the number of observations.
- **Mathematically:**
- L(A) = Σ(u, i) ∈ Observed_A (R(A)ui - Rhat(A)ui)^2 / N_observed_A
- Where:
    - (u, i) represents a user-item pair on Platform A.
    - Observed_A denotes the set of observed user-item interactions on Platform A.
    - N_observed_A is the total number of observed interactions on Platform A.

For Platform B (Loss):

- **Loss Function (L(B))**: Like Platform A, the loss function L(B) for Platform B measures the error between the predicted ratings (Rhat(B)) and the actual ratings (R(B)) for Platform B. It quantifies how well the recommendation model captures user-item interactions on Platform B.
- **Prediction (Rhat(B))**: Rhat(B) represents the predicted ratings for user-item pairs on Platform B, and it is calculated based on the learned latent matrices P(B) and Q(B).
- **Calculations**: Like Platform A, you use a loss function (e.g., MSE) to calculate L(B) by comparing R(B) and Rhat(B) for the observed user-item interactions on Platform B.

- **Mathematically:**
- L(B) = Σ(u, i) ∈ Observed_B (R(B)ui - Rhat(B)ui)^2 / N_observed_B

- Where:
    - (u, i) represents a user-item pair on Platform B.
    - Observed_B denotes the set of observed user-item interactions on Platform B.
    - N_observed_B is the total number of observed interactions on Platform B.

These loss functions L(A) and L(B) quantify how well the recommendation models are performing on each platform by measuring the squared differences between actual and predicted ratings. The goal of the optimization process in Step 6 is to minimize these loss functions, effectively improving the accuracy of the recommendations on both platforms.

- Rhat(A)(1, 10) = P(A)1 * Q(A)10 = (1.2 + 2.4) = 3.6
- Rhat(A)(2, 20) = P(A)2 * Q(A)20 = (0.8 + 1.6) = 2.4
- Rhat(A)(3, 30) = P(A)3 * Q(A)30 = (0.6 + 0.6) = 1.2
-

We will perform similar calculations for Platform B.

Given that Observed_B contains the following observed interaction:

- (4, 40), Rating = 2

And we have already calculated Rhat(B)(4, 40) based on the matrices P(B) and Q(B):

- Rhat(B)(4, 40) = P(B)4 * Q(B)40 = (2.2) = 2.2

Now, we calculate the squared difference:

(R(B)4, 40 - Rhat(B)(4, 40))^2 = (2 - 2.2)^2 = 0.04

Now, L(B) is the sum of squared differences:

L(B) = 0.04

Since there is only one observed interaction on Platform B, N_observed_B = 1. Therefore, L(B) = 0.04

These calculated loss values represent the error between the actual ratings and predicted ratings for both Platform A and Platform B

Step 7: Recommendation

Once the model is trained, you can make item recommendations for both platforms.

For Platform A:

We have previously calculated the predicted ratings Rhat(A) based on the optimized matrices P(A) and Q(A) for Platform A. To recommend items to a user on Platform A, we follow these steps:

1. Get the user's rating history for Platform A:
    1. User 1: Item 10 (Rating = 5)
    2. User 2: Item 20 (Rating = 4)
    3. User 3: Item 30 (Rating = 3)
2. Calculate the predicted ratings for all items on Platform A using Rhat(A).
3. For example, for User 1:
    1. Rhat(A)(1, 10) = 3.6
    2. Rhat(A)(1, 20) = ... (Calculate for all items)
4. Sort the items by their predicted ratings in descending order.
5. For User 1:
    1. Predicted Ratings: {Item 10: 3.6, Item 20: ..., Item 30: ...}
6. Recommend the top-rated items to the user. For example, you can recommend the top 3 items with the highest predicted ratings.
    1. Recommended Items for User 1: {Item 10, Item 20, Item 30}

Repeat these steps for each user on Platform A to generate personalized recommendations.

For Platform B:

Similarly, for Platform B, we have calculated the predicted ratings Rhat(B) based on the optimized matrices P(B) and Q(B). To recommend items to a user on Platform B, follow similar steps:

1. Get the user's rating history for Platform B:
    1. User 4: Item 40 (Rating = 2)
2. Calculate the predicted ratings for all items on Platform B using Rhat(B).
3. For example, for User 4:
    1. Rhat(B) (4, 40) = 2.2
    2. Rhat(B) (4, 50) = ... (Calculate for all items)

4. Sort the items by their predicted ratings in descending order.
5. For User 4:
    1. Predicted Ratings: {Item 40: 2.2, Item 50: ...}
6. Recommend the top-rated items to the user, e.g., the top 3 items with the highest predicted ratings.
    1. Recommended Items for User 4: {Item 40, Item ..., Item ...}

Repeat these steps for each user on Platform B to generate personalized recommendations.

The final recommended items for each user on their respective platforms are based on the highest predicted ratings, reflecting their interests and behaviours. These recommendations are personalized for each user, making them more likely to engage with and purchase items.

2.5 RS Connecting Social Media to Ecommerce

A couple of systems showcase how the two domains: Social Media and Ecommerce can connect and share data across both domains. These systems **(Zhang & Pennacchiotti, 2015)** and **(Zhao et al., 2016)** are discussed in this section.

2.5.1 Zhang, Y., & Pennacchiotti, M. (2015). Recommending branded products from social media.

The main takeaway from this research is "Facebook connect". They discuss a set of users that' sign on to the ecommerce platform with their social media account on Facebook. Zhang and Pennacchiotti paper explore the untapped potential of user social media profiles, particularly on Facebook, to enhance e-commerce experiences. In the evolving landscape of e-commerce, where platforms like eBay seek increased user interaction, the authors propose leveraging social connections to personalize recommendations. Focusing on the correlation between brands liked on Facebook and those purchased on eBay, they introduce a brand prediction system. This system pioneers the application of unsupervised algorithms to harness social media information for personalized brand recommendations.

Zhang and Pennacchiotti establish a crucial connection between social media and e-commerce domains, particularly focusing on the integration of Facebook and eBay. In the contemporary landscape of e-commerce, major platforms like eBay are increasingly embracing social commerce, encouraging users to link their accounts with social media platforms. This integration is strategically designed to enhance user engagement and adoption on social media. Through dedicated API services, users can seamlessly connect their social media profiles, sharing liked pages and basic social information with the e-commerce platform. T

Let us walk through the proposed solution.

Step 1 : Logistic Regression Module:

This module predicts the probability of a user purchasing from a specific eBay meta-category based on Facebook likes and demographics. Using a hypothetical scenario, let us assume a user's likes and demographics result in the following probabilities:

User: Likes - [Nike, Adidas], Demographics - Female

Probabilities: Sporting Goods - 0.7, Home & Garden - 0.2

Step 2 : Brand Selection Module:

1. Baseline Selection (B1): Selects the most popular brands based on user purchases. Assuming the top brands are [Nike, Adidas, Puma]: Recommended Brands: [Nike, Adidas, Puma]
2. Baseline Selection (B2): Recommends brands liked by the user. If the user liked [Nike, Gucci, Apple]: Recommended Brands: [Nike, Gucci, Apple]
3. KNN Selection (Lknn): Applies K Nearest Neighbours for brand selection based on liked brands. If the top-k neighbours like [Nike, Adidas, Puma]: Recommended Brands: [Nike, Adidas, Puma]

4. Improving Recommendation with Related Brands: Expands the recommendation set with related brands. If related brands for Nike are [Jordan, Converse, Reebok]: Expanded Recommended Brands: [Nike, Adidas, Puma, Jordan, Converse, Reebok]

2.5.2 Zhao, W. X., Li, S., He, Y., Chang, E. Y., Wen, J. -R., & Li, X. (2016). Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information

The paper titled "Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information" by W. X. Zhao, S. Li, Y. He, E. Y. Chang, J. -R. Wen, and X. Li (2016) explores the intersection of social media and e-commerce, addressing the challenge of recommending products to users on social networking sites in "cold-start" situations, where users lack historical purchase records. The authors propose a solution leveraging linked users across social networking and e-commerce sites. They introduce a novel method involving recurrent neural networks to learn user and product embeddings from e-commerce data.

A modified gradient boosting trees approach is then employed to transform users' social networking features into embeddings. The paper details the steps involved, including microblogging feature extraction, distributed representation learning, and heterogeneous representation mapping. The proposed framework is validated on a dataset from SINA WEIBO and JINGDONG, demonstrating its effectiveness for cross-site cold-start product recommendation.

The authors highlight the increasing convergence of e-commerce and social networking. Platforms like eBay exhibit social network characteristics, and mechanisms such as social login and direct product purchasing from social media have become prevalent. The paper addresses the challenge of leveraging social networking information, particularly for users without historical purchase records on e-commerce sites. The proposed solution connects users' social networking features to a feature representation for product recommendation, using linked users across platforms as a bridge.

Let us walkthrough their solution.

Problem Formulation: Define sets U (users), P (products), and the purchase record matrix R. Introduce UL as linked users with microblogging attributes. Formulate the cross-site cold-start recommendation problem.

Sample Data:
U = {u1, u2, u3}
P = {p1, p2, p3}
R matrix with purchase records.

Step 1 : Microblogging Feature Extraction:
Identify demographic, text, network, and temporal attributes for microblogging users.
Extract topic distributions, word embeddings, latent group preferences, and temporal activity distributions.
Demographic attributes for u1.
Text attributes (topic distributions, word embeddings).
Network attributes (latent group preferences).
Temporal attributes (daily and weekly activity distributions).

Step 2 : Distributed Representation Learning:
Use recurrent neural networks to generate distributed feature representations (user embeddings) from e-commerce data.
Learn product embeddings using the Skip-gram model.
User embedding vu for u1.
Product embedding vp for p1.

Step 3: Heterogeneous Representation Mapping:
Use gradient boosting regression trees (MART) to map microblogging features to user embeddings.
Include attribute-level importance sampling.

Training set {au, vu} for linked users.

Apply MART for mapping.

Step 4: Applying Transformed Features to Recommendation:

Incorporate {au, vu} into the SVDFeature framework for product recommendation.

Code users, products, microblogging attributes, user embeddings, and product embeddings.

Formulate SVDFeature framework equations.

Code user and product features.

2.6 Data Mining Algorithms

In this section we discuss the main data mining algorithms: Association Rule Mining, Sequential Pattern Mining, Classification, and Clustering.

2.6.1 Association Rule Mining

In data mining, association is useful for analysing and predicting customer behaviour. (Agarwal & Srikant, 1994) play an important part in shopping basket data analysis. Association rule mining is primarily focused on finding frequent co-occurring associations among a collection of items. It is sometimes referred to as "Market Basket Analysis" since that was the original application area of association mining. Apriori **(Agrawal & Srikant, 1996)**, is an algorithm for frequent item set mining and association rule learning over transactional databases. An association rule expression is of the form $X \Rightarrow Y$, where "$\Rightarrow$" is intended to give a direction to the nature of correlation between the set of items X and Y.

Support(s): The support of an itemset $X \subseteq I$ is the fraction of transactions in (T) that contain both X and Y. The support count of an Itemset in a transaction database can be calculated as the number of transactions of the database that contain the itemset.

Support (itemset) = (number of tuples in the itemset)/(total number of tuples in the database)

Confidence(c): The confidence is a measure how often does items in Y appear in transactions that contain X. The confidence of the rule X ⇒ Y is the conditional probability that a transaction in T contains Y, given that it also contains X.

Confidence (X -> Y) = (Support (X U Y))/(Support (A))

Problem: Consider for the given transactions, let say T= {T1, T2, T3, T4} given in Table 1.7, some items are bought in all these transactions, where candidate set (C1) = {A, B, C, D} using association rule mining (Apriori algorithm), we can find the set of frequent patterns from large itemsets (Li) iteratively by computing the support of each itemset in the candidate set Ci.

Transaction ID	**Items**
T1	A,B,C,D
T2	A,B,D
T3	A,B
T4	B,C,D
T5	B,C
T6	C,D
T7	B,D

Table 2.34 Sample Transactional Data for Association Rule Mining

Input: Transaction database with transaction id and items purchased as given in Table 2.34 and minimum support =2.

Output: Frequent pattern items

Step 1: Find frequent item (L1) from candidate set (C1).

The principal step in Apriori process is to find frequent item by the counting occurrence of each item. The items that don't satisfy the minimum support count are pruned and produced frequent item (L1). In our case, frequent item (L1) = {A:3, B:6, C:4, D:5}.

Step 2: Generate candidate set (C2) from frequent item (L1) by Apriori join (L1 App-join L1).

We can generate a candidate set (C2) by L1 App-join L1. Frequent item (L1) can be joined only with an item that comes after it in frequent item (L1). Which will give candidate set (C2) = {AB, AC, AD, BC, BD, CD}.

Step 3: Find frequent item (L2) from candidate set (C2).

Frequent item (L2) is obtained by following the same procedure as in step 1. We can count the occurrence of each item in candidate set (C2), and infrequent items are removed to create frequent itemset (L2) = {AB: 3, BC: 3, BD: 4, CD: 3}.

Step 4: Generate candidate set (C3) from frequent item (L2) by Apriori join (L2 App-join L2).

We can apply the same process as in step 2 to generate candidate set (C3) by joining L2 with L2 using Apriori join and it produces candidate set (C3) = {ABC, ABD, BCD}.

Step 5: Find frequent item (L3) from candidate set (C3).

None of the item in candidate set (C3) satisfied minimum support. So, we need to stop here and join frequent item to get the final frequent item (L) =L1 U L2= {A, B, C, D, AB, BC, BD, CD}.

2.6.2 Classification

Classification is a data mining function that assigns items in a collection to target Categories or classes. The objective of classification is to accurately predict the target class for each record in the data. For example, a classification model used to identify loan applicants as low, medium, or high credit risks **(Kotsiantis, Zaharakis & Pintelas, 2007).** The classification using decision tree induction **(Apté & Weiss, 1997)** is one of the most widely used classification technique. The decision tree has two types of nodes, decision node (internal nodes) and a leaf node. A decision node specifies test (asks a question) on a single attribute. A leaf node indicates a class.

2.6.3 Sequential Pattern Mining

A sequence occurring in an ordered list of events with respect to time are called the Sequential Pattern (Agrawal & Srikant, 1996). A sequential Pattern is generally enclosed within the angular brackets (< >), and each itemset contains sets of items separated by commas (,). For example, a sequential pattern in an e-commerce system such as < (Bread, Milk), (Bread, Milk, Sugar), (Milk), (Tea, Sugar)> means customer has bought(Bread, Milk)together in his first purchase transaction, (Bread, Milk and Sugar) in second purchase, Milk alone in third purchase and (Tea and Sugar) together in the fourth purchase.

A sequential pattern comprising of n-itemsets is called an n-event sequence. An item cannot occur twice in an event (itemset) but can be multiple times in different events (itemsets) within the same sequential pattern. Thus, the number of instances of items in a sequence is called the length of a sequence. For example, < (Bread, Milk), (Bread, Milk, Sugar), (Milk), (Tea, Sugar)> is 4-events sequence with length 8.

Sequence database is composed of a collection of sequences {s1, s2, ..., sn} that are arranged with respect to time (Han, Pei & Kamber, 2011). A sequence database can be represented as a tuple <SID, sequence-item sets>, where SID: represents the sequence identifier and sequence-item sets specifies the sets in item enclosed in parenthesis (). Let us consider a very common example of a grocery store as shown in Table 2.35, which contains <CustomerID, Purchased Item, Timestamp>.

CustomerID	**Purchased Item**	**Timestamp**
01	Bread, Milk	13, Dec 2018 00:48:44
01	Bread, Milk, Sugar	19, Dec 2018 09:48:44
02	Bread	14, Dec 2018 1:48:44
01	Milk	21, Dec 2018 00:48:44
02	Bread, Milk, Sugar	18, Dec 2018 10:48:44

Table 2.35 Ecommerce Historic Data for Sequential Pattern Mining

The above is a sequential database created from the historical data, which could be interpreted as in Table 2.36 where SID represents the Sequence Identifier.

SID	**Sequences**

01	< (Bread, Milk), (Bread, Milk, Sugar), (Milk)>
02	< (Bread), (Bread, Milk, Sugar)>

Table 2.36 Sequential Database for Sequential Pattern Mining

The above Table 2.36 is the sequential database created from the historic data, the SID (01) has the purchase sequences for the customer (01) such as < (Bread, Milk), (Bread, Milk, Sugar), (Milk)>. In the first purchase, he bought (Bread, Milk), then (Bread, Milk, Sugar) and finally (Milk).

2.6.4 Clustering

Clustering is a process grouping several similar objects together **(Jain & Dubes, 1998),** clustering is unsupervised data mining technique, which does not need to be labelled manually and can automatically divide the data into set or group of clusters of similar objects. The K-means clustering **(Hartigan & Wong, 1979)** is one of the used clustering approaches in the field of data mining **(Steinbach, Karypis, & Kumar, 2000).** K-means clustering is used, when we have unlabelled data which cannot be defined into categories or groups. The K-means algorithm works iteratively to assign each data point to one of K groups based on the features that are given.

Consider the Table 2.37 as Input data with Height and Weight, the two important attributes. Using the K-means algorithm for clustering, we aim to find the possible clusters using the Table 2.37.

Height (cm)	Weight (kg)
185	72
170	56
168	60
179	68
182	72
188	77

Table 2.37 Sample Weight and Height Data for Clustering Data Mining Method

Solution: The K-means clustering algorithm consists of five major steps:

Input: a set of objects O = {I1, I2, I3, … In}and each object has n-dimensional attributes Oi such as Height and Weight, 1<= i <=n.
Output: subsets of objects such as [{O1, O4}; {O2, O6, O3} …].

Step 1: Randomly pick centroid from available objects.
Initialize cluster centroid. Let us consider, two centroids one containing minimum value of Height, Weight and another containing maximum value of Height, Weight as given in Table 2.38 and name then H1 and W1.

Cluster 1	185	72
Cluster 2	170	56

Table 2.38 Maximum and Minimum Clusters

Step 2: Calculate the distance between the centroid and other objects.
The distance can be calculated using the Euclidean distance formula.
E. D=√(〖(AH-H1)〗 ^2+ 〖(Aw-W1)〗 ^2)

Where, XH= Observation value of height, H1= Centroid value of cluster 1 for height, Xw = Observation value of height, W1= Centroid value of cluster 1 for weight. Here, we are using (Height: 168, Weight: 60) as object value from input data.
Euclidian Distance from Cluster 1 = √(168−185)2+60−722= 20.808
Euclidian Distance from Cluster 2 = √(168−185)2+(60−72)2 = 4.472
From Euclidean distance, we can see that record with (168, 60) is very close to cluster 2.

Step 3: Update centroid of each new cluster, by computing the average attributes of all objects in a cluster.
Updated Centroid

	Height	Weight
Cluster 1	185	72
Cluster 2	(170+168)/2=169	(56+60)/2=58

Table 2.39 Updated Centroids for Clusters

Step 4: Repeat step 1, 2 and step 3 until the centroids stop changing.

The output created in our example is present below.

Cluster 1 = {(185,72), (179,68), (182,72), (188,77)}

Cluster 2 = {(170,56), (168,60)}

Cluster created by K-means method.

Step 5: Return the k clusters.

In this case, clusters returned are {(185,72), (179,68), (182,72), (188,77)} and {(170,56), (168,60)}.

2.7 Comparison of Existing Systems

This section presents tables discussing each existing systems challenges in comparison with each other emphasizing the gap we want to fill in this thesis.

2.7.1 Comparison of Data Mining Algorithms

Author	Algorithm	Idea	Challenges

Agarwal & Srikant (1994, 1996)	Apriori	Apriori algorithm focuses on finding frequent item sets and association rules. It iteratively identifies and prunes itemsets based on minimum support.	Challenges include Scalability with large datasets, multiple scans of data, and handling high-dimensional data.
Kotsiantis, Zaharakis & Pintelas (2007)	Classification	Classification assigns items to target categories based on attributes.	Challenges include decision tree construction, handling missing values, overfitting, and interpretability of complex trees.
Agrawal & Srikant (1996)	Sequential Pattern Mining	Identifying sequential patterns in ordered lists of events over time	Challenges include handling large sequential databases, defining meaningful event sequences, and efficiently discovering sequential pattern
Jain & Dubes (1998)	Clustering	Clustering groups similar objects together in an unsupervised manner.	Challenges include determining the optimal number of clusters (K), handling outliers, and sensitivity to initial centroids

Table 2.40 Comparison of Data Mining Algorithms

2.7.2 Comparison of Ecommerce Recommender Systems

Author	Title	Idea	Challenges

Zhang, Hu, Li, & Chen (2009)	A Hybrid Recommender System for Large-Scale E-commerce	Combining matrix factorization and k-nearest neighbours to improve recommendation accuracy for sparse user-item matrices.	Sensitivity to hyperparameter tuning, limited interpretability due to hybrid nature, and potential scalability issues with large-scale data.
Jiabei Dai and Bin Zeng (2016)	An Association Rule Algorithm for Online E-Commerce Recommendation Service	Introducing the profit-support association rule algorithm to incorporate item profitability into the recommendation process based on sales volume.	Potential complexity in defining an optimal profit threshold, potential biases in profit-oriented rules, and the need for continuous adjustment to changing market conditions.
Liu, Lai, and Lee (2011)	A Hybrid of Sequential Rules and Collaborative Filtering for Product Recommendation	Addressing the limitations of sequential rule methods by incorporating RFM segmentation, transaction matrix clustering, and association rule mining.	Potential noise in RFM clustering, sensitivity to parameter choices in clustering, and challenges in capturing dynamic user behaviours effectively.
JianZhong Yang, Huirong Chen, Shengnan Xiong, Zhongqiang Yang, Yu Jiang (2020)	Custom Data Mining Association Rules of Nixing Pottery Product Recommendation System	Implementing a recommendation system for Nixing Pottery products using custom data mining association rules based on user interests and buying behaviours.	Potential biases from user registration data, challenges in handling real-time user interactions, and the need for continuous updates to adapt to evolving user preferences.
Bhatta, Ezeife, and Butt (2019)	Mining the Sequential Pattern Based on Daily Purchase History for Collaborative Filtering	Extracting frequent sequential patterns from e-commerce historical data to enhance a user-item rating matrix using the HSPRec19 and SHOD algorithms.	Inefficiency in pattern extraction may result in identification of less relevant patterns, impacting matrix enhancement.

Table 2.41 Comparison of Ecommerce RS

2.7.3 Comparison of Recommender Systems Addressing Cold Start Problem

Author	Title	Idea	Challenges
J. Lin, K. Sugiyama, M. Kan, T. Chua (2013)	Addressing Cold Start in App Recommendation (LinRec2013)	The paper addresses the cold-start problem in app recommendations by constructing latent user models from Twitter followers	Assumes that Twitter followers' preferences accurately reflect a user's app preferences.
Y. Seroussi, F. Bohnert, I. Zukerman (2011)	Personalised Rating Prediction for New Users Using Latent Factor Models (SeroussiRec11)	The paper addresses the challenge of inaccurate rating predictions for new users by extending the matrix factorization (MF) algorithm	Effectiveness relies on accurate user attributes and inference from user-generated texts

Table 2.42 Comparison of RS Addressing Cold Start Problem

2.7.4 Comparison of Transfer Learning Cross Domain Recommender Systems

Author	Title	Idea	Challenges
W. Pan, E. W. Xiang, N. N. Liu, Q. Yang (2010)	Transfer Learning in Collaborative Filtering for Sparsity Reduction (PanXiangRec10)	The paper addresses the data sparsity issue in collaborative filtering through Coordinate System Transfer (CST). It leverages auxiliary data from mature domains and applies sparse Singular Value Decomposition (SVD) for coordinate system adaptation, reducing the impact of data sparsity in the target domain	Effectiveness relies on the quality and relevance of auxiliary data.
Anwar and Uma (2022)	CD-SPM: Cross-domain book recommendation using sequential pattern mining and rule mining	The CD-SPM algorithm integrates data from movies and books to provide cross-domain recommendations in e-commerce. The input involves combining user preferences from both domains into an integrated dataset. Sequential pattern mining identifies frequent patterns, and collaborative filtering refines recommendations based on these patterns.	- The algorithm may struggle to provide accurate recommendations for new users who lack sufficient data, posing a challenge in addressing the needs of users with minimal or no historical information.

Table 2.43 Comparison of Transfer Learning CDRS

2.7.5 Comparison of Matrix Factorization Cross Domain Systems

Author	Title	Idea	Challenges
Gao (2023)	Cross-Platform Item Recommendation for Online Social E-Commerce	It integrates user-item interaction data from both e-commerce and social media platforms, employing Collective Matrix Factorization (CMF) to unveil latent features shared across platforms. Additionally, the algorithm utilizes Social Matrix Factorization (SocialMF) on one platform, incorporating social relationships to predict item ratings. The optimization phase minimizes loss functions, quantifying the error between predicted and actual ratings.	- Product item data is shared across domains so can't be applied to systems who do not share such data - Inaccurate or incomplete information about user connections could impact the performance of Social Matrix Factorization - When a user has no purchase history, the algorithm lacks personalized data to infer preferences and generate accurate suggestions.
Gao (2019)	Item Recommendation for Online Social E-Commerce	The Neural Social Collaborative Ranking (NSCR) algorithm addresses the challenge of cross-domain social recommendation by seamlessly integrating neural collaborative filtering (NCF) with graph regularization. oriented domains, bridging the gap between diverse online platforms. The output is a set of personalized recommendations for users, reflecting their preferences and behaviours based on learned correlations and interactions.	- Product item data is shared across domains so can't be applied to systems who do not share such data - Sparsity of user-item interactions between information-oriented and social-oriented domains Since collaborative filtering techniques, including graph regularization used in NSCR, rely on historical interactions to identify patterns and similarities, users with no history present a data gap
Wang (2021)	Leveraging Product Adopter Information from Online Reviews for Product Recommendation	The algorithm proposed in the paper utilizes a bootstrapping-based extraction method to identify adopter mentions from online product reviews, categorizing them into demographic groups for personalized recommendations.	- Users with no purchase history may pose challenges in adopting this algorithm as it heavily relies on adopter mentions in reviews. Lack of historical interaction data makes it difficult to categorize users into demographic groups, potentially leading to less personalized recommendations.

Table 2.44 Comparison of Joint Matrix Factorisation

CHAPTER 3: PROPOSED FACEBOOK-DATA CROSS DOMAIN RECOMMENDER

In this thesis, we propose a way to extract and transform user product interests from their facebook likes to make what products they may like to buy based on their facebook likes. The goal here is to discover user product interests from a like on post for example "I like cooking" for product recommendation on the ecommerce by proposing the Facebook Data Cross Domain (FD-CDRS'23) System.

As demonstrate in tables 1.2 and 1.3, the main challenge to discover what specific product a user might be interested in based on the social media action when that same product does not exist on the ecommerce platform. It is even more challenging to transform the user activity on social media into usable representations on the ecommerce platform.

For example, in a situation where we have products [A, B] on both social media and ecommerce platform, when a user interacts with product A on the social media platform, we can simply map that same product A on the ecommerce and recommend it for the user. In the situation where a user likes a comment that says, "I love football" and there are products on the ecommerce platform product [A , B] , there is no direct way to simply map that comment to a product. In this scenario, we are not sure if to recommend A or B as we are unable to tie the user's interest to specific product interests.

The proposed Facebook Data Cross Domain '2023 (FD-CDRS '23), which mines users' likes from Facebook and e-commerce historical purchases discovers patterns and association rules between user preferences across both domains. For instance, a rule can be "Users (user) who typically like cooking posts on Facebook, purchase cooking recipes (item) from Amazon". With such rules, the system can unearth associations of likes and purchases for product recommendations without having to share the items across the domains, as existing systems do.

Given historic e-commerce purchases, facebook likes and a set of users, the goal is to discover all prevalent associations between the new user like and purchases, sorting them based on confidence, and ultimately recommending items by prioritizing those associated with the highest-confidence rules. These associations are discovered by counting the frequency of likes and purchases combinations. It means something when a lot of users who liked a specific postA, also buy specific productB.

Once we can transform the activities on both platforms into itemsets and discover frequent occurrence of activities, we eliminate the need to have the same product on both platforms, which essentially solves the main challenge of this thesis. For example, if we discover that likes on postA always lead to productB, we can recommend productB to users who eventually like postA, without needing to directly have productB on the social media platform to model a representation of user's interests in productB.

The core of the task that FD-CDR solves includes:

1. Using data from a source domain to enrich user preferences in a target domain due to lack activity in the target domain.
2. Modelling user item preferences across domains without standardizing products across platforms, connecting databases or creating same item representations on both platforms.
3. Transforming user activity on social media in combination with historical purchase into user interests across both domains
4. Recommending specific items on ecommerce platforms rather than using complex machine learning algorithms to generate categories or assign classes to user activity behaviour on social media. For example, some other systems since do not share products may rather use similar categories across platforms, but will not always accurate and not get the actual specific item rather only the category of interest such as football, music, versus specific item such as Nike Air Force 1s 1st Gen.

The aim is to generate personalized recommendations for a new user, we would follow these steps by first.

1. Generating association rules by counting the frequency of combinations of historic purchases and likes given a minimum support and minimum confidence For instance {postA} => {productB}. Which essentially means a like on post A is typically followed by a purchase of productB. The goal here is to discover what two combinations of activities (likes, purchases) occur a lot. Which is, what liked posts followed by purchased products occur a lot in dataset.
2. Now given new user with no ecommerce history's like on postA, we find all the rules containing that liked postA. Which is, we find all the rules where countless actions of liked postA lead to a product purchase.
3. Calculate the confidence level for each of those rules. We calculate how often this rule occurs, as we believe user behaviour may be unpredictable. A user can randomly like a post, and a user can randomly like post which does not much communicate their interests in the posts. However, several users liking a specific post and liking a specific product countless time, may show interest.
4. Rank the rules by confidence level in descending order. The more confident in the rule, the more the rule occurs for a good range of users, the stronger the user interests.
5. Return the purchased items {productB} from the top-ranked rule as the recommended items for the new user.

.

3.1 Proposed FD-CDR'23 System

The FD-CDR'23 combines procedures of the MLTU (Mine User Per Likes) Algorithm, the modified Apriori algorithm to find rules of likes leading to purchases and the HARR (Hybrid Association Rule Recommender) Algorithm that predicts a product a user may like to buy given the new user's like based on the generated association rules.

Algorithm 3.1 : FD-CDR (Facebook Data Cross Domain Recommendation)

Input: minimum support (s), minimum confidence (c), historical facebook activity log (F), historical purchase database (PDB), facebook to ecommerce user mapping (FP), new user like (l)
Output: Top recommended items ranked by confidence level
Intermediates: Itemsets per User (IPDB), Association Rules (AR), Confidence level Rule Set (CRS), number of recommendations (n)

```
1  : Generate Itemsets per User, $u$ (IPDB) $\leftarrow$ MLTU(F,PDB,FP) as in Algorithm
     3.2 in section 3.1.1
2. : Generate Association rules (AR) $\leftarrow$ ModifiedApriori($s$, $c$, IPDB) as in
     Algorithm 3.3 in section 3.1.1
3  : for each rule $r$ in Association Rules (AR)
4  :     if $r$ antecedent contains new user like $l$
5  :             Confidence Rule Set (CRS) = calculate conf ($r$)
6. : end
7  : sort Confidence Rule Set (CRS) by descending order
8  : return Confidence Rule Set for $n$ recommendations (CRS)[:$n$] consequent
```

Algorithm 3.1 Facebook Data Cross Domain Recommendation (FD-CDR)

3.1.1 Steps in the proposed FD-CDR'23 Algorithm

Step 1: Generate all historic purchases per User.

Extract user ecommerce IDS and corresponding product IDS from sample Historical Purchase Database (PDB) present in **Table 3.1** using step 1 procedure in **Algorithm 3.2**

userID	transactionID	productID	timestamp
u1	1001	p1	2023-01-15T08:30:00
u2	1002	p2	2023-01-16T10:45:00
u1	1003	p3	2023-02-01T15:20:00
u3	1004	p4	2023-02-10T14:00:00
u2	1005	p1	2023-02-18T09:10:00
u4	1006	p5	2023-03-05T11:45:00
u1	1007	p2	2023-03-12T16:30:00
u3	1008	p3	2023-04-02T13:15:00
u1	1009	p4	2023-04-20T10:00:00
u2	1010	p5	2023-05-06T12:20:00
u1	1011	p6	2023-05-20T08:45:00
u2	1012	p3	2023-06-02T17:10:00
u3	1013	p5	2023-06-15T14:30:00
u4	1014	p2	2023-07-01T09:00:00

Table 3.1 Sample Historical Purchase Database (PDB) for FD-CDR

Following the procedure in step 1 of algorithm 3.2, we would follow these steps to retrieve our purchases by each user.

1. Create an empty dictionary data structure PDBU, to store user ecommerce IDs (userID) from Table 3.1 and associated product IDs for each user transaction in Table 3.1
2. For each entry in the original data structure PDB in Table 3.1, retrieve the userID and productID.
3. Check if the userID is not in the set of known users. If the user is new, initialize an empty list for that user in PDBU.
4. Append the productId to the list associated with the userId in PDBU.

Return Result: After processing all entries in PDB, return the resulting PDBU dictionary containing user ecommerce IDs and associated product IDs.

PDBU = extract_ecommerce_user_products(PDB)

```
# Output: = PDBU = {
  'u1': ['p1', 'p3', 'p2', 'p4', 'p6'],
  'u2': ['p2', 'p1', 'p5', 'p3'],
  'u3': ['p4', 'p3', 'p5'],
  'u4': ['p5', 'p2']
}
```

Step 2: Generate Ecommerce to Facebook userID mapping.

For each ecommerce user, retrieve their Facebook UserID from Facebook to ecommerce mapping (FP) in **Table 3.2** using step 2 procedure in **Algorithm 3.2**

userEcommerceID	userFacebookID
u1	1041851b-87db-4054-b56a-d1e4c280f39b
u2	0844682d-6e20-4f92-86e9-a0c60ee81eb2
u3	3b404f36-7ed0-44d2-9a57-e3980e433825
u4	0a175d58-bcac-40ec-b1e8-a89ae079f945
u5	1b2c3d4e-5f6g-7h8i-9j0k-a1b2c3d4e5f6

Table 3.2 Sample Facebook to Ecommerce User Mapping (FP)

Following the procedure in step 2 of **Algorithm 3.2**, we would have to follow these steps:

1. Create an empty dictionary FMU to store the mapping of Facebook IDs to ecommerce user IDs.
2. For each entry in the original data structure FP in Table 3.2, extract the userEcommerceID and userFacebookID.
3. Assign the userEcommerceID to the key userFacebookID in the FMU dictionary.
4. After processing all entries in FP, return the resulting FMU dictionary containing the mapping of Facebook IDs to ecommerce user IDs as shown below.

```
# Output: FMU = {
  '1041851b-87db-4054-b56a-d1e4c280f39b': 'u1',
  '0844682d-6e20-4f92-86e9-a0c60ee81eb2': 'u2',
  '3b404f36-7ed0-44d2-9a57-e3980e433825': 'u3',
  '0a175d58-bcac-40ec-b1e8-a89ae079f945': 'u4',
  '1b2c3d4e-5f6g-7h8i-9j0k-a1b2c3d4e5f6': 'u5'
}
```

Step 3: Generate liked posts per User.

Use the Facebook User IDs to extract liked posts from facebook activity log (F) present in **Figure 3.1** using procedure in step 3 of **Algorithm 3.2**

```
{
    "id": "123456789",
     "message": "Enjoying a great meal with @Friend1 and @Friend2!",
     "created_time": "2023-01-01T12:34:56+0000",
     "likes": {
         "data": [
             {
                "id": "1041851b-87db-4054-b56a-d1e4c280f39b",
                 "name": "User 1"
             },
             {
                "id": "0844682d-6e20-4f92-86e9-a0c60ee81eb2",
                "name": "User 2"
             }
         ]
     },
```

Figure 3.1 Sample Facebook Activity Log (F) for FD-CDR

Following the procedure in step 3 of Algorithm 3.2 we would have the following steps:

1. Create an empty dictionary FDL to store liked posts organized by ecommerce user IDs.
2. For each post in the data structure F shown in Figure 3.1, extract the userFacebookID from the likes data of the post.
3. Check if the userFacebookID is present in the mapping FMU.
4. If userFacebookID is found in FMU, retrieve the corresponding userEcommerceID.

5. Check if the userEcommerceID is not in FDL. If userEcommerceID is not in FDL, initialize an empty list for that user in FDL and append the post to the list associated with userEcommerceID in FDL.
6. Return Result: After processing all posts in F, return the resulting FDL dictionary containing liked posts organized by ecommerce user IDs a shown below.

```
# Output: FDL = {
  'u1': ['post1', 'post2'],
  'u2': ['post1', 'post3'],
  'u3': ['post2'],
  'u4': ['post1']
}
```

Step 4: Create Itemsets of activities (Likes and Purchases) per User

Create a transactional table with by combining outputs from step 3 (FDL) and step 1 (PDBU) as set of items per each user using **step 4** procedure from **Algorithm 3.2**

Following the procedure in step 4 of Algorithm 3.2 we would have the following steps:

1. Create an empty list IPDB to store transactional tables as itemsets per user.
2. For each userId and associated products in the dictionary PDBU items, retrieve the liked posts (likes) for the current userId from FDL, or an empty list if the userId is not present in FDL.
3. Combine the user's products and liked posts to create combinedItems.
4. Append a dictionary with keys 'UserID' and 'Items' to IPDB, where 'UserID' is the current userId and 'Items' is the combined list of products and likes.
5. Return Result: After processing all user-product combinations, return the resulting IPDB list containing transactional tables as itemsets per user as shown below

The resulting output (IPDB) which is the set of items per each user is shown below:

```
# Output: IPDB = transactional_data = [
  ['p1', 'p3', 'p2', 'p4', 'p6', 'post1', 'post2'],
  ['p2', 'p1', 'p5', 'p3', 'post1', 'post3'],
  ['p4', 'p3', 'p5', 'post2'],
  ['p5', 'p2', 'post1']
]
```

In the resulting outputs the datatypes are attached to each item in IPDB, so we can tell if their posts or products. Looking at their IDS alone would not tell us, and we also want to find frequent combinations of which liked posts lead to purchase products, to tackle the cold start on the ecommerce. It looks more like the output below in implementation:

```
transactional_data = [
  {'UserID': 'u1', 'Items': [{'p1Id': 'p1', 'Type': 'Product'}, {'post1Id': 
'post1', 'Type': 'Post'}, {'post2Id': 'post2', 'Type': 'Post'}]},
  {'UserID': 'u2', 'Items': [{'p2Id': 'p2', 'Type': 'Product'}, {'post3Id': 
'post3', 'Type': 'Post'}]},
  {'UserID': 'u3', 'Items': [{'p3Id': 'p3', 'Type': 'Product'}, {'post1Id': 
'post1', 'Type': 'Post'}, {'post2Id': 'post2', 'Type': 'Post'}, {'post4Id': 
'post4', 'Type': 'Post'}]},
  {'UserID': 'u4', 'Items': [{'p4Id': 'p4', 'Type': 'Product'}, {'post2Id': 
'post2', 'Type': 'Post'}]}
]
```

Let us look at the algorithm MTLU that combines all these steps to achieve the itemsets per user.

Algorithm 3.2 : MLTU (Mine Likes and Transactions Per User)

Input: historical facebook activity log (F), historical purchase database (PDB), facebook to ecommerce user mapping (FP)

Output: Itemsets per User (CRS)

Intermediates: Historical Purchase per User (PDBU), Facebook to E-commerce unique Identifier Mapping per User (FMU), facebook likes per Ecommerce user (FDL), Itemsets per User (IPDB)

```
1 : Extract user ecommerce IDS and product IDs from PDB, PDBU = {}
        For each in PDB:
            userId = each(userId), productId  = each(productId)
             if userId not in FP(users):
                  PDBU [userId] = [ ] # initialize empty list
             PDBU[userId] append productId
         return PDBU.
2  : For each ecommerce user, retrieve Facebook ID from FP, FMU = {}
          For each in FP:
               userEcommerceID = each(userEcommerceID)
               userFacebookID  = each(userFacebookID)
               FMU[userFacebookID] = userEcommerceID
            return FMU
3  : Use facebook ID to extract liked posts from F, FDL = {}
            For each post in F[data]:
               userFacebookID post['likes']['data']['user']
                   if  userFacebookID is in FMU
                         userEcommerceID = FMU[userFacebookID]
                              if userEcommerceID not in FDL :
                                 FDL[userEcommerceID] = [ ]
                              append post to FDL[userEcommerceID]
            return FDL
4   : Create transactional table as itemsets per user, IPDB = [ ]
         For userId, products in PDBU items:
             likes = FDL.get(userId, [ ])
             combinedItems = products + likes
                append ({'UserID': userId, 'Items': combinedItems}) to IPDB.
         return IPDB.
```

Algorithm 3.2 MLTU (Mine Likes and Transactions Per User Algorithm)

Step 5: Generate frequent itemsets and Association Rules

After retrieving itemsets per User (IPDB) we generate frequent itemsets and then association rules using our Modified Apriori Algorithm present in **Algorithm 3.3**

Step 5a : Initialize:

Generate 1-itemsets (C1) from unique items in the transactional data (IPDB) shown here:

```
# Output: IPDB = transactional_data = [
  ['p1', 'p3', 'p2', 'p4', 'p6', 'post1', 'post2'],
  ['p2', 'p1', 'p5', 'p3', 'post1', 'post3'],
  ['p4', 'p3', 'p5', 'post2'],
  ['p5', 'p2', 'post1']
]
```

# Calculate support for each 1-itemset.

C1 = [['p1'], ['p2'], ['p3'], ['p4'], ['p5'], ['p6'], ['post1'], ['post2'], ['post3'], ...]

# Calculate support for each 1-itemset

L1 = [['p1'], ['p2'], ['p3'], ['post1'], ['post2'], ...]

Step5b : Iterate (k = 2, 3, ...):

a. Join frequent (k-1)-itemsets to generate candidate k-itemsets (Ck).
b. Calculate support for each candidate k-itemset.
c. Prune candidate k-itemsets that don't meet the minimum support

k = 2

C2 = [['p1', 'p2'], ['p1', 'p3'], ['p1', 'post1'], ['p1', 'post2'], ['p2', 'p3'], ...]

Calculate support for each candidate 2-itemset.

L2 = [['p1', 'p2'], ['p1', 'post1'], ['p1', 'post2'], ['p2', 'post1'], ['p2', 'post3'], ...]

k = 3

C3 = [['p1', 'p2', 'post1'], ['p1', 'p2', 'post2'], ...]

Calculate support for each candidate 3-itemset.

L3 = [['p1', 'p2', 'post1'], ...]

Step 5c: Stop when no frequent k-itemsets can be generated.

Output:

Frequent itemsets (Lk) for k = 1, 2, 3, ...

```
frequent_itemsets = [
  [{'p1Id': 'p1', 'Type': 'Product'}, {'p2Id': 'p2', 'Type': 'Product'}],
  [{'p1Id': 'p1', 'Type': 'Product', 'post1Id': 'post1', 'Type': 'Post'}],
  [{'p1Id':  'p1',  'Type':  'Product',  'p2Id':  'p2',  'Type':  'Product',
'post1Id': 'post1', 'Type': 'Post'}]
]
```

Step 5d: Association Rule Generation

Input:

Frequent itemsets (Lk) from Procedure 3 in step 2

Minimum confidence (min_confidence): 0.6

Generate association rules from frequent itemsets and filter out rules where the potential antecedents are posts so we can have Posts => Products.

Association Rules: (L1 is not used for rules)

Rules from L2:

- ['p1'] => ['p2'], Confidence: 0.6
- ['p2'] => ['p1'], Confidence: 0.7
- ...

Rules from L3:

- ['p1', 'p2'] => ['post1'], Confidence: 0.9

-[post1] => [p2], confidence 0.9

- ...

Output:

Association rules with confidence for each rule.

```
association_rules (AR) = [
  {'Antecedent':  [{'p1Id':  'Post1',  'Type':  'Post'}],  'Consequent':
[{'p2Id': 'p2', 'Type': 'Product'}], 'Confidence': 0.9},]
```

What the rule generated above post1 => p2, tells us that most of the time users who liked post1 purchased p2 and thus, we can recommend p2 to a new user who likes post1 provided the confidence level is high

Here is the modified Apriori algorithm to achieve the step 5 above.

Algorithm 3.3 : Modified Apriori (Posts => Products)

Input: minimum support (s), minimum confidence (c), Itemsets per User (IPDB)
Output: Association Rules (AR)

```
1.      Initialize frequent_itemsets with frequent itemsets of size 1.
2.      Set k to 2.
3.      While frequent_itemsets[k-2] is not empty:
            a. Generate candidate_itemsets by joining frequent_itemsets[k-2].
            b. Prune candidate_itemsets that do not meet min_support.
            c. Add remaining candidate_itemsets to frequent_itemsets.
            d. Increment k
4.      Generate association_rules:
        a. For each frequent_itemset in frequent_itemsets:
            i. Generate all possible non-empty subsets of the itemset.
            ii. For each subset:
                1. If the subset contains only items of type "Post," consider it as a potential antecedent.
                2. If the remaining items in the itemset (excluding the subset) are of type "Product," consider them as potential consequents.
                3. Calculate confidence for the rule (antecedent implies consequent).
                4. If confidence exceeds the min_confidence threshold:
                    a. - Add the rule to the list of association_rules
5.      Return association_rules (AR)
```

Algorithm 3.3 Modified Apriori Algorithm

The highlighted red parts of Algorithm 3.3 indicate the modifications done to Apriori. The goal here is to solve the cross-site cold start problem where users may low activity on ecommerce. Thus, we want to generate rules where actions on social media lead to ecommerce purchases, such that when new users come in with no ecommerce history, we can leverage their social media history. What we did in Algorithm 3.3 is to consider the datatypes of the items in the itemsets: Posts or Products, so we can filter rules for where posts imply products.

Step 6: Generate Recommendations

Use HARR (Hybrid Association Rule Recommendation) Algorithm 3.3 to generate personalized recommendations based on association rules (AR) derived from step 5.

We can consider the new user like (l) to be post1.

Using the steps in Algorithm 3.3,

1. We first go through our association rules (AR) in step5d to retrieve all rules that contain post1 -[post1] => [p2],
2. We calculate the confidence level of this rule, to be 0.9.
3. There is only one rule with post1, sorting by confidence level we still have -[post1] => [p2], on top of the list in the ranking
4. Now we simply extract p2 as the item to be recommended to the user.

Algorithm 4 : HARR (Hybrid Association Rule Recommendation)

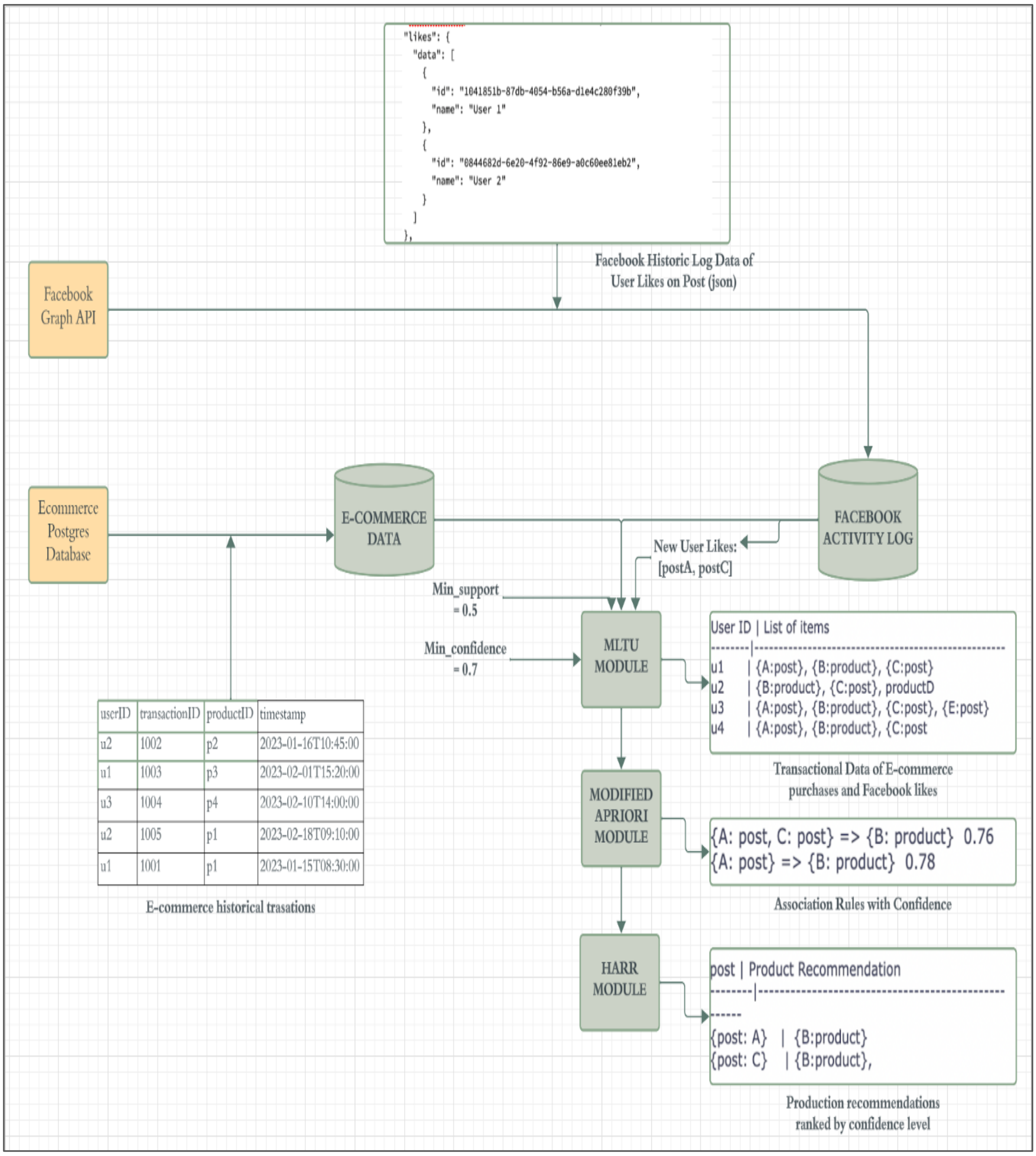
Input: Association rules (AR), new user like (l)
Output: Top recommended items ranked by confidence level
Intermediates Rule Set (CRS), number of recommendations (n)

```
1  : for each rule $r$ in (AR)
2  :     if $r$ antecedent contains l
3  :             (CRS) $\leftarrow$ calculate conf ($r$)
4. : end
5  : sort (CRS) by descending order
6  : return (CRS)[:$n$] consequent
```

Algorithm 3.4 HARR (Hybrid Association Rule Recommendation)

Let us look at the overall architecture of the system flow on how FD-CDR works.

3.2 Proposed FD-CDR’23 System Architecture



```
"likes": {
  "data": [
    {
      "id": "1041851b-87db-4054-b56a-d1e4c280f39b",
      "name": "User 1"
    },
    {
      "id": "0844682d-6e20-4f92-86e9-a0c60ee81eb2",
      "name": "User 2"
    }
  ]
},
```

userID	transactionID	productID	timestamp
u2	1002	p2	2023-01-16T10:45:00
u1	1003	p3	2023-02-01T15:20:00
u3	1004	p4	2023-02-10T14:00:00
u2	1005	p1	2023-02-18T09:10:00
u1	1001	p1	2023-01-15T08:30:00

```
User ID | List of items
--------|-----------------------------------------------
u1      | {A:post}, {B:product}, {C:post}
u2      | {B:product}, {C:post}, productD
u3      | {A:post}, {B:product}, {C:post}, {E:post}
u4      | {A:post}, {B:product}, {C:post
```

```
{A: post, C: post} => {B: product}  0.76
{A: post} => {B: product}  0.78
```

```
post | Product Recommendation
--------|---------------------------------------------
------
{post: A}   |  {B:product}
{post: C}    | {B:product},
```

Facebook Historic Log Data of User Likes on Post (json)
Facebook Graph API
Ecommerce Postgres Database
E-COMMERCE DATA
FACEBOOK ACTIVITY LOG
New User Likes: [postA, postC]
Min_support = 0.5
Min_confidence = 0.7
MLTU MODULE
MODIFIED APRIORI MODULE
HARR MODULE
E-commerce historical trasations
Transactional Data of E-commerce purchases and Facebook likes
Association Rules with Confidence
Production recommendations ranked by confidence level

Figure 3.2 FD-CDR'23 System Architecture

In **Figure 3.2,** of the system architecture, the ecommerce purchase history is extracted from the Postgres database and the Facebook activity log is extracted from Facebook Graph API into the MLTU module. THE MLTU module then prepares and transforms this data into itemsets of activities across both domains based on each user’s uniquely identified mapping. The modified Apriori module takes these itemsets to discover which combinations of purchases and likes frequently to occur to discover rules. Those rules are sent to the HARR module and once a new user likes a post, the item is recommended based on the generated rule.

3.3 System Implementation

The system was implemented using Python 3.10 as the primary programming language, leveraging its extensive libraries and community support. The backend database management was handled by PostgreSQL, providing a robust and scalable solution for data storage and retrieval. Facebook Graph API key integration facilitated seamless interaction with Facebook's social graph, enhancing the system's capabilities. The development environment was Visual Studio Code, a versatile and widely adopted code editor, ensuring efficient and streamlined coding processes. For testing and experimentation, Jupyter Notebook in Python 3 was employed, offering an interactive and collaborative platform. Essential libraries such as Pandas, NumPy, and Seaborn provided powerful data manipulation, numerical computing, and visualization capabilities. MATLAB was also utilized for specific tasks, contributing to the diversity of tools employed in the implementation.

- Implementation tools
    - Python 3.10 (Python Software Foundation, 2021)
    - Postgres Database (PostgreSQL Global Development Group, 2021)
    - Facebook Graph API key (Facebook, 2024)
    - Facebook Graph API Integration in Python
    - Visual Studio Code (Visual Studio Code, 2024)
- Tools for Testing and Experiments
    - Jupyter Notebook Python 3
    - Libraries Pandas (McKinney, et al., 2021)
    - NumPy (Harris et al., 2020)
    - Seaborn (Seaborn Development Team, 2021)
    - MATLAB (The MathWorks Inc., 2021)

CHAPTER 4: EXPERIMENTATAL EVALUATION AND ANALYSIS

To evaluate the system, we used the Facebook-eBay dataset **(Zhang & Pennacchiotti, 2013)** of real facebook activity and ecommerce purchases of anonymous users on a study conducted by eBay engineers Zhang and Pennacchiotti. The dataset utilized in our research originates from a repository of eBay's "Facebook connect" users. It comprises a randomized sample of 13,619 eBay users who anonymously signed up using their Facebook accounts between June and August 2012. The dataset is filtered for users below the age of 18 and those lacking Facebook likes or devoid of any eBay purchases in 2012. For each user, the dataset retains the subsequent details:

1. Fundamental demographic data procured from Facebook, encompassing age and gender.
2. Facebook likes along with their respective categories.
3. A roster of items bought on eBay from January to August 2012, including item names and categories.

Table 4.1 shows example information of a user and his activity on both domains from the dataset.

Name	Anonymous
Gender	Male
Age Group	35-44
Facebook Likes	Beatles (Musician/band) iPhone 5 (Electronics) Starbucks (Food/beverage) Walt Disney Studios (Movie)
eBay Purchases	iPhone 4S (Electronics) Beatles T-shirt (Clothing) Beatles Mug (Collectibles)

Table 4.1 Example User Information of Facebook-eBay Dataset

From looking at **Table 4.1**, we can already tell the user's interests are similar on both platforms where the like Beatles music on Facebook and buy their shirts on eBay. This is the assumption we ride on to believe There is lot of rich information we can leverage from user activity on social media to predict purchase behaviour and interests.

Table 4.2 shows the statistics of the dataset used for experiments. It is a large dataset with over million activities on both platforms to utilize.

Users	13,619
Facebook Likes	4,165,690
Facebook pages	1,373,984
eBay purchases	628,753

Table 4.2 Facebook-eBay Dataset Summary Statistics

4.1 Evaluation Parameters

We adopt a comprehensive evaluation approach to assess the performance of our cross-domain social recommender system. Our evaluation focuses on two key metrics: Precision, Recall and F-score. The choice on sticking with this two lies in

1. Precision shows accuracy and we aim to improve recommendation accuracy **(Adomavicius & Tuzhilin, 2005).**
2. These are the same metrics used by the closest existing systems for smooth comparison.

To understand the evaluation parameters used in this research let us consider table 4.3 below:

	Purchased	Not Purchased
Recommended	**Tue Positive (TP)** (Recommended and Purchased)	**False Positive (FP)** (Not purchased on recommended)
Not Recommended	**True Negative (TN)** (Not recommended and purchased	**False Negative (FN)** (Not purchased and not recommended)

Table 4.3 Confusion Matrix for Evaluation

4.1.1 Precision

The precision metric is the ratio of relevant items retrieved to the total items in the recommendation system **(Adomavicius & Tuzhilin, 2005).** If TP represents the fraction of items the user is interested in and FP represents the fraction of items the user is not interested in, precision is mathematically expressed as:

$$Precision = \frac{TP}{TP + FP} = \frac{Recommended\ and\ Purchased}{All\ recommended\ Items}$$

Consider a scenario where True Positive (TP) represents the number of items the user is interested in, and False Positive (FP) represents the number of items the user is not interested in. For example, if our precision is 60%, it indicates that 60% of the recommendations made are relevant to the user.

Suppose we are recommended with items A, B, C, and D and the user is interested in items B and D. In this case, precision is calculated as:
Precision = BD / ABCD = 2/4 = 0.5
Here, the precision is 0.5, indicating that half of the recommended items are relevant to the user.

4.1.2 Recall

Recall, a metric indicating the fraction of relevant items that were successfully retrieved out of all relevant items **(Herlocker et al., 1999),** is calculated as follows:

$$Recall = \frac{TP}{TP + FN} = \frac{Recommended\ and\ Purchased}{All\ relevant\ Items}$$

In this context, True Positive (TP) represents the number of items the user is interested in, and False Negative (FN) represents the number of relevant items that were not retrieved.
For instance, if our recall is 70%, it means that 70% of the items the user is interested in were successfully retrieved but which were purchased.

Let us consider an example: Suppose we are recommended with items A, B, C, and D and the user is interested in items B, C, and D. The recall is then calculated as: Recall = (B, C, D) / (B, C, D) . The recall value will be between 0 and 1, where 1 indicates that all relevant items were successfully retrieved by the system.

4.1.3 F-score

F1 Score is a metric that combines both precision and recall into a single value, providing a balanced measure of a system's performance **(Herlocker et al., 1999).** It is calculated using the following formula:

$$Fscore = \frac{2\times\ Precision \times Recall}{Precision + Recall}$$

In the context of recommender systems, F1 Score considers both how many of the recommended items are relevant (Precision) and how many of the relevant items were successfully retrieved (Recall). A higher F1 Score indicates a better balance between

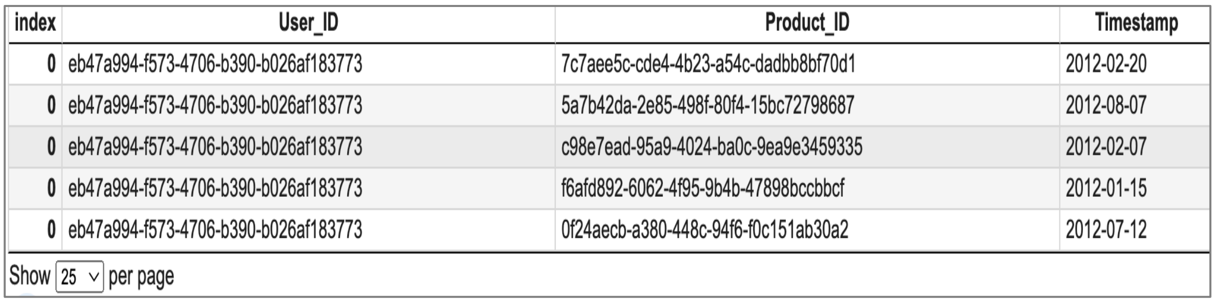
precision and recall. It ranges from 0 to 1, where 1 indicates a perfect balance, meaning that both precision and recall are maximized.
Let's consider an example where we have recommended items A, B, C, and D at the and the user is interested in items B, C, and D. The F1 Score is then calculated based on the Precision and Recall values for this scenario.

4.1.4 Experimental Setup

1. The dataset used for testing the systems is comparable dataset with comparable systems. Existing systems would not be able to test on such data because of the integration of the two domains to share product details.
2. The experiments were run a series of over 10 times to get a range of values for precision and recall. The highest recorded within the range of the 10-time run was used to evaluate the system.
3. The systems compared include GaoLinRec23 (Gao et al., 2023), WangZhoaHeLi15 (Wang et al., 2015), ZhangPenacchiotti15 (Zhang & Pennacchiotti, 2015) an our proposed Facebook-Data Cross Domain Recommender System (FD-CDR’23)

4.2 Results and Analysis

The very first step is to run MLTU on the dataset to retrieve a usable data format for the recommender system.
The ecommerce historic data as shown in **Figure 4.1** contains UserIDs and purchased products IDs as extracted using MLTU (Mine Likes and Transactions per User) of this thesis. In this Data, we are interested in what purchases have users had.

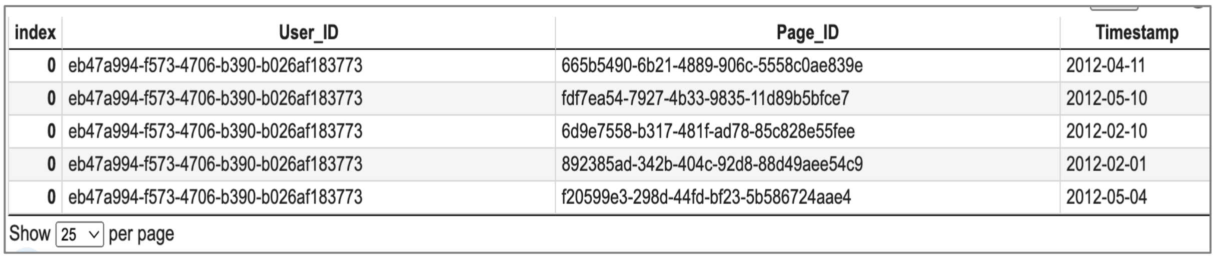


index	User_ID	Product_ID	Timestamp
0	eb47a994-f573-4706-b390-b026af183773	7c7aee5c-cde4-4b23-a54c-dadbb8bf70d1	2012-02-20
0	eb47a994-f573-4706-b390-b026af183773	5a7b42da-2e85-498f-80f4-15bc72798687	2012-08-07
0	eb47a994-f573-4706-b390-b026af183773	c98e7ead-95a9-4024-ba0c-9ea9e3459335	2012-02-07
0	eb47a994-f573-4706-b390-b026af183773	f6afd892-6062-4f95-9b4b-47898bccbbcf	2012-01-15
0	eb47a994-f573-4706-b390-b026af183773	0f24aecb-a380-448c-94f6-f0c151ab30a2	2012-07-12

Show 25 per page

Figure 4.1 Extracted e-commerce purchased of Facebook-eBay dataset.

Figure 4.2 also showcases liked pages per user extracted by MLTU of this thesis.

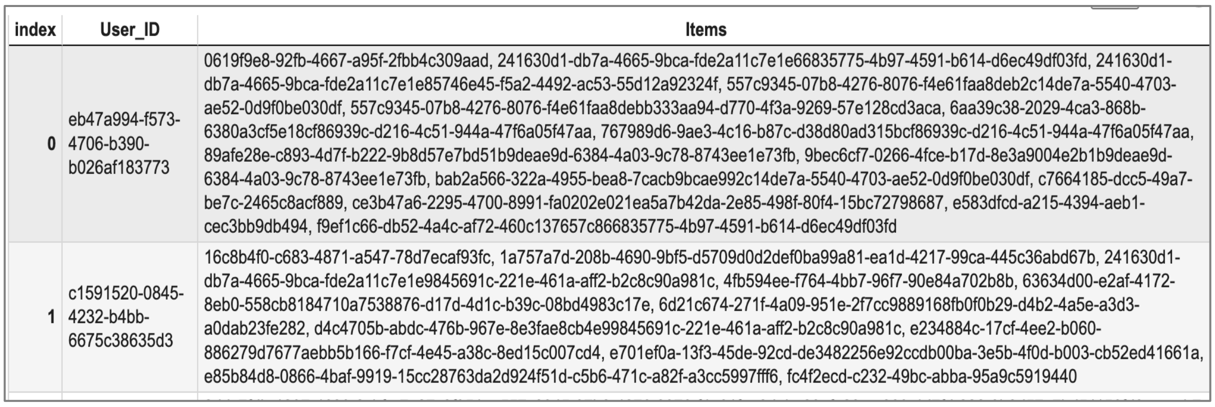


index	User_ID	Page_ID	Timestamp
0	eb47a994-f573-4706-b390-b026af183773	665b5490-6b21-4889-906c-5558c0ae839e	2012-04-11
0	eb47a994-f573-4706-b390-b026af183773	fdf7ea54-7927-4b33-9835-11d89b5bfce7	2012-05-10
0	eb47a994-f573-4706-b390-b026af183773	6d9e7558-b317-481f-ad78-85c828e55fee	2012-02-10
0	eb47a994-f573-4706-b390-b026af183773	892385ad-342b-404c-92d8-88d49aee54c9	2012-02-01
0	eb47a994-f573-4706-b390-b026af183773	f20599e3-298d-44fd-bf23-5b586724aae4	2012-05-04

Show 25 per page

Figure 4.2 Extracted facebook likes of facebook-eBay dataset.

The proposed MLTU converts the dataset into itemsets of per user as explained in **section 3.1.1**, which contains all the IDS of purchased products and liked pages per user as shown in Figure 4.3.



index	User_ID	Items
0	eb47a994-f573-4706-b390-b026af183773	0619f9e8-92fb-4667-a95f-2fbb4c309aad, 241630d1-db7a-4665-9bca-fde2a11c7e1e66835775-4b97-4591-b614-d6ec49df03fd, 241630d1-db7a-4665-9bca-fde2a11c7e1e85746e45-f5a2-4492-ac53-55d12a92324f, 557c9345-07b8-4276-8076-f4e61faa8deb2c14de7a-5540-4703-ae52-0d9f0be030df, 557c9345-07b8-4276-8076-f4e61faa8debb333aa94-d770-4f3a-9269-57e128cd3aca, 6aa39c38-2029-4ca3-868b-6380a3cf5e18cf86939c-d216-4c51-944a-47f6a05f47aa, 767989d6-9ae3-4c16-b87c-d38d80ad315bcf86939c-d216-4c51-944a-47f6a05f47aa, 89afe28e-c893-4d7f-b222-9b8d57e7bd51b9deae9d-6384-4a03-9c78-8743ee1e73fb, 9bec6cf7-0266-4fce-b17d-8e3a9004e2b1b9deae9d-6384-4a03-9c78-8743ee1e73fb, bab2a566-322a-4955-bea8-7cacb9bcae992c14de7a-5540-4703-ae52-0d9f0be030df, c7664185-dcc5-49a7-be7c-2465c8acf889, ce3b47a6-2295-4700-8991-fa0202e021ea5a7b42da-2e85-498f-80f4-15bc72798687, e583dfcd-a215-4394-aeb1-cec3bb9db494, f9ef1c66-db52-4a4c-af72-460c137657c866835775-4b97-4591-b614-d6ec49df03fd
1	c1591520-0845-4232-b4bb-6675c38635d3	16c8b4f0-c683-4871-a547-78d7ecaf93fc, 1a757a7d-208b-4690-9bf5-d5709d0d2def0ba99a81-ea1d-4217-99ca-445c36abd67b, 241630d1-db7a-4665-9bca-fde2a11c7e1e9845691c-221e-461a-aff2-b2c8c90a981c, 4fb594ee-f764-4bb7-96f7-90e84a702b8b, 63634d00-e2af-4172-8eb0-558cb8184710a7538876-d17d-4d1c-b39c-08bd4983c17e, 6d21c674-271f-4a09-951e-2f7cc9889168fb0f0b29-d4b2-4a5e-a3d3-a0dab23fe282, d4c4705b-abdc-476b-967e-8e3fae8cb4e99845691c-221e-461a-aff2-b2c8c90a981c, e234884c-17cf-4ee2-b060-886279d7677aebb5b166-f7cf-4e45-a38c-8ed15c007cd4, e701ef0a-13f3-45de-92cd-de3482256e92ccdb00ba-3e5b-4f0d-b003-cb52ed41661a, e85b84d8-0866-4baf-9919-15cc28763da2d924f51d-c5b6-471c-a82f-a3cc5997fff6, fc4f2ecd-c232-49bc-abba-95a9c5919440

Figure 4.3 Generated Itemset of facebook-eBay Dataset.

That data is then split on 80% for training and 20% for testing. Since the itemsets per user contains 13,619 users, 10,895 of those user's itemsets would be used as training data to generate the association rules and 2,724 would be used for evaluating the system.

We then now run the recommender through on series of different number of users ranging from 10 to 1000 and then compare to existing systems on recall and precision as shown in **Table 4.3** below:

Recommender System	Number of Users	Precision	Recall	F1Score
GaoLinRec23	100	0.222	-	-
	1000	0.216	-	-
WangZhoaHeLi15	100	0.104	0.07	0.067797
	1000	0.120	0.03	0.075000
ZhangPenacchiotti15	100	0.6	0.7	0.647059
	1000	0.4	0.63	0.480000
FD-CDR'23	100	0.61	0.54	0.697248
	1000	0.57	0.44	0.682927

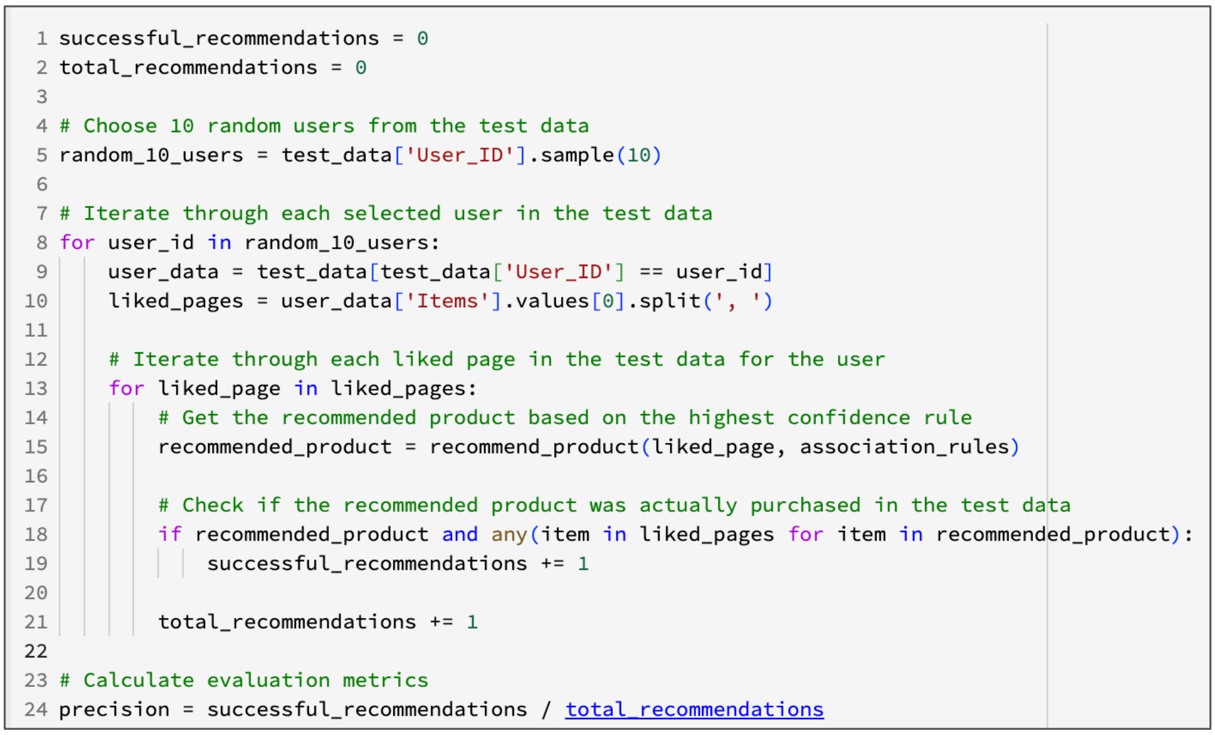
Table 4.4 Evaluation Results from Experiments on Tested Systems

In this experimentation the same range of number of users was used on all systems to test on comparable datasets and comparable algorithms over a series of run and the highest value was recorded. FD-CDR'23 and ZhangPenacchiotti15 (Zhang & Penacchiotti, 2015) were run on the same dataset, the rest were run on comparable datasets.

Precision scores are crucial in evaluating the reliability of recommender systems. Higher precision indicates a better ability to recommend items that align with users' preferences and interests.

1. GaoLinRec23: The precision scores for GaoLinRec23 indicate its effectiveness in making relevant recommendations, especially at 100 users where it achieves a precision of 0.56. This suggests that over half of the recommended items are relevant to the users. However, there is a slight decrease in precision as the user count increases to 1000, indicating potential scalability challenges or a decrease in recommendation accuracy with a larger user base.
2. WangZhoaHeLi15: shows lower precision scores compared to the other systems, both at 100 and 1000 users. This suggests that the precision of recommendations generated by this system is relatively low. It might struggle to consistently provide highly relevant suggestions to users, impacting the overall quality of the recommendations.

3. ZhangPenacchiotti15: demonstrates reasonable precision at 100 users (0.6), but its precision drops significantly at 1000 users (0.4). This decline suggests that the system may face challenges in maintaining recommendation accuracy as the user base grows. It could be less effective in handling the increased complexity of making relevant suggestions to a larger audience.
4. FD-CDR'23: The proposed FD-CDR'23 consistently achieves high precision scores, indicating a robust ability to make accurate and relevant recommendations. At both 100 and 1000 users, FD-CDR'23 outperforms other systems in terms of precision. This implies that most of the recommended items are well-received by users, showcasing the effectiveness of the recommender system.



```
 1 successful_recommendations = 0
 2 total_recommendations = 0
 3
 4 # Choose 10 random users from the test data
 5 random_10_users = test_data['User_ID'].sample(10)
 6
 7 # Iterate through each selected user in the test data
 8 for user_id in random_10_users:
 9     user_data = test_data[test_data['User_ID'] == user_id]
10     liked_pages = user_data['Items'].values[0].split(', ')
11
12     # Iterate through each liked page in the test data for the user
13     for liked_page in liked_pages:
14         # Get the recommended product based on the highest confidence rule
15         recommended_product = recommend_product(liked_page, association_rules)
16
17         # Check if the recommended product was actually purchased in the test data
18         if recommended_product and any(item in liked_pages for item in recommended_product):
19             successful_recommendations += 1
20
21         total_recommendations += 1
22
23 # Calculate evaluation metrics
24 precision = successful_recommendations / total_recommendations
```

Figure 4.4 Function to compute evaluation metrics.

It is important to note that in selecting the recommended products for FD-CDR, the products with the highest confidence were selected on the sample of random users on the test data as shown line 5 of **Figure 4.4**

The F1 score is a metric that combines precision and recall into a single value, providing a balance between false positives and false negatives. Let's interpret the F1 scores for each system:

1. WangZhoaHeLi15:

The F1 score for 100 users is 0.067797, and for 1000 users, it's 0.075000. While there is a slight improvement in the F1 score for 1000 users, the scores are still relatively low, suggesting challenges in achieving a balance between precision and recall.

2. ZhangPenacchiotti15:

The F1 score for 100 users is 0.647059, and for 1000 users, it's 0.480000. While the F1 scores are reasonable, there is a notable decrease in performance when moving from 100 to 1000 users. This suggests a potential scalability or generalization issue with the system.

3. FD-CDR'23: The F1 score for 100 users is 0.697248, and for 1000 users, it's 0.682927. These scores are relatively high, indicating a good balance between precision and recall. The system appears to perform well in identifying relevant items.

Let us present the data on a graph to understand the trend of performance over time on the various systems.

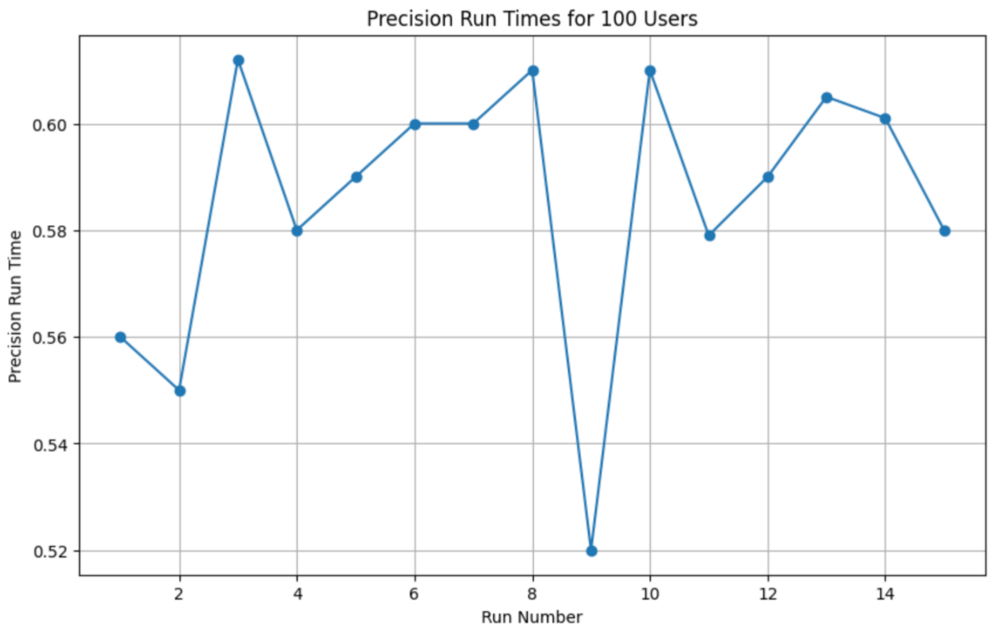


Figure 4.5 Precision run multiple times on FD-CDR'23

As demonstrated in **Figure 4.5**, the precision run times for FD-CDR exhibit some variability, with values ranging between 0.52 and 0.61. This indicates that the precision scores are not static and can vary across different runs. The system might be sensitive to certain conditions or user interactions, leading to fluctuations in the recommendation accuracy.

While there is variability, there is a pattern of relatively stable precision around the range of 0.58 to 0.61 in several consecutive runs. This stability suggests that, under certain conditions or datasets, FD-CDR consistently performs well in maintaining high precision. This is a positive sign as it indicates the system's robustness and reliability in delivering accurate recommendations for 100 users.

The run times present opportunities for optimization. If the goal is to achieve even higher precision or reduce variability, further fine-tuning of the recommender algorithm, parameter adjustments, or incorporating additional features might be considered.

Let us look at precision, recall and F1 score run graph on the tested systems.

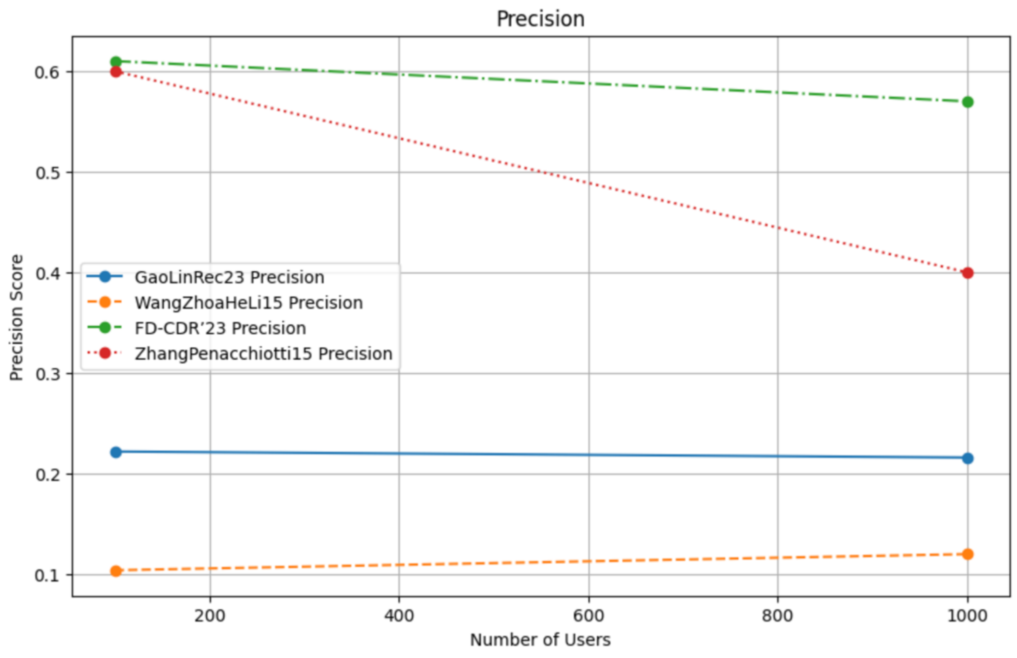


Figure 4.6 Precision Results for Tested Systems Compared to FD-CDR'23

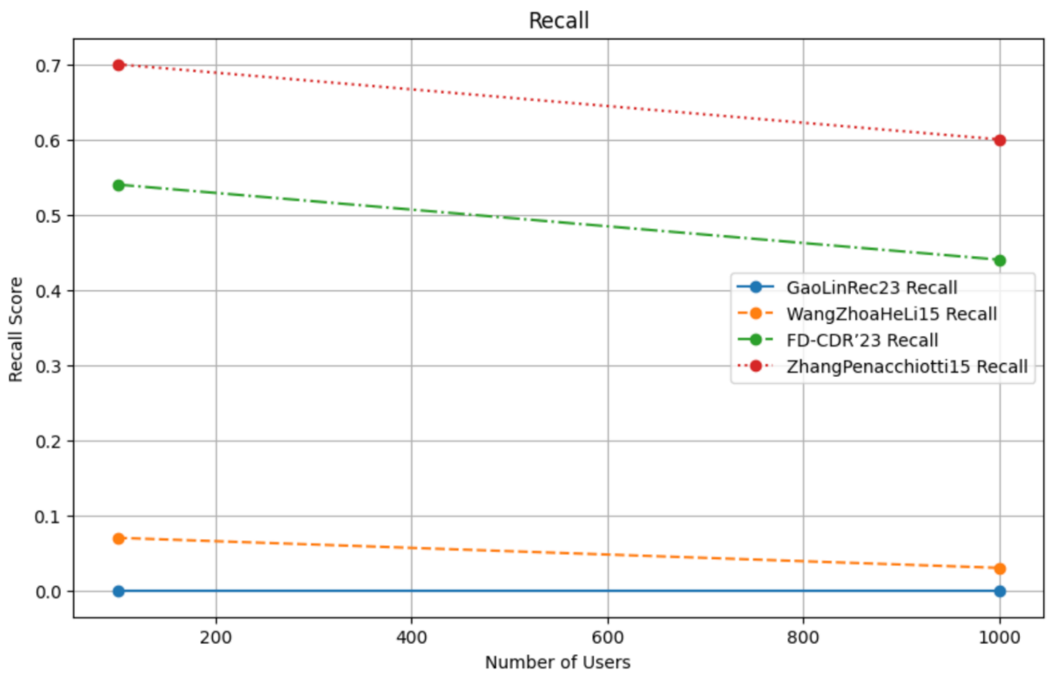


Figure 4.7 Recall Results for Tested Systems Compared to FD-CDR'23

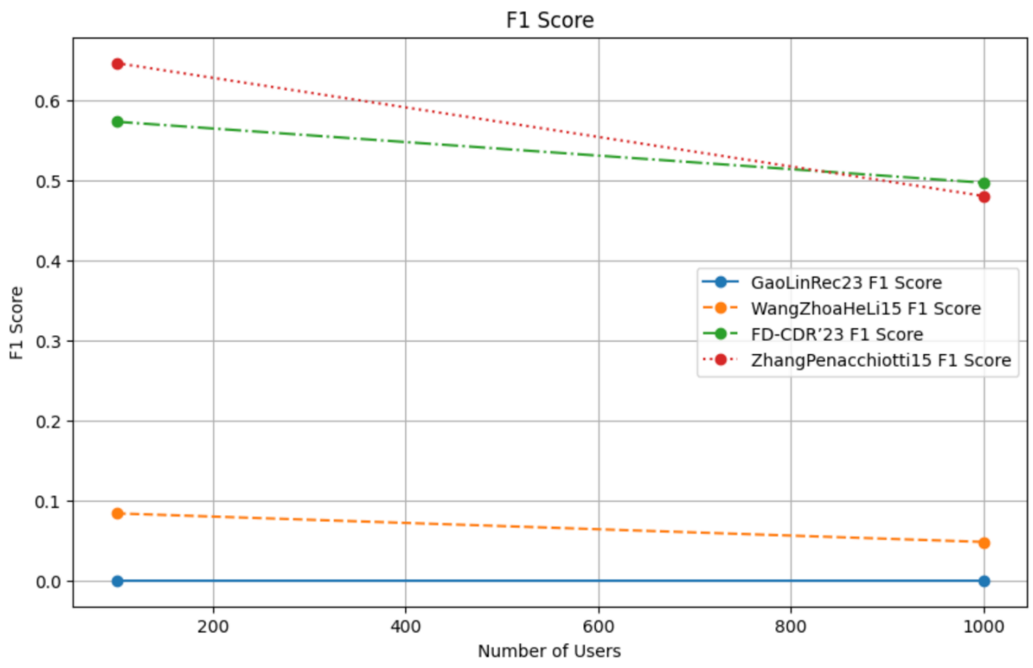


Figure 4.8 F1 Score Results for Tested Systems Compared to FD-CDR'23

Precision measures the accuracy of a recommender system by calculating the fraction of relevant items among the recommended items. In the context of our study, precision is a key metric as it signifies the effectiveness of the recommendations made to users.

From the results analysis as shown in **Figure 4.6, 4.7 and 4.8**, we realize that our proposed system that's better in accurately recommending items for 100 users and more.

(Gao et al., 2023) exhibits a moderate precision, hovering around 22.2% for 100 users and slightly decreasing to 21.6% as the user base expands to 1000. This suggests that, on average, about 22% of the recommended items are relevant to the user's interests. While the system provides reasonably accurate recommendations, there is a room for improvement.

(Wang et al., 2015), on the other hand, starts with a precision of 10.4% for 100 users, but interestingly, precision increases to 12% with a larger user base of 1000. This improvement might indicate that the system benefits from a larger dataset, leading to more accurate recommendations for a broader user population.

Our proposed system, FD-CDR'23 stands out with a high precision of 61% for 100 users, showcasing its effectiveness in providing accurate recommendations. Impressively, the precision remains robust even as the user base expands to 1000 users, maintaining a precision of 57%. This suggests that FD-CDR'23 excels in accurately identifying relevant items, making it a commendable choice for recommendation tasks.

In the realm of precision, FD-CDR'23 outperforms its counterparts, maintaining a high level of accuracy across different user populations because there exist countless number of combinations of likes and purchases that provide high confident rules for recommending products users are interested in.

The results indicate that as the user base grows, FD-CDR'23 consistently delivers precise recommendations, showcasing its reliability in providing relevant suggestions.. We also believe as social media usage has increased enormously in recent years and thus a more recent dataset would yield better results.

CHAPTER 5: CONCLUSION AND FUTURE WORKS

In conclusion, this research presents a novel approach to enhancing cross-domain recommendation systems through the integration of social media and e-commerce data. The proposed system demonstrates the potential of utilizing patterns to capture hidden associations between user activities in different domains, leading to more accurate and personalized recommendations.

The experimental evaluation showcases the effectiveness of the developed system in terms of precision and recall. We successfully proved that users' interests are similar on these two domains and answered the research question of recommending items to users on the ecommerce platform with no activity, by mining activity from the social media domain. In conclusion, the FD-CDR'23 recommender system stands out as a robust and efficient solution for making accurate recommendations. Its superior precision and commendable recall, coupled with scalability, position FD-CDR'23 as a promising choice for practical implementation in various recommendation scenarios. The findings of this study provide valuable guidance for researchers, developers, and businesses seeking effective recommender systems, emphasizing the noteworthy performance of FD-CDR'23 in delivering accurate and relevant recommendations to users.

However, there are several avenues for future exploration in this domain.

1. One major setback of the proposed algorithm is answering the question what happens when a new user likes a post that has not been liked before? The system would not be able to generate recommendations for unknown posts. A good direction in the future is to explore ways to handle scenarios of users liking unknown posts and how else we can discover their preferences and recommend a product to them.
2. One significant direction is to explore the reverse scenario, where social media and e-commerce domains are exchanged, allowing recommendations to flow from the e-commerce domain back to social media or even between both domains simultaneously. This two-way recommendation system could potentially

uncover even more nuanced user preferences and behaviours across platforms, leading to more comprehensive and context-aware recommendations.

3. Additionally, further research could focus on adapting the proposed approach to different types of cross-domain recommendation scenarios, such as music and movie recommendations or news and entertainment recommendations, thereby broadening its applicability and impact.

References

1. Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), 934-949.

2. Aggarwal, C. C. (2016). Recommender Systems. Springer International Publishing. https://doi.org/10.1007/978-3-319-29659-3

3. Agrawal, R. and Srikant, R. (1994) Fast Algorithms for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Data Bases, Santiago de Chile, 12-15 September 1994, 487-499.

4. Agrawal R, Mannila H, Srikant R, Toivonen H, Verkamo AI (1996) Fast discovery of association rules. In: Fayyad UM, Piatetsky-Shapiro G, Smyth P, Uthurusamy R (eds) Advances in knowledge discovery and data mining. AAAI Press, Menlo Park, pp 307–328

5. A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, “Methods and metrics for cold-start recommendations,” in Proc. 25th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2002, pp. 253–260.

6. Anwar, T., & Uma, V. (2022). CD-SPM: Cross-domain book recommendation using sequential pattern mining and rule mining. Journal of King Saud University - Computer and Information Sciences, 34(3), 793–800. https://doi.org/10.1016/j.jksuci.2019.01.012

7. Apté, C., & Weiss, S. (1997). Data mining with decision trees and decision rules. Future Generation Computer Systems, 13(2), 197–210. https://doi.org/10.1016/S0167-739X(97)00021-6

8. A. P. Singh and G. J. Gordon, “Relational learning via collective matrix factorization,” in Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2008, pp. 650–658.

9. Bahaweres, R. B., & Almujaddidi, A. R. (2022). Improving Recommender Systems Performance with Cross-domain Scenario: Anime and Manga Domain Studies. 2022 9th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), 361–365. https://doi.org/10.23919/EECSI56542.2022.9946560

10. Bhatta, R., Ezeife, C. I., Butt, M. (2019). Mining Sequential Patterns of Historical Purchases for E-commerce Recommendation. Submitted to International Conference on Big Data Analytics and Knowledge Discovery, 57-72.

11. Bin Li. 2011. Cross-domain collaborative filtering: A brief survey. In International Conference on Tools with Artificial Intelligence. 1085–1086.

12. Boyd, D. M., & Ellison, N. B. (2007). Social Network Sites: Definition, History, and Scholarship. Journal of Computer-Mediated Communication, 13(1), 210–230. https://doi.org/10.1111/j.1083-6101.2007.00393.x

13. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), 331-370. DOI: 10.1023/A:1021240730564

14. Cantador, I., Fernández-Tobías, I., Berkovsky, S., & Cremonesi, P. (2015). Cross-Domain Recommender Systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), Recommender Systems Handbook (pp. 919–959). Springer US. https://doi.org/10.1007/978-1-4899-7637-6_27

15. Chen, J., & WU, C. (2021). The Design of Cross-border E-commerce Recommendation System Based on Big Data Technology. 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), 381–384. https://doi.org/10.1109/ICSP51882.2021.9409014

16. Chen, M., Mao, S., Zhang, Y., & Leung, V. C. M. (2014). Big Data: Related Technologies, Challenges and Future Prospects. Springer International Publishing. https://doi.org/10.1007/978-3-319-06245-7

17. Da Cao, Xiangnan He, Liqiang Nie, Xiaochi Wei, Xia Hu, Shunxiang Wu, and Tat-Seng Chua. 2017. Cross-platform app recommendation by jointly modeling

ratings and texts. TOIS 35, 4 (2017), 37.

18. Dai, J., & Zeng, B. (2016). An Association Rule Algorithm for Online E-Commerce Recommendation Service. Journal of Economics, Business and Management, 4(10), 573–576. https://doi.org/10.18178/joebm.2016.4.10.454x

19. Digital 2021: Global Overview Report—DataReportal – Global Digital Insights. (n.d.). Retrieved 6 June 2023, from https://datareportal.com/reports/digital-2021-global-overview-report

20. Dunham, M. H. (2006). Data Mining: Introductory And Advanced Topics. Pearson Education. https://books.google.ca/books?id=O6F9iwsqZQwC

21. Dwivedi, Y. K., Ismagilova, E., Rana, N. P., & Raman, R. (2023). Social Media Adoption, Usage And Impact In Business-To-Business (B2B) Context: A State-Of-The-Art Literature Review. Information Systems Frontiers, 25(3), 971–993. https://doi.org/10.1007/s10796-021-10106-y

22. Eldridge, A. (2024, January 21). Instagram. Encyclopedia Britannica. https://www.britannica.com/topic/Instagram

23. Enrich, M., Braunhofer, M., & Ricci, F. (2013). Cold-Start Management with Cross-Domain Collaborative Filtering and Tags. In C. Huemer & P. Lops (Eds.), E-Commerce and Web Technologies (pp. 101–112). Springer. https://doi.org/10.1007/978-3-642-39878-0_10

24. Evans, A. D., Vraga, E. K., et al. (2017). Why We Engage: How Theories of Human Behavior Contribute to Our Understanding of User Engagement in Social Media. Computers in Human Behavior, 68, 537-547. https://doi.org/10.1016/j.chb.2016.09.041

25. Facebook. (2024). Graph API Documentation. Retrieved January 22, 2024, from https://developers.facebook.com/docs/graph-api

26. Fuchs, C. (2014). Social Media: A Critical Introduction. https://doi.org/10.4135/9781446270066

27. Gao, C., Chen, X., Feng, F., Zhao, K., He, X., Li, Y., & Jin, D. (2019). Cross-domain Recommendation Without Sharing User-relevant Data. In Proceedings of the World Wide Web Conference (WWW) (pp. 491–502).

28. Gao, C., Lin, T.-H., Li, N., Jin, D., & Li, Y. (2023). Cross-Platform Item Recommendation for Online Social E-Commerce. IEEE Transactions on Knowledge and Data Engineering, 35(2), 1351-1364. https://doi.org/10.1109/TKDE.2021.3098702

29. Gupta, D., & Rani, R. (2019). A study of big data evolution and research challenges. Journal of Information Science, 45(3), 322–340. https://doi.org/10.1177/0165551518789880

30. Hall, M. (2024, January 21). Facebook. Encyclopedia Britannica. https://www.britannica.com/topic/Facebook

31. Han, J. and Kamber, M. (2000) Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco.

32. Han, J., Pei, J., & Tong, H. (2022). Data Mining: Concepts and Techniques. Elsevier Science. https://books.google.ca/books?id=NR1oEAAAQBAJ

33. Harris, Charles R.; et al. (2020). NumPy (Version 1.26.3) [Computer software]. Retrieved from https://numpy.org/

34. Hartigan, J. A., & Wong, M. A. (1979). A K-Means Clustering Algorithm. Journal of the Royal Statistical Society: Series C (Applied Statistics), 28(1), 100–108. https://doi.org/10.2307/2346830

35. Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.

36. Hewitt, A., & Forte, A. (2006). Crossing Boundaries: Identity Management and Student/Faculty Relationships on the Facebook.

37. Hoffman, D. L., & Fodor, M. (2010). Can You Measure the ROI of Your Social Media Marketing? MIT Sloan Management Review. https://sloanreview.mit.edu/article/can-you-measure-the-roi-of-your-social-media-marketing/

38. Huilgol, P. (2019, August 24). Accuracy vs. F1-Score. Analytics Vidhya. https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2

39. Hwangbo, H., & Kim, Y. (2017). An empirical study on the effect of data sparsity and data overlap on cross domain collaborative filtering performance. Expert Systems with Applications, 89, 254–265. https://doi.org/10.1016/j.eswa.2017.07.041

40. Malaeb, M. (2020, September 2). Recall and Precision at k for Recommender Systems. Medium. https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54

41. Meng Jiang, Peng Cui, Xumin Chen, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2015. Social recommendation with cross-domain transferable knowledge. TKDE 27, 11 (2015), 3084–3097.

42. Ishwarappa, & Anuradha, J. (2015). A Brief Introduction on Big Data 5Vs Characteristics and Hadoop Technology. Procedia Computer Science, 48, 319–324. https://doi.org/10.1016/j.procs.2015.04.188

43. Jain, A. K., & Dubes, R. C. (1988). Algorithms for Clustering Data. Prentice Hall. https://books.google.ca/books?id=7eBQAAAAMAAJ

44. J. Tang, S. Wu, J. Sun and H. Su, "Cross-domain collaboration recommendation", Proc. Int. ACM SIGKDD Conf. Knowl. Discov. Data Mining, pp. 1285-1293, 2012.

45. Jie Tang, Sen Wu, Jimeng Sun, and Hang Su. 2012. Cross-domain collaboration recommendation. In ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). 1285–1293.

46. J. Lin, K. Sugiyama, M. Kan, and T. Chua, "Addressing cold-start in app recommendation: Latent user models constructed from twitter followers," in Proc. 36th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2013, pp. 283–292

47. Jurafsky, D. and Martin, J.H. (2019) Naive Bayes and Sentiment Classification. In Speech and Language Processing, 3rd Edition (Draft), 56-74.

48. Kaplan, A. M., & Haenlein, M. (2010). Users of the world, unite! The challenges and opportunities of Social Media. Business Horizons, 53(1), 59–68.

49. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37. DOI: 10.1109/MC.2009.263

50. Kotsiantis, S., Zaharakis, I., & Pintelas, P. (2006). Machine learning: A review of classification and combining techniques. Artificial Intelligence Review, 26, 159-190. https://doi.org/10.1007/s10462-007-9052-3

51. Kuang, H., Xia, W., Ma, X., & Liu, X. (2021). Deep Matrix Factorization for Cross-Domain Recommendation. 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 5, 2171–2175. https://doi.org/10.1109/IAEAC50856.2021.9390866

52. Laudon, K. C., & Traver, C. G. (2017). E-commerce 2017: Business, technology, society (Thirteenth Edition). Pearson.

53. Liang Hu, Jian Cao, Guandong Xu, Longbing Cao, Zhiping Gu, and Can Zhu. 2013. Personalized recommendation via cross-domain triadic factorization. In WWW. 595–606

54. Lin, J., Sugiyama, K., Kan, M., & Chua, T. (2013). Addressing cold start in app recommendation: Latent user models constructed from Twitter followers. In Proceedings of the 22nd International Conference on World Wide Web (WWW) (pp. 119-130).

55. Liu, D. R., Lai, C. H., & Lee, W. J. (2009). A hybrid of sequential rules and collaborative filtering for product recommendation. Information Sciences, 179(20), 3505-3519.

56. Li, Y., Niu, Z., Chen, W., & Zhang, W. (2011, December). Combining collaborative filtering and sequential pattern mining for recommendation in e-learning environment. In International Conference on Web-Based Learning (pp. 305-313). Springer, Berlin, Heidelberg

57. Loni, B., Shi, Y., Larson, M., & Hanjalic, A. (2014). Cross-Domain Collaborative Filtering with Factorization Machines. In M. de Rijke, T. Kenter, A. P. de Vries, C. Zhai, F. de Jong, K. Radinsky, & K. Hofmann (Eds.), Advances in Information Retrieval (pp. 656–661). Springer International Publishing. https://doi.org/10.1007/978-3-319-06028-6_72

58. L. Chen, J. Zheng, M. Gao, A. Zhou, W. Zeng and H. Chen, "TLRec:Transfer Learning for Cross-Domain Recommendation," 2017 IEEE International Conference on Big Knowledge (ICBK), Hefei, China, 2017, pp. 167-172, doi: 10.1109/ICBK.2017.30.

59. L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu and C. Zhu, "Personalized recommendation via cross-domain triadic factorization", Proc. Int. World Wide Web Conf., pp. 595-606, 2013.

60. Ma, H., Yang, H., Lyu, M. R., & King, I. (2011). Learning to Recommend with Social Trust Ensemble

61. Malik, A. (2022, February 9). Instagram rolls out 'Your activity' and 'Security checkup' features worldwide. TechCrunch. https://techcrunch.com/2022/02/09/instagram-rolls-out-your-activity-and-security-checkup-features-worldwide/

62. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C. & Byers, A. H. (2011). Big Data: The Next Frontier for Innovation, Competition, and Productivity (). McKinsey Global Institute.

63. McKinney, Wes; et al. (2021). Pandas (Version 2.2.0) [Computer software]. Retrieved from https://pandas.pydata.org/

64. Naimi, A. I., & Westreich, D. J. (2014). Big Data: A Revolution That Will Transform How We Live, Work, and Think. American Journal of Epidemiology, 179(9), 1143–1144. https://doi.org/10.1093/aje/kwu085

65. Nickerson, R. (2002). AN E-COMMERCE SYSTEM MODEL. https://www.semanticscholar.org/paper/AN-E-COMMERCE-SYSTEM-MODEL-Nickerson/8fec924eba8c80f4d0e77829f3711826ce8ebe08

66. Nield, D. (2019, April 1). How Nathan Pyle's Strange Planet went from Instagram to bestseller. The Verge. https://www.theverge.com/2019/4/1/18280819/nathan-pyle-strange-planet-comic-instagram-interview

67. Panteli, A., & Boutsinas, B. (2023). Addressing the Cold-Start Problem in Recommender Systems Based on Frequent Patterns. Algorithms, 16(4), 182. https://doi.org/10.3390/a16040182

68. Pan, W., Xiang, E. W., Liu, N. N., & Yang, Q. (2010). Transfer learning in collaborative filtering for sparsity reduction. In Proceedings of the 10th IEEE International Conference on Data Mining (ICDM) (pp. 956-961).

69. Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In The Adaptive Web (pp. 325-341). Springer. DOI: 10.1007/3-540-67822-x_17

70. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M.-C. (2001). PrefixSpan,: Mining sequential patterns efficiently by prefix-projected pattern growth. Proceedings 17th International Conference on Data Engineering, 215–224. https://doi.org/10.1109/ICDE.2001.914830

71. PostgreSQL Global Development Group. (2021). PostgreSQL Database [Computer software]. Retrieved from https://www.postgresql.org/

72. Python Software Foundation. (2021). Python (Version 3.10) [Computer software]. Retrieved from https://www.python.org/

73. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work (pp. 175–186).

74. Safko, L., & Brake, D. K. (2009). The social media bible: Tactics, tools, and strategies for business success. John Wiley & Sons.

75. Salton, G., & McGill, M. J. (1983). Introduction to modern information retrieval. New York: McGraw-Hill.

76. Schafer, R. (2023, June). Introduction to Working with JSON Objects. SerpApi. Retrieved from https://serpapi.com/blog/introduction-to-working-with-json-objects

77. Seaborn Development Team. (2021). seaborn (Version 0.13.1) [Computer software]. Retrieved from https://seaborn.pydata.org/

78. Seroussi, Y., Bohnert, F., & Zukerman, I. (2011). Personalised rating prediction for new users using latent factor models. Proceedings of the 20th ACM international conference on Information and knowledge management, CIKM '11, 505-514.

79. Smith, A., & Anderson, M. (2018). Social Media Use 2018: Demographics and Statistics. Washington DC: Pew Research Center. Retrieved from https://www.pewresearch.org/internet/2018/03/01/social-media-use-in-2018/

80. S. J. Pan and Q. Yang, “A survey on transfer learning,” IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

81. Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In P. Apers, M. Bouzeghoub, & G. Gardarin (Eds.), Advances in Database Technology-EDBT '96 (pp. 1-17). Springer. https://doi.org/10.1007/BFb0014140

82. Steinbach, M., Karypis, G., & Kumar, V. (2000). A Comparison of Document Clustering Techniques [Report]. http://conservancy.umn.edu/handle/11299/215421

83. Tan, P. N., Steinbach, M., & Kumar, V. (2006). Introduction to Data Mining. Pearson Education, Inc.

84. Tarus, J. K., Niu, Z., & Kalui, D. (2018). A hybrid recommender system for e-learning based on context awareness and sequential pattern mining. Soft Computing, 22(8), 2449–2461. https://doi.org/10.1007/s00500-017-2720-6

85. The MathWorks, Inc. (2021). MATLAB (Version 3.3) [Computer software]. Retrieved from https://www.mathworks.com/

86. Tong Man, Huawei Shen, Xiaolong Jin, and Xueqi Cheng. 2017. Cross-domain recommendation: an embedding and mapping approach. In International Joint Conferences on Artificial Intelligence (IJCAI). 2464–2470.

87. Xi Zhang, Jian Cheng, Ting Yuan, Biao Niu, and Hanqing Lu. 2013. TopRec: domain-specific recommendation through community topic mining in social network. In WWW. 1501–1510.

88. Vasili, R., Xhina, E., Ninka, I., & Terpo, D. (2021). Sentiment Analysis on Social Media for Albanian Language. OALib, 08(06), 1–31. https://doi.org/10.4236/oalib.1107514

89. Visual Studio Code. (2024). Visual Studio Code [Computer software]. Retrieved from https://code.visualstudio.com/

90. Waheed, H., Anjum, M., Rehman, M., & Khawaja, A. (2017). Investigation of user behavior on social networking sites. PLOS ONE, 12(2), e0169693. https://doi.org/10.1371/journal.pone.0169693

91. Wang, J., Zhao, W., He, Y., & Li, X. (2015). Leveraging Product Adopter Information from Online Reviews for Product Recommendation. Proceedings of the International AAAI Conference on Web and Social Media, 9(1), Article 1. https://doi.org/10.1609/icwsm.v9i1.14585

92. Waters, J. (2024, January 22). Amazon.com. Encyclopedia Britannica. https://www.britannica.com/topic/Amazoncom

93. Watson, R. T., Berthon, P., Pitt, L. F., & Zinkhan, G. M. (2008). Electronic Commerce: The Strategic Perspective. https://opentextbc.ca/electroniccommerce/

94. Weaver, J., & Tarjan, P. (2013). Facebook Linked Data via the Graph API. Semantic Web, 4(3), 245–250.

95. Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2016). Collaborative Filtering and Deep Learning Based Hybrid Recommendation for Cold Start Problem. 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 874–877. https://doi.org/10.1109/DASC-PICom-DataCom-CyberSciTec.2016.149

96. Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. 2010. Transfer learning in collaborative filtering for sparsity reduction. In The AAAI Conference on Artificial Intelligence (AAAI), Vol. 10. 230–235.

97. W. Pan, E. W. Xiang, N. N. Liu, and Q. Yang, "Transfer learning in collaborative filtering for sparsity reduction," in Proc. AAAI, 2010, pp. 230–235.

98. W. X. Zhao, S. Li, Y. He, E. Y. Chang, J. -R. Wen and X. Li, "Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information," in IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 5, pp. 1147-1159, 1 May 2016, doi: 10.1109/TKDE.2015.2508816.

99. Yang, J., Chen, H., Xiong, S., Yang, Z., & Jiang, Y. (2021). Custom Data Mining Association Rules of Nixing Pottery Product Recommendation System. Proceedings of the 2020 4th International Conference on Electronic Information Technology and Computer Engineering, 89–93. https://doi.org/10.1145/3443467.3443734

100. Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. Computer 42, 8 (2009).

101. Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In International Conference on Data Mining (ICDM). 263–272.

102. Zang, T., Zhu, Y., Liu, H., Zhang, R., & Yu, J. (2022). A Survey on Cross-domain Recommendation: Taxonomies, Methods, and Future Directions (arXiv:2108.03357). arXiv. http://arxiv.org/abs/2108.03357

103.Zhang, J. (2021, March 18). What is Weibo. China Gravy. https://chinagravy.com/what-is-sina-weibo-know-your-chinese-social-media/

104. Zhang, Y., & Pennacchiotti, M. (2013). Predicting purchase behaviours from social media. WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web, 1521-1532.

105. Zhang, Y., & Pennacchiotti, M. (2013). Recommending branded products from social media. Proceedings of the 7th ACM Conference on Recommender Systems, 77–84. https://doi.org/10.1145/2507157.2507170

106. Zhang, Z., Wang, H., & Chen, H. (2021). Big data in e-commerce: A review. ACM Transactions on Management Information Systems (TMIS), 12(2), 1-32. doi:10.1145/3447202

107.Zhao, W. X., Li, S., He, Y., Chang, E. Y., Wen, J.-R., & Li, X. (2016). Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information. IEEE Transactions on Knowledge and Data Engineering, 28(5), 1147-1159. https://doi.org/10.1109/TKDE.2015.2508816

108. Zhao, X. W., Guo, Y., He, Y., Jiang, H., Wu, Y., & Li, X. (2014). We know what you want to buy: A demographic-based system for product recommendation on microblogs. Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1935–1944. https://doi.org/10.1145/2623330.2623351

Vita Auctoris

NAME	Emmanuel Ainoo
PLACE OF BIRTH	Accra, Ghana
YEAR OF BIRTH	1997
EDUCATION	Ashesi University, Accra Ghana (2015 – 2019)
	University of Windsor, Ontario, Canada (2022 – 2024)