

4-27-2018

Recommending Best Products from E-commerce Purchase History and User Click Behavior Data

Ying Xiao

University of Windsor

Follow this and additional works at: <https://scholar.uwindsor.ca/etd>

Recommended Citation

Xiao, Ying, "Recommending Best Products from E-commerce Purchase History and User Click Behavior Data" (2018). *Electronic Theses and Dissertations*. 7455.

<https://scholar.uwindsor.ca/etd/7455>

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.

Recommending Best Products from E-commerce Purchase History and User Click
Behavior Data

By

Ying Xiao

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

Windsor, Ontario, Canada

2018

© 2018 Ying Xiao

Recommending Best Products from E-commerce Purchase History and User Click
Behavior Data

by

Ying Xiao

APPROVED BY:

Z. Hu
Department of Mathematics & Statistics

J. Lu
School of Computer Science

C. Ezeife, Advisor
School of Computer Science

April 10, 2018

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

In E-commerce collaborative filtering recommendation systems, the main input data of user-item rating matrix is a binary purchase data showing only what items a user has purchased recently. This matrix is usually sparse and does not provide a lot of information about customer purchases or product clickstream behavior (eg., clicks, basket placement, and purchase) history, which possibly can improve product recommendations accuracy. Existing recommendation systems in E-commerce with clickstream data include those referred in this thesis as Kim05Rec, Kim11Rec, and Chen13Rec. Kim05Rec forms a decision tree on click behavior attributes such as search type and visit times, discovers the possibility of a user putting products into the basket and uses the information to enrich the user-item rating matrix. If a user clicked a product, Kim11Rec then finds the associated products for it in three stages such as click, basket and purchase, uses the lift value from these stages and calculates a score, it then uses the score to make recommendations. Chen13Rec measures the similarity of users on their category click patterns such as click sequences, click times and visit duration; it then can use the similarity to enhance the collaborative filtering algorithm. However, the similarity between click sequences in sessions can apply to the purchases to some extent, especially for sessions without purchases, this will be able to predict purchases for those session users. But the existing systems have not integrated it, or the historical purchases which shows more than whether or not a user has purchased a product before.

In this thesis, we propose HPCRec (Historical Purchase with Clickstream based Recommendation System) to enrich the ratings matrix from both quantity and quality aspects. HPCRec firstly forms a normalized rating-matrix with higher quality ratings from historical purchases, then mines consequential bond between clicks and purchases with weighted frequencies where the weights are similarities between sessions, but rating quantity is better by integrating this information. The experimental results show that our approach HPCRec is more accurate than these existing methods, HPCRec is also capable of handling infrequent cases whereas the existing methods can not.

Keywords: E-commerce recommendation system, collaborative filtering, CF, clickstream history, weighted frequent item, data mining.

DEDICATION

I would like to dedicate this thesis to my parents, supervisor, friends who have helped and supported me.

ACKNOWLEDGEMENTS

Thanks to everyone who has given me courage and confidence.

Special thanks to my supervisor, Dr. Christie Ezeife for her patience, support and professional guidance during my graduate study.

I would also like to thank my internal reader, Dr. Jianguo Lu, my external reader, Dr. Zhiguo Hu, my committee chair, Dr. Sherif Saad Ahmed, for their time, effort and valuable opinions.

TABLE OF CONTENTS

DECLARATION OF ORIGINALITY	iii
ABSTRACT.....	iv
DEDICATION	v
ACKNOWLEDGEMENTS	vi
LIST OF TABLES	x
LIST OF FIGURES	xiii
LIST OF EQUATIONS	xiv
LIST OF ALGORITHMS.....	xv
Chapter 1 INTRODUCTION.....	1
1.1 Recommendation System, and Techniques.....	5
1.1.1 Some Recommendation Systems.....	5
1.1.2 Collaborative Filtering Recommendation System.....	5
1.2 Understanding Customers in E-commerce Business	9
1.2.1 Behavior Analysis.....	9
1.2.2 Valuable Data in E-commerce.....	10
1.3 Recommendation Systems in E-commerce.....	17
1.4 Data Mining.....	18
1.4.1 Association rule mining.....	18
1.4.2 Clustering.....	19
1.4.3 Classification	20
1.5 Existing Recommendation Systems Integrated with Clickstream Data.....	21
1.6 Thesis Contributions	26
1.6.1 Observations and Thesis Hypotheses	27
1.6.2 Method Contributions.....	30
1.6.3 Feature Contributions	31

1.7 Thesis Outline	32
Chapter 2 RELATED WORK	33
2.1 E-commerce Recommendation Systems on Clickstream Data	33
2.1.1 A Stage-based Approach (Kim, Yum, Song, & Kim, 2005)	35
2.1.2 An Association Rule Approach (Kim & Yum, 2011)	39
2.1.3 A Clustering Approach (Chen & Su, 2013)	41
2.2 E-commerce Recommendation Systems on Transaction Data	45
2.2.1 Item-Item Collaborative Filtering (Linden, Smith, & York, 2003)	45
2.2.2 Combine Content-based CF and User Activity (Fan, Pan, & Jiang, 2014)	47
2.3 Sequential Similarity Measurement	51
2.3.1 Edit Distance & Modified Edit Distance	52
2.3.2 LCS: Longest Common Subsequences	53
Chapter 3 PROPOSED RECOMMENDATION SYSTEM	56
3.1 Input Data	56
3.1.1 Consequential Table	56
3.1.2 User-item Purchase Frequency Matrix	58
3.2 Proposed Method: HPCRec (Historical Purchase and Clickstream based Recommendation System)	58
3.2.1 FN: Frequency Normalization: Step 1 of HPCRec Algorithm	61
3.2.2 CSSM-Clickstream Sequence Similarity Measurement: Step 2 of HPCRec Algorithm	62
3.2.3 TWFI-Transaction-based Weighted Frequent Item: Step 3 of HPCRec Algorithm	62
3.3 An Example Application of Proposed Algorithm	63
3.3.1 Kim05Rec Method	65
3.3.2 Kim11 Method	66
3.3.3 Chen13 Method	68
3.3.4 HPCRec	71
3.4 An Example Application of Proposed Algorithm	74
3.4.1 Kim05Rec Method	75

3.4.2 Kim11 Method.....	76
3.4.3 Chen13 Method	77
3.4.4 HPCRec	79
Chapter 4 EXPERIMENTS EVALUATION AND ANALYSIS	82
4.1 Dataset and Sample Selection	82
4.2 Evaluation Metrics	82
4.3 Evaluation Result and Analysis.....	84
4.4 Implementation and Code	87
4.4.1 Develop Environment and Tools.....	87
4.4.2 Deploy Environment and Tools.....	87
4.4.3 Setup Development.....	88
4.4.4 Run on Linux Server.....	88
Chapter 5 CONCLUSION AND FUTURE WORK.....	90
REFERENCES/BIBLIOGRAPHY.....	91
VITA AUCTORIS	97

LIST OF TABLES

Table 1.1: A table of different recommendation methods	1
Table 1.2: An example of the transaction database	2
Table 1.3: An example table of users' product category/shipping preferences	2
Table 1.4: An example of user-item rating table for a movie site	3
Table 1.5: An example table of user-item rating matrix for an E-commerce site	4
Table 1.6: A piece of clickstream data from ACM 2015 RecSys challenge (Ben-Shimon et al., 2015)	4
Table 1.7: Example recommendation systems in E-commerce	5
Table 1.8: An example of a user-item rating table.....	8
Table 1.9: Typology of Shopping Strategies	10
Table 1.10: A taxonomy of input data	11
Table 1.11: User's Implicit Preferences from Activity.....	12
Table 1.12: An example of transactional purchase database	12
Table 1.13: A session-based historical transaction table	13
Table 1.14: An example of user-item rating table for a movie site	13
Table 1.15: An example of scale rating schema	13
Table 1.16: An example of rating data in E-commerce	14
Table 1.17: Session-based clickstream data.....	15
Table 1.18: An example table for Apriori frequent pattern mining.....	19
Table 1.19: Example data for clustering	20
Table 1.20: Example data for classification.....	21
Table 1.21: Comparison of existing some recommendation systems.....	24
Table 1.22: Sample consequential table.....	28
Table 1.23: A user item purchasing frequency matrix.....	29
Table 1.24: Normalized user-item purchase frequency matrix.....	29
Table 1.25: Enriched user-item purchase frequency matrix	29
Table 2.1: Literature review of clickstream data related work on variables.....	33
Table 2.2: Literature review stage-based approaches on clickstream data	35
Table 2.3: Data type collected from the experimental E-commerce site	36
Table 2.4: An example structure of collected data	36
Table 2.5: Data type collected from the experimental E-commerce site	39
Table 2.6: Example Data for Amazon CF	46
Table 2.7: Associated Customers for Wallet in Amazon CF.....	46
Table 2.8: A Table of Candidate items	46
Table 2.9: Result Table of Calculating the similarity	46
Table 2.10: Example Rating Table for Hybrid RS	49
Table 2.11: Example Preference Table for Hybrid RS	49

Table 2.12: An example of user-item rating matrix.....	49
Table 2.13: Item attribute matrix	49
Table 2.14: Preliminary Predictive Rating Table	50
Table 2.15: Updated Rating Table	50
Table 2.16: Final Predicted Rating	51
Table 2.17: Initialized Matrix for Edit Distance Calculation	52
Table 2.18: Result matrix for Edit Distance	53
Table 2.19: Edit Distance Result Table	53
Table 2.20: Initialized Matrix for LCS	54
Table 2.21: Result matrix for LCS.....	55
Table 2.22: LCS Result Table.....	55
Table 3.1: An example of consequential table.....	57
Table 3.2: Sample schema of Clickstream table.....	57
Table 3.3: Sample schema of the Transaction table	58
Table 3.4: User-item purchase frequency matrix.....	58
Table 3.5: Normalized user-item purchase frequency matrix.....	60
Table 3.6: Enriched user-item normalized purchase frequency matrix of HPCRec	61
Table 3.7: User-item rating matrix with predicted ratings for HPCRec	61
Table 3.8: Clickstream data for a walk through an example	64
Table 3.9: Purchase data for a walk through an example	64
Table 3.10: Rating matrix for a walk through example	64
Table 3.11: Product details for a walk through an example	64
Table 3.12: Decision tree for a walk through an example	65
Table 3.13: Basket placement possibilities for clicked products.....	65
Table 3.14: Useful Basket placement possibilities	66
Table 3.15: Enriched matrix for kim05Rec	66
Table 3.16: Evaluation result for kim05Rec	66
Table 3.17: Support from clickstream for user 2	67
Table 3.18: Lift from clickstream for user 2.....	67
Table 3.19: Support from purchases for user 2.....	67
Table 3.20: Lift from purchases for user 2	68
Table 3.21: Final lift for unpurchased product for user 2	68
Table 3.22: User-item matrix for Kim11Rec	68
Table 3.23: Evaluation result for Kim11Rec	68
Table 3.24: Category sequences from click sequences.....	69
Table 3.25: Visit frequency on categories	69
Table 3.26: Visit duration time on categories.....	70
Table 3.27: New similarity matrix	70
Table 3.28: User-item rating matrix.....	70

Table 3.29: Evaluation result for Chen 13	70
Table 3.30: Consequential Table	71
Table 3.31: User-item purchase frequency Table	71
Table 3.32: Normalized purchase frequency table	71
Table 3.33: An example of the similarity between clickstream sequences	72
Table 3.34: A table of weighted transactions for user 0 and 2	73
Table 3.35: A result of weighted frequent items.....	73
Table 3.36: Normalized weighted frequent items.....	73
Table 3.37: Enriched user-item matrix from HPCRec.....	74
Table 3.38: Evaluation result of HPCRec.....	74
Table 3.39: Clickstream data for a walk through example	74
Table 3.40: Purchase data for a walk through example	75
Table 3.41: Rating matrix	75
Table 3.42: Product metadata Table	75
Table 3.43: Decision tree example.....	76
Table 3.44: Support from clickstream data.....	77
Table 3.45: Support from purchase data	77
Table 3.46: Consequential Table	79
Table 3.47: User-item purchase frequency table	80
Table 3.48: Normalized user-item purchase frequency table	80
Table 3.49: Enriched user-item matrix	81
Table 4.1: Confusion Matrix.....	83

LIST OF FIGURES

Figure 1.1: Product metadata of iPhone 7 on Amazon	16
Figure 1.2: User category preference meta data on Amazon.....	16
Figure 1.3: Decision tree example	21
Figure 1.4 CF RecSys diagram integrated with clickstream data.....	22
Figure 1.5: HPCRec in CF RecSys diagram.....	27
Figure 1.6: The data source of ACM code challenge in 2005	28
Figure 2.1: A general workflow for possible actions in E-commerce sites	36
Figure 2.2: Constructed decision tree for reaching basket placement	37
Figure 2.3: An example of using constructed decision.....	38
Figure 2.4: A comparison of the original and enriched user-item rating matrix	38
Figure 2.5: Tree of an E-commerce website topology.....	42
Figure 2.6: Activity Rate Distribution	48
Figure 4.1: Evaluation on different number of sessions	85
Figure 4.2: Evaluation on different number of top-N scores	87

LIST OF EQUATIONS

Equation 1.1: Mean rating computation.....	6
Equation 1.2: The formula of cosine-based similarity.....	7
Equation 1.3: The formula of correlation-based similarity.....	7
Equation 1.4: The prediction formula of weighted sum	7
Equation 1.5: The prediction formula of mean-centered ratings	7
Equation 2.1: A formula for calculating support	40
Equation 2.2: A formula for calculation lift.....	40
Equation 2.3: Calculating the frequency a user spent on a category	42
Equation 2.4: Calculating the frequency ratio a user spent on a category	42
Equation 2.5: A category-user matrix recording the click frequency	43
Equation 2.6: Calculating the accumulated time a user spent on a category	43
Equation 2.7: Calculating the accumulated time ratio a user spent on a category..	43
Equation 2.8: A category-user matrix recording the accumulated time	44
Equation 2.9: Measure the similarity between visiting paths	44
Equation 2.10: Measure the similarity between visiting frequency.....	44
Equation 2.11: Measure the similarity between visiting accumulated time	44
Equation 2.12: Measure the similarity between two users	44
Equation 2.13: Activity Rate Formula	47
Equation 2.14: User-item Rate Calculation Formula.....	48
Equation 2.15: Preliminary Predictive Rating Formula.....	50
Equation 2.16: Hybrid RS Final Predict Formula.....	50
Equation 2.17: Edit Distance Equation	52
Equation 2.18: LCS Equation	54
Equation 3.1: Unit vector normalization.....	61
Equation 3.2: Feature scaling normalization	62
Equation 3.3: Longest common subsequence.....	62
Equation 4.1: Precision Formula.....	83
Equation 4.2: Recall Formula	83
Equation 4.3: Average Precision Formula	84

LIST OF ALGORITHMS

Algorithm 1.1: User-based Collaborative Filtering	6
Algorithm 2.1: Amazon Item-Item Collaborative Filtering.....	45
Algorithm 3.1: Algorithm for HPCRec recommendation system	59

CHAPTER 1

INTRODUCTION

Recommendation systems are techniques providing suggestions for items to be purchased, rented or used by a user. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read (Ricci, Rokach, & Shapira, 2011). Different types of recommendation system (Aggarwal, 2016) takes different input data such as user profile and preferences, user-item rating matrix, product specification, clickstream data and domain knowledge as input, then generate a list of recommendations. Table 1.1 gives the conceptual goal and input data type of different recommendation methods:

Table 1.1: A table of different recommendation methods

Method	Conceptual Goal	Input
Collaborative Filtering	Recommendations based on a collaborative approach that leverages the ratings and actions of a user and the community.	Rating matrix
Content-based	Recommendations based on the content (attributes) a user has favored in his past ratings and actions.	Rating matrix; product specification.
Knowledge-based	Give me recommendations based on a user's explicit specification of the kind of content (attributes) the user wants.	User profile; product specification; domain knowledge

For instance in Table 1.2, a recommendation system may recommend product “b” to customer “1” because two customer “3” and “7” have purchased product “a” and “c” just like customer one, and they also purchased product “b”, so product “b” would be a potentially interesting product to customer “1”.

The best products for a user in E-commerce are the most relevant product considering the user's purchasing history (Table 1.2), and the user's preferences (Table 1.3) for product features such as shipping, price, category, and sellers.

The product purchase history data is usually stored in a transactional table such as Table 1.2 where tid is the primary key transaction id, uid is the user id, the purchased products

are drawn from the set (“a”, “b”, “c”, “d”, “e”), the set in the example where time records the time of the transaction. From the Table 1.2, the following information for customer “1” can be inferred:

- Customer “1” purchased product “a”, “c”, and “e”;
- Product “a” has been sold to customer “1”, “3”, “4”, and “7”;
- Product “c” has been sold to customer “1”, “3”, and “7”;
- Product “e” has been sold to customer “1” and “4”.

Table 1.2: An example of the transaction database

tid	uid	purchase	time (yyyy.MM.dd.hh.mm.ss)
1	1	{ace}	2017.09.05.13.23.30
2	3	{aab}	2017.06.15.09.23.34
3	3	{ac}	2017.06.15.09.30.34
4	4	{ade}	2017.03.05.18.59.19
5	7	{bcd}	2017.09.25.15.33.22
6	7	{abbc}	2017.09.25.15.56.22

A user’s preferences are usually configured in their profiles such as the structure in Table 1.3, which indicates each user’s preference, such as what category of products (eg., eatable, office supply, fashion, electronics) and shipping method (eg., 2-day shipping), such valued product features to the online shopper. Value 1 in the table represents a positive preference from the user on a specific feature, while 0 means the user is not interested. For instance, user “1” may like chocolate (in category “Eatable”), printers (in category “Office supply”), but not interested in smartphones (in category “Electronics”) or fancy handbags (in category “Fashion”) according to the user profile. This data also can be extracted from the attributes of highly rated items for a user.

Table 1.3: An example table of users’ product category/shipping preferences

userId	Eatable	Office supply	Fashion	Electronics	Two-day shipping
1	1	1	0	0	1
2	0	1	1	0	1

Collaborative filtering is the most popular method used for recommendations; it uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users (Su & Khoshgoftaar, 2009). Given an incomplete user-rating matrix R of m users for n items with missing ratings (r_{uj}) of item j for user u , the user-based neighborhood model CF (Aggarwal, 2016) would predict the unknown ratings (r_{uj}) through the following steps: (1) computing the mean rating for each user u ; (2) calculating the similarity between target user u and all other users v ; (3) computing user u 's peer group P ; and then (4) predicting rating for target user u for item j .

For instance, in Table 1.4, each cell is the rating value from a user on a movie on a scale from 1 to 5. Then predicts a rating for a user on an un-rated item, such as Chris on Avatar, using the collaborative filtering algorithm in section 1.1.2, we can have a predicted rating value 4.55 using cosine similarity (Equation 1.2) and the prediction method using weighted sum without other's ratings (Equation 1.2). Finally, with a predicted rating as 4.55, we can recommend Avatar to Chris given the predicted rating is high.

Table 1.4: An example of user-item rating table for a movie site

userId\item	Star Wars	Harry Potter	Deadpool	Transformers	Avatar
Alice	2	?	3	?	5
Allen	3	1	5	?	?
Chester	1	?	?	3	4
Chris	2	4	1	1	?(4.55)

While in an **e-commerce recommendation system**, CF usually takes a binary user-item rating matrix as the example in Table 1.5 as input, where “1” indicates that a user has purchased a product before, and “?” means a user has not, only showing whether or not an item has been purchased by a user previously. The user-item rating matrix (eg., Table 1.5) in E-commerce usually contains part information of the historical transaction records (eg., Table 1.2). In Table 1.2, each row records a purchase (a collection of item names) happened in a session, and there may be multiple products for each purchase. This user-item matrix does not provide a lot of information about customer purchase history or item purchase history for purposes of improving recommendations of products to users.

Table 1.5: An example table of user-item rating matrix for an E-commerce site

userId\products	a	b	c	d	e
1	1	?	1	?	1
3	1	1	1	?	?
4	1	?	?	1	1
7	1	1	1	1	?

In addition to the rating matrix, **clickstream in E-commerce** (usually includes clicks, basket placement, and purchase activities) which is the electronic record of a user's activity on the Internet (Bucklin & Sismeiro, 2009) has been used by some E-commerce recommendation systems to address the deficiency of uninformative user-item input data. ACM Recommender Systems hosted a code challenge workshop in Austria, 2015, the dataset (Ben-Shimon, et al., 2015) was given for both clicks and purchases. Table 1.6 is a piece of clickstream data from the provided dataset; it records all the clicks happened on each item in a session. Some enhancement work such as Kim05Rec (Kim, Yum, Song, & Kim, 2005), Kim11Rec (Kim & Yum, 2011), Chen13Rec (Chen & Su, 2013), has been done on recommendation systems by integrating information from clickstreams to improve the recommendation accuracy.

Table 1.6: A piece of clickstream data from ACM 2015 RecSys challenge (Ben-Shimon et al., 2015)

sessionId	timestamp	itemId	categoryId
1	2014-04-07T10:51:09.277Z	214536502	0
1	2014-04-07T10:54:09.868Z	214536500	0
1	2014-04-07T10:54:46.998Z	214536506	0
1	2014-04-07T10:57:00.306Z	214577561	0
2	2014-04-07T13:56:37.614Z	214662742	0
2	2014-04-07T13:57:19.373Z	214662742	0
2	2014-04-07T13:58:37.446Z	214825110	0
2	2014-04-07T13:59:50.710Z	214757390	0
2	2014-04-07T14:00:38.247Z	214757407	0
2	2014-04-07T14:02:36.889Z	214551617	0

Weighted frequency pattern mining. Based on the basic frequent pattern mining algorithms such as Apriori (Agrawal, Srikant, & others, 1994), FP-growth (Han, Pei, & Yin, 2000) and Equivalence Class Transformation: Eclat (Zaki, 2000), more features have been added onto the patterns such as weight assigned to each item in addition to support. A few approaches have been proposed regarding to this topic such as WFIM in (Yun & Leggett, 2005) which takes the maximum and minimum weights into calculation, WAF in (Yun & Ryu, 2011) for tolerating noisy environment effects, and MWFIM in (Yun, Shin, Ryu, & Yoon, 2012) finding the maximal weighted frequent patterns.

In this chapter, we will introduce a few recommendation methods such as collaborative filtering, content-based recommendation systems, customer behaviors in e-commerce, recommendation systems in e-commerce, and some data mining techniques used in recommendation systems. In the end, we will give the thesis contributions and thesis outline.

1.1 Recommendation System, and Techniques

1.1.1 Some Recommendation Systems

Recommendation systems are widely used in different areas with different techniques; Table 1.7 shows some famous websites and their recommendation methods.

Table 1.7: Example recommendation systems in E-commerce

Websites	Website Type	Recommendation Techniques
Amazon.com	E-commerce	Item-item collaborative filtering
Netflix.com	Movie	Item-based collaborative filtering
Youtube.com	Videos	Content-based filtering
Movielens.com	Movie	Collaborative filtering
News.google.com	News/Articles	Collaborative filtering and information learning mechanism
Facebook.com	Social media	Graph-based filtering

1.1.2 Collaborative Filtering Recommendation System

“Collaborative filtering uses the known preferences of a group of users to make recommendations or predictions of the unknown preferences for other users” in (Su & Khoshgoftaar, 2009) defined collaborative filtering. The input for collaborative filtering is

a user-item rating matrix R where each value is a user's rating on an item, for example, a matrix with m users $\{u_1, u_2, \dots, u_m\}$ and n items $\{i_1, i_2, \dots, i_n\}$, r_{ui} is the rating of user u on item i and it can be unknown. The user-based neighborhood model CF (Aggarwal, 2016) would predict the unknown ratings (r_{uj} of item j for user u ,) through the following steps: (1) computing the mean rating for each user u ; (2) calculating the similarity between target user u and all other users v ; (3) computing user u 's peer group P ; and then (4) predicting rating for target user u for item j .

There are two popular methods for collaborative filtering, user-based and item-based. User-based collaborative filtering takes the ratings from similar users of the target user whereas item-based collaborative filtering considers the ratings from similar items of the target item as a preference. We will explain user-based collaborative filtering algorithm in Algorithm 1.1.

Algorithm 1.1: User-based Collaborative Filtering

Input: A user-item rating matrix $[(r_{11}, r_{12}, \dots, r_{1n}), (r_{21}, r_{22}, \dots, r_{2n}), \dots, (r_{m1}, r_{m2}, \dots, r_{mn})]$

Output: Predictions for unknown ratings r_{ui} ;

- (1) Compute the mean ratings \bar{r}_u ($\bar{r}_1, \bar{r}_2, \dots, \bar{r}_m$) for each user using Equation 1.1 where r_{ui} is the observed rating of user u for item i , only consider the specified ratings;

Equation 1.1: Mean rating computation

$$\bar{r}_u = \frac{\sum_{i \in I} r_{ui}}{|I|}, i \in I, r_{ui} \neq 0$$

- (2) Calculate the similarity ($s_{u1}, s_{u2}, \dots, s_{um}$) between user u and other users using methods such as cosine similarity (Sarwar, Karypis, Konstan, & Riedl, 2001) in Equation 1.2 while vector $u = \{r_{u1}, r_{u2}, r_{u3}, \dots, r_{un}\}$, and vector $v = \{r_{v1}, r_{v2}, r_{v3}, \dots, r_{vn}\}$, or correlation-based similarity (Sarwar, Karypis, Konstan, & Riedl, 2001) in Equation 1.3 where I donates common rated items of u and v , and some other methods such as Conditional Probability-based Similarity (Sarwar, Karypis, Konstan, & Riedl, 2001).

For each empty rating from user u on product i , predict rating r_{ui} .

Equation 1.2: The formula of cosine-based similarity

$$sim(u, v) = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|_2 * \|\vec{v}\|_2} = \frac{r_{u1} \cdot r_{v1} + r_{u2} \cdot r_{v2} \dots + r_{un} \cdot r_{vn}}{\sqrt{r_{u1}^2 + r_{u2}^2 + \dots + r_{un}^2} \cdot \sqrt{r_{v1}^2 + r_{v2}^2 + \dots + r_{vn}^2}}$$

Equation 1.3: The formula of correlation-based similarity

$$sim(u, v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$

(3) Compute user u's peer group U(i) for item i as follows:

- Compute the peer group of the target user as the set of k users having the highest similarity with the target user. Or find the closest k users for the target user separately for each predicted item such that each of these k users have specified ratings for that item.
- Find the predicted rating for each item of target user i as the weighted average of these k closest ratings for the item. Here, rating is weighted with the similarity of its owner user to the target user.
- To minimize the effect of observed ratings from users on varying scales, the raw ratings need to be mean-centered in row-wise fashion before finding the weighted average rating of the peer group.
- The mean-centered rating S_{ui} of a user u for item i is defined by subtracting her mean rating from the raw rating r_{ui} .

$$S_{ui} = r_{ui} - \bar{r}_u$$

- Then, the mean rating of the target user is added back to this prediction to provide a row rating prediction r_{ui} for target user u for item i.
- $U(i)$ = the set of k closest users to target user u, who have specified ratings for item i.

(4) Compute the predicted rating r_{ui} for target user u for item i using Equation 1.4 or Equation 1.5; the mean-centered is more relative.

Equation 1.4: The prediction formula of weighted sum

$$r_{ui} = \frac{\sum_{w \in U} sim(u, u') \times r_{wi}}{\sum_{w \in U} |sim(u, u')|}$$

Equation 1.5: The prediction formula of mean-centered ratings

$$r_{ui} = \bar{r}_u + \frac{\sum_{w \in U} sim(u, u') \times (r_{wi} - \bar{r}_w)}{\sum_{w \in U} |sim(u, u')|}$$

To explain the collaborative filtering algorithm, we use a rating table in Table 1.8 to calculate the rating of user “7” on item “e”, the steps are as follows:

Table 1.8: An example of a user-item rating table

userId\products	a	b	c	d	e
1	2		3		5
3	3	1	5		
4	1			3	4
7	2	4	1	1	?

(1) The average ratings for user 1, 3, 4 and 7 are $10/3$, 3, $8/3$, 2 respectively.

(2) The cosine similarity between user “1” and user “7” is $sim(1,7) = \frac{2 \times 2 + 3 \times 1}{\sqrt{2^2 + 3^2} \times \sqrt{2^2 + 1^2}} = 0.868$ following Equation 1.1. Similarly $sim(3,7) = 0.553$, $sim(4,7) = 0.707$;

The correlation similarity between user “1” and “7” is $sim(1,7) = \frac{(2 - \frac{10}{3}) \times (2 - 2) + (3 - \frac{10}{3}) \times (1 - 2)}{\sqrt{(2 - \frac{10}{3})^2 + (3 - \frac{10}{3})^2} \times \sqrt{(2 - 2)^2 + (1 - 2)^2}} = 0.245$. Similarly, $sim(3,7) = -0.878$, $sim(4,7) = -0.196$.

(3) Find top-N most similar users for user “7”, where N is set as 2 in this case, according to the cosine similarity result, U would be (1,4).

(4) Compute the rating value $r_{7,e}$ using cosine similarity results by Equation 1.3,

$$r_{7,e} = \frac{sim(1,7) \times r_{1,e} + sim(4,7) \times r_{4,e}}{|sim(1,7)| + |sim(4,7)|} = \frac{0.868 \times 5 + 0.707 \times 4}{0.868 + 0.707} = 4.55 \quad ; \quad r_{7,e} = \bar{r}_7 + \frac{sim(1,7) \times (r_{1,e} - \bar{r}_1) + sim(4,7) \times (r_{4,e} - \bar{r}_4)}{|sim(1,7)| + |sim(4,7)|} = \frac{0.868 \times (5 - \frac{10}{3}) + 0.707 \times (4 - \frac{8}{3})}{0.868 + 0.707} = 1.517$$

if using the weighted sum including other’s ratings in Equation 1.4.

There are some known fundamental issues with collaborative filtering as follows: (1) cold start: when new items or new users appear in the database, these items are not rated yet by any users, and the users' preferences are unknown; (2) sparsity issue: When the known rating data takes only a very small proportion in the user-item rating matrix, for instance, the amount of products is usually billions in the real world and most of the users only purchased probably hundreds of them, which leads to confusing and compromised

recommendations; (3) scalability issue: As the numbers of users and products grow rapidly, the time complexity and space complexity issues become more prominent.

1.2 Understanding Customers in E-commerce Business

The traditional form of commerce such as shopping in stores, consuming in restaurants, purchasing in malls has been cloned to the internet as the mobile and computer technology developing. E-commerce is not like the business happening in physical stores; a customer does not have any friend or shop assistant around to discuss her purchase decisions when shopping in online stores. Then online recommendation becomes necessary and important to provide users some guidance they may need, filtering products according to their shopping preferences, and sorting products referring to user's shopping habits. To improve user experience during online shopping from recommendation systems, mining data from user's explicit (such as explicit ratings, text comments) and implicit (behaviors such as purchase history, browsing history, search patterns, etc.) feedback became a hot topic for researches (Sivapalan, Sadeghian, Rahnama, & Madni, 2014).

1.2.1 Behavior Analysis

Hedonic versus utilitarian. Hedonic shopping was defined as fun shopping, which fulfills mental satisfactions, it usually refers to unnecessary products but what customers can gain emotional pleasures from. Whereas utilitarian shopping is more practical and direct, it was referred to as task-oriented and shopping for necessary products. Different value for hedonic and utilitarian shopping has been researched in (Babin, Darden, & Griffin, 1994), (Babin & Darden, 1995) and (Jones, Reynolds, & Arnold, 2006).

Discovery versus efficiency. Two scenarios for shopping was introduced in (Picot-Clémente, Bothorel, & Lenca, 2014). Discovery mode is when a user wanders around, check items and deals to fill her curiosity, impulsive purchases are more likely to be produced in this mode. Efficiency scenario is purpose-oriented purchasing; the user checks offer for the desired products and make a decision quickly.

Strategy typology. A typology table of shopping strategies was given in (Moe, 2003) as Table 1.9 composed of directed buying, search and deliberation, hedonic browsing and knowledge building in dimensions of search behavior and purchase horizon. Direct buying the efficiency scenario in (Picot-Clémente, Bothorel, & Lenca, 2014); search/deliberation

shares the same features with direct buying but the timing when the purchase happens, it is goal-oriented but focused on the planning phase; hedonic browsing is similar as described in (Jones, Reynolds, & Arnold, 2006); knowledge building explores the relevant information on a specific field such as electronics for future purchases.

Table 1.9: Typology of Shopping Strategies

Purchasing horizon	Search Behavior	
	Directed	Exploratory
Immediate	Directed buying	Hedonic browsing
Future	Search/deliberation	Knowledge building

Contextualized intentions. Based on the context, consumers' intentions were organized into five types in (Shi & Ghedira, 2016). Repurchase and planned purchase are goal-oriented; researching reflects users' interests and lack of knowledge; comparing shows users' different preferences like low price, fast shipping; exploring happens when consumers try to substantiate a vague for a known purpose; wandering is lack of purpose but seeking for pleasure shopping.

Observable intentions. Behaviors Intentions was categorized in (Shi & Ghedira, 2017) as research shopping intention, comparative browsing intention, idea searching intention and hedonic intention. With consumer data, the paper was proven able to identify different intentions.

1.2.2 Valuable Data in E-commerce

A taxonomy of input data for E-commerce recommendation systems was given in (Wei, Huang, & Fu, 2007) as Table 1.10 with explanations. We will specifically introduce clickstream data lying in behavior pattern data and transaction data.

Table 1.10: A taxonomy of input data

Data type	Explanation
Demographic data	Name, age, gender, profession, birthdate, telephone, address, hobbies, salary, education experience and so on.
Rating data	Rating scores, such as discrete multi-levels ratings and continuous rating; and latent comments, such as best, good, bad, worse and so on.
Behavior pattern data	Duration of browsing, click times, the links of webs; save, print, scroll, delete, open, close, refresh of webs; selection, edition, search, copy, paste, bookmark and even download of web content and so on.
Transaction data	Purchasing date, purchase quantity, price, discounting and so on.
Production data	For movies or music, it means actor or singer, topic, release time, price, brand and so on, While for webs or documents, it means content description using keywords, the links to others, the viewed times, the topic and so on.

Analyzing user's implicit preference is finding patterns from the user's behaviors like clicking, browsing, saving as a tag, etc. In E-commerce websites, the user's clicks, adding products to shopping cart and purchasing products makes a difference, and they can be calculated with different weights. Whereas in search engines, the user's clicks, the time that the user stays on the detail page, whether the user saves the result page as browser bookmark or not, all of the behaviors can be considered as valuable information. The possible actions of users are summarized by a few engineers from IBM (Zhao & Chun-e, 2011) as in Table 1.11:

1) Transaction Data

Transaction data is the detail purchase history of customers. The typical transactional database schema in (Agrawal & Srikant, 1995) is in Table 1.12. For each transaction, customer id, transaction time, and a set of items bought by the customer are recorded on by the company.

Table 1.11: User's Implicit Preferences from Activity

Activity	Type	Feature	Functionality
Rating	Explicit	Value of Integer in [0,n]	Get the user's preference from the rate that the user gives to the item
Vote	Explicit	Value of Boolean in (0,1)	Get the user's preference from the vote that the user gives to the item
Forward	Explicit	Value of Boolean in (0,1)	Get the sender's preference
Save as Bookmark	Explicit	Value of Boolean in (0,1)	Get the user's obvious interest from the bookmark
Tag	Explicit	Tag with words	Get the user's preference by analyzing the user's labels
Comment	Explicit	Text in words	Get the user's preference by analyzing the user's comments
Click Traffic	Implicit	A group of clicks from a user showing the user's interests.	Get the user's preference by analyzing the user's interests
Staying Time	Implicit	A group of time that the user staying in the detail pages.	Get the user's preference by analyzing the user's concentration
Purchase	Implicit	Value of Boolean in (0,1)	Get the user's obvious interest in the transaction records

Table 1.12: An example of transactional purchase database

Customer Id	TransactionTime	Items Bought
1	June 25 '93	30
1	June 30 '93	90
2	June 10 '93	10, 20
2	June 15 '93	30
2	June 20 '93	40, 60, 70
3	June 25 '93	30, 50, 70
4	June 25 '93	30
4	June 30 '93	40, 70
4	July 25 '93	90
5	June 12 '93	90

Different companies have different transaction table schema to store the records, session-based clickstream data schema in Table 1.13, where each row records a purchase occurred in a session, and there may be multiple products for each purchase, each record belongs to a specific user.

Table 1.13: A session-based historical transaction table

SessionId	UserId	Purchases
1	1	2
2	1	2,3
3	2	1,2,4
4	2	2,4,4
5	3	1
6	3	

2) Rating Data

Rating data is the basic data unit for collaborative filtering recommendation systems, is also one of the explicit feedback from user. Ratings options usually are given as Boolean or a numeric scale. There is no obvious ratings in E-commerce, but purchases can positively reflect the interest on an item from a user.

Ratings in boolean. Rating options provided by some websites simply are like/dislike (Table 1.14), where the ratings can be transferred to Boolean as 1 for “Like” and 0 for “Dislike.”

Table 1.14: An example of user-item rating table for a movie site

userId\products	Shrek	Snow White	Spider-man	Super-man
Alice	Like	Like	?	Dislike
Bob	?	Like	Dislike	Like
Chris	?	Dislike	Like	?
Tony	Like	?	Dislike	?

Ratings in scale. Rating options are a numeric scale (Shardanand & Maes, 1995) such as Table 1.15, where different numbers represent a different level of likeness.

Table 1.15: An example of scale rating schema

Score	1	2	3	4	5
Opinion	Strongly dislike	Dislike	Neutral	like	Strongly like

Ratings in unary. Some rating method such as Facebook only supports “Like,” so the rating values would be 1s and empty.

Rating data in E-commerce. However, in E-commerce, rating table (Table 1.16) as indicated in (Sarwar, Karypis, Konstan, & Riedl, 2000) is “the input data is a collection of historical purchasing transactions of n customers on m products. It is usually represented as a $m \times n$ customer-product matrix, R , such that $r_{i,j}$ is one if the i th customer has purchased the j th product, and zero, otherwise.” In item-based CF (Linden, Smith, & York, 2003), each vector corresponds to an item rather than a customer, and the vector’s M dimensions correspond to customers who have purchased that item.

Table 1.16: An example of rating data in E-commerce

Customer\Item	1	2	3	4
1	?	1	1	?
2	1	1	?	1
3	1	?	?	?

3) Clickstream Data

Clickstream data was defined as “the electronic records of Internet usage recorded by company web servers and syndicated data services” in (Bucklin, et al., 2002). Another explanation was given in (Bucklin & Sismeiro, 2009) as “the electronic record of Internet usage collected by web servers or third-party servers” and “the electronic record of a user’s activity on the Internet.” Clickstream data has been used to predict next request (Gündüz & Özsu, 2003), discover patterns to build profiles for customers (Park & Chang, 2009), find the possibilities of purchasing items (Van den Poel & Buckinx, 2005), etc. In e-commerce, clickstream data reflects a user’s Internet footprints for behaviors such as clicks, basket placement, purchases, reading reviews and so on. A session-based clickstream data example is given in Table 1.17 simply shows a few main attributes of click events, more detailed attributes such as visit duration, visit types, IP address may also be included in clickstream data.

Table 1.17: Session-based clickstream data

SessionId	UserId	ItemId	Category	Time
1	1	1	1	2014-04-0211:44:11
1	1	2	1	2014-04-0211:46:37
2	1	3	3	2014-04-0811:15:35
2	1	5	3	2014-04-0811:18:23
2	1	2	1	2014-04-0811:21:12
2	1	3	3	2014-04-0811:23:44
3	2	2	1	2014-04-2511:16:14
3	2	1	1	2014-04-2511:19:47
3	2	4	2	2014-04-2511:23:07
...				

How to get clickstream data? Some common ways of capturing clickstream data were given in (Bucklin & Sismeiro, 2009). Web server log maintained by a website can record all the internet communications, this also called “in-site” or “site-centric” because all the clickstream data are inside of a website. Internet service provider (ISP) can capture clickstream data for multiple websites by recording and analyzing all the requests sent out by a user from the local computer; this approach is “user-centric” which is user-oriented.

What is valuable in clickstream data? Most systems use the three indicators from clickstream data, which are browsing path, browsing frequency and visit duration. Browsing path is the web page click sequence, whereas frequency is the times where a user visiting the same product or category during a session, and duration is the time length the user spent on a product or category in a session.

The difference from web server log. “Web server log file is a log file automatically created and maintained by a server of activity performed by it; it contains information about the request, client IP address, request date/time, the page requested, HTTP code, bytes served, user agent, and referrer (Rathipriya & Thangavel, 2011).”

4) Meta Data

Meta data is the description for users (such as Figure 1.2 showing what categories a user prefers for online shopping) and products (such as Figure 1.1 showing some attributes such as weight, color and reference code of a product) stored in database tables respectively. But instead of using this information for displaying purpose on the webpage, this information also can be used for recommendation engines, such as comparing products and recommending similar products.

OS	iOS
Item Weight	136 g
Product Dimensions	13.8 x 6.7 x 0.7 cm
Batteries:	1 Lithium Polymer batteries required.
Manufacturer reference	a1778
Weight	136 grams
Colour	Black

Figure 1.1: Product metadata of iPhone 7 on Amazon

- | | |
|---|---|
| <input type="checkbox"/> CA_Corporate_Subscription | <input type="checkbox"/> General Offers |
| <input checked="" type="checkbox"/> Automotive | <input type="checkbox"/> Home & Garden |
| <input type="checkbox"/> Baby | <input type="checkbox"/> Industrial & Scientific |
| <input checked="" type="checkbox"/> Beauty | <input type="checkbox"/> Luggage & Bags |
| <input type="checkbox"/> Grocery | <input checked="" type="checkbox"/> Movies & TV |
| <input type="checkbox"/> Health & Personal Care | <input checked="" type="checkbox"/> Music |
| <input checked="" type="checkbox"/> Sports & Outdoors | <input type="checkbox"/> Prime Music |
| <input checked="" type="checkbox"/> Tools & Building Supplies | <input checked="" type="checkbox"/> Prime Student |
| <input type="checkbox"/> Amazon Family | <input type="checkbox"/> Seller Communications |
| <input type="checkbox"/> Amazon Partners | <input type="checkbox"/> Seller Feedback |
| <input checked="" type="checkbox"/> Associates | <input checked="" type="checkbox"/> Software |
| <input type="checkbox"/> Books | <input type="checkbox"/> Surveys & Feedback |
| <input checked="" type="checkbox"/> Customer Surveys | <input type="checkbox"/> Toys & Games |
| <input checked="" type="checkbox"/> Electronics | <input type="checkbox"/> Video Games |

Figure 1.2: User category preference meta data on Amazon

1.3 Recommendation Systems in E-commerce

Recommendation systems have been integrated into all kinds of business since the 1990s and proven very helpful in improving the number of sales in E-commerce area.

Business goals. In addition to the common goals of recommendation systems, (Schafer, Konstan, & Riedl, 2001) gave five important goals for recommendation systems in E-commerce as follows:

- (1) Helping new and infrequent visitors: broad recommendation lists. Every customer is a potential asset especially those who just started playing in the area, making good and right recommendations can convert those customers to permanent customers;
- (2) Building credibility through the community: customer comments and ratings. E-commerce sites are customer-oriented, and the companies are always looking for stable consumers who trust them and loyal to them, therefore getting new customers through community and keep customers by showing them credibility is important;
- (3) Inviting customers back: Notification services. When new products or good deals come in, notify customers to come visit on a regular basis to make them interested;
- (4) Cross-selling: Product-associated recommendations. Better than physical stores which probably only have some specific type of products, online retailers are made to provide a wider selection, therefore mining the relationships between products which associate products become important;
- (5) Building long-term relationships: Deep personalization. A good recommendation system knows customers, gives them exactly what is needed, the service of providing convenience and accurate recommendations keeps customers, so knowing customers better leads to a longer-term relationship with them.

Research challenges. Given the huge transaction dataset in E-commerce and the fact that they are built to get customers and make money, there are some specific challenges (Schafer, Konstan, & Riedl, 2001) for E-commerce recommendation systems:

- (1) Scalability and real-time performance for the huge dataset and better user experience ;

- (2) Incorporating rich data. All the input from a customer on a website expresses the customer's interest, making the best use of them with the correct mining algorithms improves the accuracy of recommendations;
- (3) Consumer-centered recommendations;
- (4) Connecting recommenders to marketers.

1.4 Data Mining

Data mining aims at exploring implicit information or patterns hidden in the large data set, but not like gamblers betting odds while playing games, or people finding easily recognizable patterns during daily life. The definition given in (Chen, Han, & Yu, 1996) was “Data mining, which is also referred to as knowledge discovery in databases, means a process of nontrivial extraction of implicit, previously unknown and potentially useful information (such as knowledge rules, constraints, regularities) from data in databases”. We will introduce some basic data mining techniques in this section.

1.4.1 Association rule mining

Association rule mining is a method for discovering interesting relations between item sets $X \rightarrow Y$ hidden underneath of data, expressing that when a transaction T contains X , T probably also contains Y (Hipp, Güntzer, & Nakhaeizadeh, 2000). For input data a large item set $I = \{I_1, I_2, I_3, \dots, I_n\}$, where each item is an item, and a transaction database $T = \{t_1, t_2, t_3, \dots, t_m\}$ where t_i is a transaction, there are one or more items involved in each transaction, and one item can be purchased more than one in each transaction. The method usually contains two steps; the first step is finding all the frequent itemsets with occurrence greater or equal than a minimum support threshold, and then generate association rules such as $X \rightarrow Y$ for item set pairs with confidence (the probability of Y happens when X happens) being greater or equal than minimum confidence. The Apriori algorithm (Agrawal, Srikant, & others, 1994) is a popular algorithm for association rule mining; it finds the set of frequent patterns (large itemsets, L_i) iteratively by computing the support of each itemset in the candidate set C_i . During each i th iteration, it finds the i th large itemsets L_i from C_i , before computing the next $(i+1)$ candidate set C_{i+1} using L_i apriori-gen join on L_i . It then, prunes itemsets from C_{i+1} which have any subset that is not already large. The process terminates when either a C_i or L_i is an empty set.

Table 1.18: An example table for Apriori frequent pattern mining

tid	items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Run the Apriori algorithm with the data in Table 1.18 and minimum support being 2. First find the $L_1 = \{1:2, 2:3, 3:3, 5:3\}$ after pruning $(4:1)$ where 1 is the support for item “4”; apply Liapriori-gen join on L_1 and L_1 to get $C_2 = \{(1,2), (1,3), (1,5), (2,3), (2,5), (3,5)\}$. Then prune the sets in C_2 where the support is less than 2 and use the rest for $L_2 = \{(1,3):2, (2,3):2, (2,5):3, (3,5):2\}$. Apply Liapriori-gen join on L_2 and L_2 to get $C_3 = \{(2,3,5)\}$, $L_3 = \{(2,3,5):2\}$. Stop the iteration and form frequent pattern $L = L_1 \cup L_2 \cup L_3 = \{(1), (2), (3), (5), (1,3), (2,3), (2,5), (3,5), (2,3,5)\}$.

1.4.2 Clustering

Clustering is a method grouping objects into subsets, where the objects in each subset share some similar patterns (observations, data items, or feature vectors) (Jain, Murty, & Flynn, 1999), clustering is unsupervised, which does not need to be labeled manually. The input data is a set of objects $O = \{O_1, O_2, O_3, \dots, O_n\}$, and each object has n-dimensional attributes $O_i = \{D_1, D_2, D_3, \dots, D_m\}$, $1 \leq i \leq n$, and the output data is some subsets of objects such as $[\{O_1, O_4\}; \{O_2, O_6, O_3\} \dots]$. A popular algorithm is k-means clustering (Hartigan & Wong, 1979) which groups all the objects to k clusters by following steps:

- (1) Randomly pick k object $C = \{C_1, C_2, C_3, \dots, C_k\}$ where C_i belongs to O, as seeds and each object is considered as a centroid for the cluster.
- (2) For each object O_i in O, $1 \leq i \leq n$, comparing the distance between O_i and O_j in C, find the closest cluster centroid O_j and reassign the object O_i to this cluster.
- (3) Recalculate the centroid C for each new cluster, by computing the average attributes of all object in a cluster.
- (4) Repeat step 2 and step 3 until the centroids C stop changing.
- (5) Return the k clusters

For example, there are five objects in Table 1.19, and there are two attributes for each object, to run a 2-means clustering algorithm. Firstly, pick two objects as centroid for the two clusters such as {2, 3}, then we now have two clusters with vector (5.0, 7.0) and (1.5, 2.0) being centroid respectively. Then for each object, compare the Euclidean distance between it and each centroid, assign (reassign) this object to the closest cluster with shorter distance. Then calculate the average attributes for each vector as a new centroid, and reassign objects again, until the centroids stop changing. The clusters are in the Cluster column, where (2.0, 3.0) and (5.0, 7.0) are the final centroids for each cluster.

Table 1.19: Example data for clustering

Objects	Attribute A	Attribute B	Cluster
1	3.0	4.0	Cluster 1 (2.0, 3.0)
3	1.0	5.0	
4	1.5	2.0	
5	3.5	1.5	
2	5.0	7.0	Cluster 2 (5.0, 7.0)

1.4.3 Classification

For a set of objects $O=\{O_1, O_2, O_3, \dots, O_n\}$, and each object have n-dimensional attributes $O_i=\{D_1, D_2, D_3, \dots, D_m\}$, $1 \leq i \leq n$, but there is one attributes D_k in charge of grouping the objects which is called class attribute. Using a classifier built by discovering the relationship between other attributes and the class attributes to group future objects (which have all some attributes but the class attribute) to the right class is called classification. Decision tree (Anyanwu & Shiva, 2009) is forming a decision tree to cover all the training data, where its leave nodes are classes, and other nodes are non-class attributes. Then to classify a new object, apply the object to the decision tree and get the class from the leaf node.

Use the data in Table 1.20, a decision tree can be built like Figure 1.3: Decision tree example, when a new object comes such as (Objects: 6, Attributes A: low, Attribute B: high) when we can classify this new object to ClassTwo according to the decision tree.

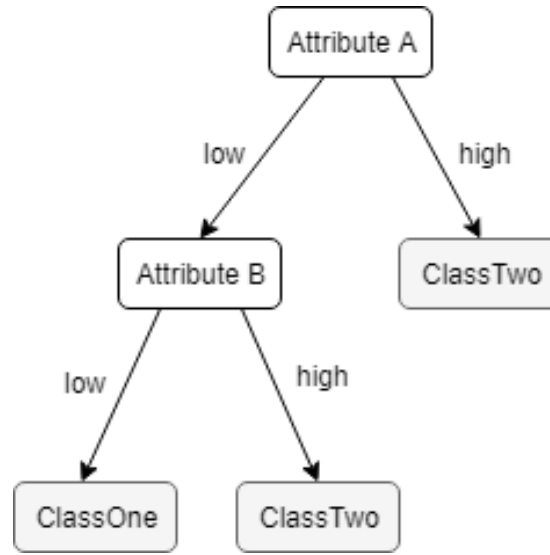


Figure 1.3: Decision tree example

Table 1.20: Example data for classification

Objects	Attribute A	Attribute B	Class
1	low	low	ClassOne
2	high	high	ClassTwo
3	low	high	ClassOne
4	low	low	ClassTwo
5	high	low	ClassTwo

1.5 Existing Recommendation Systems Integrated with Clickstream Data

Over the years, based on the relevant techniques and research ((Agrawal, Imieliński, & Swami, 1993), (Agrawal, Srikant, & others, 1994), (Babin, Darden, & Griffin, 1994), (Agrawal & Srikant, 1995), (Babin & Darden, 1995), (Berry & Linoff, 1997), (Ungar & Foster, 1998), (Schafer, Konstan, & Riedl, 2001)) proposed during the 1990s, many companies and researchers have been improving recommender methods and systems. Methods for enhancing input data have been studied in various papers, such as (Kim, Im, & Atluri, 2005), (Kim & Yum, 2011) and (Chen & Su, 2013), since the user-item rating matrix in e-commerce only shows what items a user has purchased previously, which does not provide a lot of information about customer purchase history or item purchase history for the purposes of improving recommendation accuracy. In addition to the rating matrix,

some other data sources such as clickstream data, metadata and transactions have been discovered and utilized to improve recommendations. Clickstream data has been used to predict a user's next request (Gündüz & Özsu, 2003), discover patterns to build profiles for customers (Park & Chang, 2009) find the possibilities of purchasing items (Van den Poel & Buckinx, 2005), etc.

Traditional collaborative filtering recommendation systems in E-commerce take user-item purchase ratings as input and generate recommendations. For improving recommendation accuracy and make better recommendations, clickstream data has been integrated into some recommendation systems such as Chen13Rec (Chen & Su, 2013), Kim05Rec (Kim, Yum, Song, & Kim, 2005), Kim11Rec (Kim & Yum, 2011) in the diagram in Figure 1.4. The comparison of them is in Table 1.21.

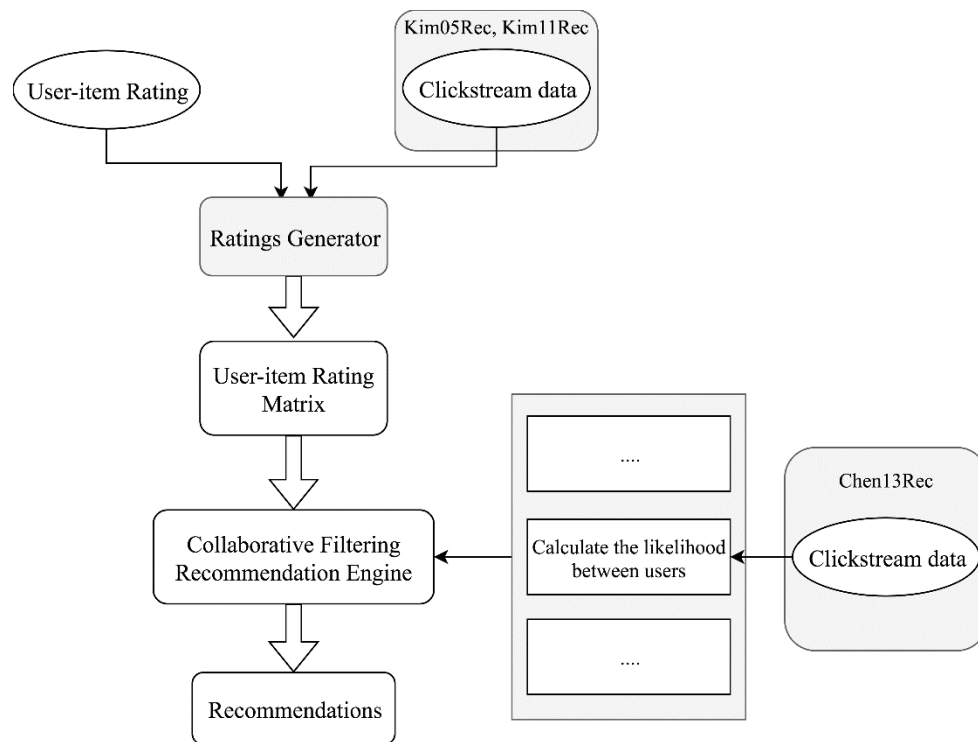


Figure 1.4 CF RecSys diagram integrated with clickstream data

Kim05Rec (Kim, Yum, Song, & Kim, 2005) analyzes the existing customer behavior data such as searching, browsing, clicking, basket placement and purchasing to build a basket placement decision tree. The input data contains a list of sessions, where each

session contains a series of behaviors, the output decision tree has functionality such as given a session and some behaviors but without basket placement or purchase behaviors, the decision tree will output a probability of this user in the session placing the product into the basket. Then the system fills the user-item matrix with the possibilities, uses CF to predict best products on the new matrix.

The proposed algorithm calculates the possibility of placing a product into the basket, and use the result to enrich the user-item rating matrix, the new matrix is more informative and therefore gives better and more accurate recommendations. Nevertheless, it only takes the products which have been previously clicked by the customer as candidates, where using collaborative techniques would make it more general. It also uses the statistical data nesting in the decision tree to find the possibilities; this method has compromised the relationship information between products by only using the average data as a reference.

Kim11Rec (Kim & Yum, 2011) integrated with association rule mining calculates the confidence between products in different stages like click, basket placement, and purchase happened in sessions. Such as for the stage of click, clicks (<abc>,<bcd>,<efa>) have happened in some existing sessions, and there are some items such as <ad> that a user has shown interest in current session, then the system finds the most relevant clicked item in (<abc>,<bcd>,<efa>) for a, and for b, then chose top-N with higher lift score. Same for basket placement groups and purchase groups, then assign weight to these three scores and calculate a final score. The top-N scores can be verified as purchased to enrich the matrix. It was proved outperformed the decision tree approach, but association rule mining only considers the major cases, for infrequent users, it will be very hard to get qualified recommendations.

It was proved outperformed the decision tree approach, but by applying association rule mining which takes all the applicable cases into the calculation, it loses the connection between users sharing a special interest. E-commerce recommendation systems should care about infrequent users who may have infrequent visiting patterns; this can be improved with collaborative mining techniques.

Table 1.21: Comparison of existing some recommendation systems

Name	Core Method	Input Data	Limitation
Kim05Rec (Kim, Yum, Song, & Kim, 2005)	Form a decision tree on variables such as click type, visit times, visit duration to decide the possibility of putting a given product into basket. Then put the possibility into the binary rating matrix if the value is unknown. Use the new matrix to predict ratings.	Input: clickstream data, binary rating matrix Considered variables: search type, browsing type, visit times, the ratio of going into next stage (stages: click, basket, purchase), and basket placement	Only focuses on products that have been clicked, so it is impossible to give recommendations on new products, and some information gets lost while building the decision tree using probabilities.
Kim11Rec (Kim & Yum, 2011)	Integrates with association rule mining calculates the confidence between products in different stages including click, basket placement, and purchase. For instance, the clicks (<abc>; <bcd>; <efa>) happened in the click stage for three different sessions, and there are some items such as a and b that a user has shown interest during current session, then the system finds the most relevant clicked item in (c; d; e; f) for a and b by comparing the lift score. Same for the basket placement and purchase stages. The next step is assigning weights to the scores of three different stages to calculate a final score.	Input: clickstream data, Purchase data Considered variables: clicks, basket placement, purchases	It only considers the major and popular cases by computing the support, for infrequent users, it will be very hard to get qualified recommendations.
Chen13Rec (Chen & Su, 2013)	It firstly uses the longest common subsequence comparing the two click sequence groups of two users; the second indicator is the similarity between user-product click frequency vectors showing the click times of a user for all products; the third indicator is the similarity between user-product visiting duration vectors. By selecting top-N similar users using three indicators, the CF method can use it for neighbor selection and improve the poor relationship between users in the rating matrix.	Input: clickstream data, binary rating matrix Considered Variables: Click category sequence, category click frequency, category click sum during.	It only focuses on the category level visits, and its technique for mining the whole dataset is not very efficient.
HPCRec	Use historical purchase to build a user-item support matrix, normalize the matrix and use it to replace the binary rating matrix. Match each session-based click sequences to a purchase, for those without a purchase, find some similar ones with purchase, then use the similarity as weight to find the weighted frequent items, and put the weighted frequency the rating matrix if rating is unknown. Predict ratings based on the enriched rating matrix.	Input: historical purchase data, clickstream data Considered Variables: click sequence; the consequential relationship between click sequence and purchase; purchase amount in the past;	There is more that can be extracted from the historical purchase data such as how frequently a user purchases a product, this can be used to enhance the quality recommendations.

Chen13Rec (Chen & Su, 2013) aims at finding the similarity between two users on their clickstream sequences to address a more accurate neighborhood with similar interest. This system partially operates on session data, for all click sequences happened in session for a user, compare that to all click sequences happened in another user by the length of the longest sub-common sequence of a sequence for the first user and another sequence from the second user, this will become the first indicator of the relationship between these two users. The second indicator is the similarity between user-product click frequency vectors; a vector shows the click times of a user for all the products; the third indicator is the similarity between user-product visiting duration vectors, just like the second indicator but the data is the total time a user spend on viewing a product. By selecting top-N similar users using clickstream data instead of purchasing data, it can use this for neighbor selection in collaborative filtering and improve the poor relationship between users in the rating matrix. However, they only focus on the category level visits, and its technique for mining the whole dataset is not very efficient.

Different from using purchase data, this approach proved that clickstream data contains information for cluster users by their browsing path. Nevertheless, the information mined from clickstream data is always confirmed by purchase data, so integrating purchase data would make a significant different on the recommendation accuracy. Moreover, this system requires specific domain knowledge for categories, and only supports category level recommendations, an enhancement on the generality can make it more useful and flexible.

HPCRec versus Kim05Rec. Firstly, HPCRec performs a collaborative session-based interest detection where similar sessions lead to similar purchase interest, whereas Kim05Rec assumes a larger proportion of choices (eg., a situation where 51% of customers choose not to buy a product and 49% of the customers who choose to buy it, then if a user has attributes fit to the 51% of the customers, but he would like to buy this product, the system would not recommend this product to him by following the choice of 51% of customers have made) should decide a user's known interest for a new product. Secondly, HPCRec improves the quality of ratings using historical purchase data before adding potential additional ratings to the sparse rating matrix and Kim05Rec does not. In addition, the session-based interest detection method makes HPCRec be able to provide

recommendations for infrequent users which Kim05Rec can not. The tables showing the rating quantity and quality will be in an example in section 3.3, Table 3.15 for Kim05Rec and Table 3.37 for HPCRec, from which we can see the enriched user-item rating matrix in HPCRec has better rating quantity and quality compared to the enriched matrix in Kim05Rec.

HPCRec versus Kim11Rec. Kim11Rec uses an association rule mining approach to find the closest product for a given one. If we say Kim11Rec only considers the vertical relationship which means activities (click, purchase) between different sessions, then HPCRec also considers the horizontal relationships such as different stages (click, purchase) in a session. HPCRec also improves the rating quality by integrating historical purchases in a way of considering purchasing quantity instead of simply using the binary user-item rating matrix . The tables showing the rating quantity and quality will be in an example in section 3.3, Table 3.22 for Kim11Rec (0.16,0.16,1,0.16 are the new ratings added into the user-item rating matrix) and Table 3.37 for HPCRec.

HPCRec versus Chen13Rec. Chen13Rec did not integrate purchase data; it identifies the common interest among users by their click patterns, frequencies, browsing durations in different categories. Based on this, first of all, we integrate historical purchased data to improve the rating quality; then we generalize the Chen13 method to an item level from category level; next we connect the similar items clicks with purchases happened in the same session and predict purchase possibilities for unknown interest in the user-item rating matrix; in the end, we use the enriched rating matrix for collaborative filtering system. The tables showing the rating quantity and quality will be in an example in section section 3.3, Table 3.27 shows the new relationship mined from clicking patterns for Chen13Rec, while applying the collaborative filtering algorithm, use the new user similarities in Table 3.27 to address peer groups; whereas Table 3.37 is the enriched user-item rating matrix for HPCRec. The evaluation result in Table 3.38 shows HPCRec performs better than Chen13Rec.

1.6 Thesis Contributions

Recommendation systems in E-commerce suffer from uninformative rating data which usually only represents if a user has purchased a product before, this user-item rating matrix

is usually sparse, less informative and lead to poor recommendations sometimes. we propose a new recommendation system (HPCRec) using purchasing history patterns to improve the rating quality for the matrix, and mining the consequential information between clicks and purchases to enhance the rating quantity in the matrix, then improves the recommendation accuracy, and meanwhile, HPCRec was also proven to be able to make better recommendations to infrequent users.

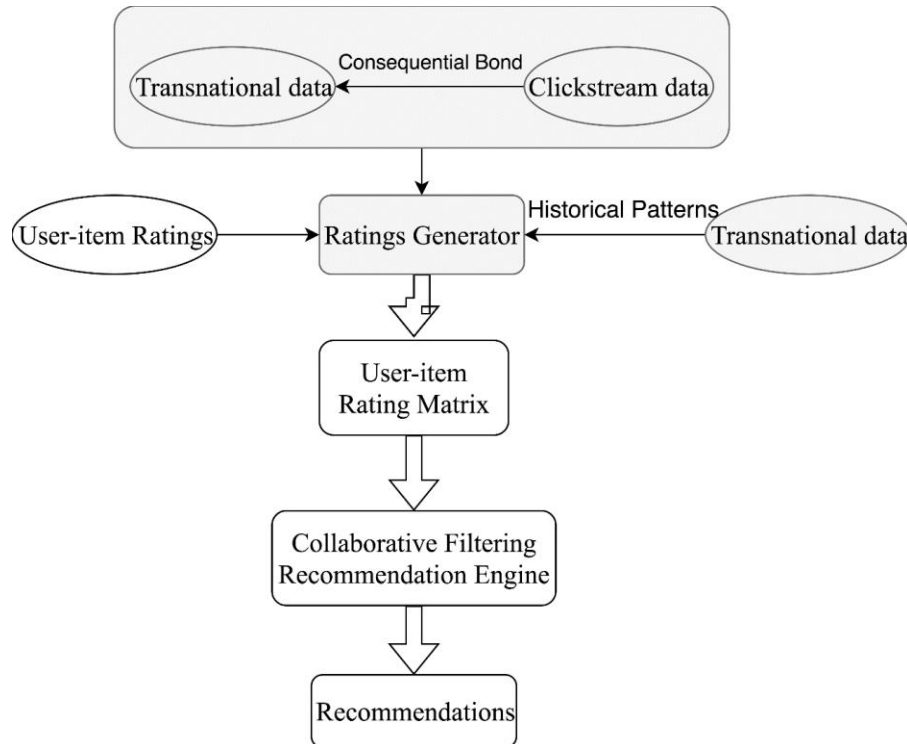


Figure 1.5: HPCRec in CF RecSys diagram

1.6.1 Observations and Thesis Hypotheses

Data Sparsity: Collaborative filtering in E-commerce suffers from data sparsity in the rating matrix given the huge amount of products. But with the sparse ratings, it only uses binary user-item rating purchase matrix (Table 1.16) which doesn't reflect much regarding: (1) how much a user likes an item; (2) how frequently or how long ago a user purchased an item; (3) what quantity of a product was purchased. This information is not integrated into the CF user-rating matrix but can potentially improve the recommendations accuracy.

Information Distribution: Traditional collaborative filtering only takes purchase data into the calculation, while there are other data available for analysis (section 1.2.2). Moreover,

from the data ACM RecSys 2015 (Ben-Shimon, et al., 2015) provided from recommendation systems, we can notice that the click data (yoochoose-clicks) in Figure 1.6 is almost 27 times more than the purchase data (yoochoose-buys).




 yoochoose-clicks Type: DAT File	Date modified: 11/5/2014 9:25 AM Size: 1.38 GB
 yoochoose-test Type: DAT File	Date modified: 11/5/2014 9:25 AM Size: 354 MB
 yoochoose-buys Type: DAT File	Date modified: 11/5/2014 9:25 AM Size: 53.0 MB

Figure 1.6: The data source of ACM code challenge in 2005

Consequential Relationship between Clicks and Purchases: Clickstream data (Table 1.17 has clicks for session (1,2,3), the click sequences for sessions (4,5,6) are (4,4,1,2), (1,2,1) and (3,5,2)) and the purchase data (Table 1.13) can be organized as in Table 1.22, where each record is a session with information such as which user logged into this session, what product this user has clicked and purchased during this session, we use product ids representing products in the table. Sometimes there are no purchases but only clicks in a session. We define this relationship as a consequential relationship because clicks lead to certain purchases.

Table 1.22: Sample consequential table

SessionId	UserId	Clicks	Purchases
1	1	1,2	2
2	1	3,5,2,3	2,3
3	2	2,1,4	1,2,4
4	2	4,4,1,2	2,4,4
5	3	1,2,1	1
6	3	3,5,2	

Thesis hypothesis 1: Improve the rating quality. The binary user-item rating matrix in Table 1.16 is less informative compared to the original transaction records in Table 1.13. If there

is a way to extract more information from the transaction records to the rating matrix, the recommendation system would perform better with the more informative input data. We first form a user-item purchase frequency matrix (Table 1.23) from Table 1.13, where each value represents the amount of a product purchased by a user. We then normalize the purchase frequency to a scaled value (0 to 1 in Table 1.24) representing how interested a user is in one item as compared to various others, the formula is introduced in chapter three.

Table 1.23: A user item purchasing frequency matrix

Customer\Item	1	2	3	4
1	?	2	1	?
2	1	2	?	3
3	1	?	?	?

Table 1.24: Normalized user-item purchase frequency matrix

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	?	?	?

Table 1.25: Enriched user-item purchase frequency matrix

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	1	0.19	0.167

Thesis hypothesis 2: Improve the rating quantity. In the sessions with purchases where purchases were made by a user in Table 1.22, the interest of the user for the purchased products is affirmative, whereas for the sessions without a purchase, we can not determine whether if a user is interested in a product. However, the clicks happened in a session without purchase may imply the potential interest. Kim05Rec (Kim, Im, & Atluri, 2005) uses a decision tree approach, but it only focuses on the major cases in the way of always choosing the popular path in the decision tree. In terms of predicting interest based on

clicks, Kim11Rec (Kim & Yum, 2011) uses an association rule approach which does not consider the consequential bond between clicked products and purchase products in the same session, neither does Chen13Rec (Chen & Su, 2013) which performs a category-based click sequence similarity, visit frequency and duration similarity between users to find the common interest. We mainly discover the session-based consequential bond to generate more potential rating scores. For example for session 6 where user 3 clicked products 3, 5 and 2 in Table 1.22, if we can find sessions sharing a similar click pattern but has purchased some product, then we can use the possibility to enrich the rating matrix such as Table 1.25 which is less sparse than the original table (Table 1.24). We assume by integrating the consequential information to the user-item purchase matrix, the accuracy of recommendations will be improved.

To conclude, Hypothesis 1 talks of using frequencies of item purchases to improve on quality of known ratings (from 1 to actual number totally purchased) by including quantity of the items purchased in the period under consideration. Hypothesis 2 talks of using the relationship between clicked items and purchased items in the clickstream sessions to improve on the quality of the unknown ratings from 0 to a value between 0 and 1 representing the possibility that item may be purchased.

1.6.2 Method Contributions

Studies in ((Sismeiro & Bucklin, 2004), (Bucklin & Sismeiro, 2009), (Moe & Fader, 2004)) have implied that there is a consequential relationship between behaviors collected as clickstream data and purchase data. We propose the HPCRec system, which enriches the rating matrix from both quantity and quality aspects, then processes the enriched matrix using the CF method. It takes the consequential table (eg., Table 1.22) and user-item purchase frequency matrix (eg., Table 1.23) as input, follows four main steps and output a rating matrix with predicted ratings:

- (1) Normalizing user-item purchase frequency matrix to a new user-item rating matrix, details in section 3.2.1.
- (2) In the consequential table, for each session without a purchase belonging to a user, find the top-N similar sessions with purchases by comparing the click sequences using function CSSM (Clickstream Sequence Similarity Measurement) in section

3.2.2. Then use the similarity as weight and assign to the purchases in the selected top-N session. This step generates a weighted transaction table where weights are similarities assigned to purchases.

- (3) Use the weighted transaction table from step 2, call function TWFI (Transaction-based Weighted Frequent Item) in section 3.2.3 and get a list of items with purchasing possibilities.
- (4) For each item from the previous step, if the user from step 2 has not previously purchased the product, enrich the result matrix from step 1 with the possibility. Then return to step 2 for the next session without a purchase if possible, and otherwise continue to step 5.
- (5) With the enriched rating matrix, run the CF algorithm and predict ratings. Return the rating matrix with predicted ratings.

1.6.3 Feature Contributions

In this thesis, we extend the existing methods of Kim05Rec (Kim, Im, & Atluri, 2005), Kim11Rec (Kim & Yum, 2011) and Chen13Rec (Chen & Su, 2013) by enhancing the input data and extracting more relational inner patterns for better recommendations. We firstly consider using the full historical transactional records instead of the binary user-item rating matrix, then from a session-based level, find interest for sessions without purchases by comparing the session behaviors to the ones with confirmed purchases, use the interest to enrich the input data for collaborative filtering algorithm. By proposing HPCRec system to do so, we intend to make following contributions:

- I. ***Improve the quality of ratings.*** By normalizing the user-item purchase frequency matrix using the method in (Weisstein, 2002), instead of using the binary rating matrix in Kim05Rec (Kim, Im, & Atluri, 2005) the normalized ratings can distinguish between the level of interest from one product to another.
- II. ***Improve the quantity of ratings.*** We define and use the consequential bond between clicks and purchases, for the sessions without a purchase but clicks, and find sessions with similar click patterns to calculate the consequential purchases from these clicks. Then use the consequential purchase possibility to improve the quantity of ratings. However, this information has not been considered by systems

such as Kim05Rec (Kim, Im, & Atluri, 2005), Kim11Rec (Kim & Yum, 2011) and Chen13Rec (Chen & Su, 2013).

- III. ***Improve the recommendation accuracy.*** By processing the enriched rating matrix generated in the CF algorithm, compared to the decision tree approach Kim05Rec (Kim, Im, & Atluri, 2005), the association rule approach Kim11Rec (Kim & Yum, 2011) and the category-based approach Chen13Rec (Chen & Su, 2013), the experimental results show that our approach HPCRec (Historical Purchase and Clickstream based Recommendation System) performs better.
- IV. ***Make recommendations for infrequent users.*** Our method HPCRec performs a task-based interest mining algorithm which does not find the most popular products for a user, but instead finds the interests of the most similar sessions compared to the existing sessions without a purchase for the user. Systems such as Kim05Rec (Kim, Im, & Atluri, 2005), Kim11Rec (Kim & Yum, 2011) and Chen13Rec (Chen & Su, 2013) are more focused on popular products and ignore the weak bond between infrequent users, where “popular” reflects products with a high click or purchase frequency.

1.7 Thesis Outline

Previously in the first chapter, we have introduced some basic concepts related to my thesis topic. We will present some recommendation systems that have been proposed on clickstream data and transaction data in chapter two. Based on a few observations, chapter three gives the proposed recommendation system addressing the problems. The system implementation and evaluation will be given in chapter four; the ideas for future work in this area are in chapter five.

CHAPTER 2

RELATED WORK

To improve the accuracy of recommendation systems caused by the user-item matrix transformed from transactional records being not informative enough. Different kinds of solutions have been proposed to address the problem in E-commerce. Some recommendation systems such as Amazon03Rec (Linden, Smith, & York, 2003) applies co-purchased to narrow down the candidates while reducing the noisy in the matrix; Kim05Rec (Kim, Yum, Song, & Kim, 2005) defines a possibility decision tree to find the possibility and prefill to the user-item matrix; Kim11Rec (Kim & Yum, 2011) utilizes association rule between items to calculate the confidence of co-clicked, co-purchased items; Chen13Rec (Chen & Su, 2013) measures the category-based similarity between two sequences to narrow down the candidates to specific categories; Fan14 (Fan, Pan, & Jiang, 2014) extracts category-based information from user-item matrix to refine the predicted values. We will introduce some relevant work in this area. Given that my thesis topic is mining information from clickstream sequences combining with purchase history, some basic sequential pattern mining techniques and a few methods measuring the relationship between sequences will also be represented.

2.1 E-commerce Recommendation Systems on Clickstream Data

Recommendation systems on clickstream data can be categorized to two groups, one uses different variables such as visiting path, visiting frequency to predict the purchase interests and preferences (Table 2.1); the other one is based on stages which calculate the consequential or conditional relationship between different stages to calculate the purchase possibility (2.2). Specifically, we will introduce Kim05Rec (Kim, Yum, Song, & Kim, 2005), Kim11Rec (Kim & Yum, 2011), which uses a stage-based approach, and Chen13Rec (Chen & Su, 2013) is a typical variable based approach.

Table 2.1: Literature review of clickstream data related work on variables

Artical	Website	Variables	Method
(Bucklin & Sismeiro, 2003)	Automotive E-commerce	<ul style="list-style-type: none">• Visit depth• Repeat visits	To predict whether a visitor continues browsing some features in current page by clicking which is also called within-site browsing, and estimate the length of time the visitor would spend on this webpage.
(Sismeiro &	Automotive	<ul style="list-style-type: none">• Browsing time	Conditional probability approach

(Bucklin, 2004)	E-commerce	<ul style="list-style-type: none"> • Repeat visit • Input effort 	
(Moe, 2003)	Nutrition products E-commerce	<ul style="list-style-type: none"> • Visit numbers • Duration 	Cluster store visitors to five categories by customer type
(Moe & Fader, 2004)	Amazon books store	<ul style="list-style-type: none"> • Purchase pattern • Visit pattern • Number of visits • Repeated visit • Purchase threshold 	Examine the conversion probabilities using Bayes Theorem
(Van den Poel & Buckinx, 2005)	Online wine retailer	<ul style="list-style-type: none"> • Number of visits 	Classification: Logit modeling
(Senecal, Kalczynski, & Nantel, 2005)	Experimental data	<ul style="list-style-type: none"> • Clickstream compactness • Clickstream stratum • Number of visited • Revisited page ratio • Shopping time. 	Analyze the behavior differences
(Kim, Im, & Atluri, 2005)	NASA, NJMC and CIMIC	<ul style="list-style-type: none"> • Didn't indicate 	A sequentially applied model
(Zheng, Cui, Yue, & Zhao, 2010)	Not specific	<ul style="list-style-type: none"> • page detention time • user browsing time, 	A module-based approach to calculate user' interest
(Rathipriya & Thangavel, 2011)	MSNBC	<ul style="list-style-type: none"> • Browsing frequency 	Using a new cluster algorithm
(Aguar & Martens, 2016)	Nielsen NetView	<ul style="list-style-type: none"> • Number of visits 	A mode-based approach
(Chiang, Wang, & Chu, 2013)	A portal site in Taiwan.	<ul style="list-style-type: none"> • Number of clicks 	Predict preference with a Time factor (CPIT) model
(Chen & Su, 2013)	E-commerce	<ul style="list-style-type: none"> • Visit path • Visit frequency • Duration 	Cluster users to groups to identify user interest
(Su & Chen, 2015)	E-commerce	<ul style="list-style-type: none"> • visiting sequence • visiting frequency • time spent on each category 	Using a new clustering algorithm
(Wang, et al., 2017)	Social media	<ul style="list-style-type: none"> • Session length • Session frequency • Click sequence 	Cluster existing data to different clusters
(Volk, Shareef, Jamous, & Turowski, 2017)	E-commerce	<ul style="list-style-type: none"> • Visiting sequence 	Implemented the approach in (Su et al., 2015)

		<ul style="list-style-type: none"> • Visiting frequency • Time spent on each category 	
(Kumbaroska & Mitrevski, 2017)	Digital bookstore	<ul style="list-style-type: none"> • Number of visits • Navigation pattern 	A Markov Chain Clustering approach

Table 2.2: Literature review stage-based approaches on clickstream data

Author	Websites	Phases
Bucklin, Sismeiro 2004	Automotive E-commerce	Product configuration, personal information, order with credit card.
(Kim, Yum, Song, & Kim, 2005)	Online CD store	Click, basket placement, purchase
(Park & Chang, 2009)	Book store	Click, basket placement, purchases
(Kim & Yum, 2011)	Online CD store	Click, basket placement, purchases

2.1.1 A Stage-based Approach (Kim, Yum, Song, & Kim, 2005)

Behavior patterns have been discovered in addition to the user-item matrix to enhance potential purchasing possibilities in Kim05Rec (Kim, Yum, Song, & Kim, 2005). It analyzes the existing customer behavior data such as searching, browsing, clicking, basket placement and purchasing statistically and forms a basket placement decision tree which takes a user's behavior data on a product as input, and outputs the possibility of the user placing the product into basket. Then it fills the user-item matrix with the basket possibility data and uses CF to calculate the purchasing possibility.

The algorithm was proposed based on the workflow in Figure 2.1, which divided the web activities in an E-commerce site into different stages, while calculating the probabilities of moving to the next stage from current stage, it eventually finds the relationship between purchasing and the previous activities such as browsing, searching and clicking. The algorithm is as follows:

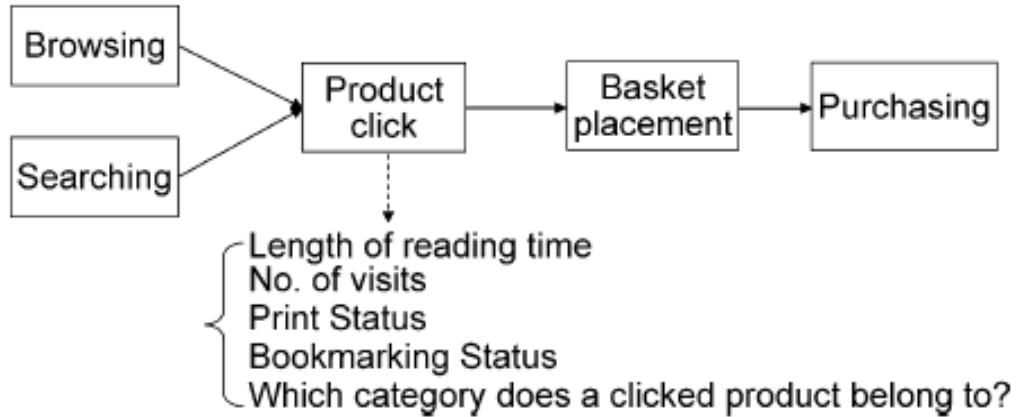


Figure 2.1: A general workflow for possible actions in E-commerce sites

- 1) Gather all the data related to purchase, navigational, and behavior patterns, then give the descriptive statics in Table 2.3 and Table 2.4:

Table 2.3: Data type collected from the experimental E-commerce site

Parameters	Descriptions
Click type	Binary variable: searching=1; browsing=0
Number of visits	Discrete variable
Length of reading time	Continuous variable (s)
Print status	Binary variable: printZ1; no printZ0
Bookmarking status	Binary variable: bookmarkingZ1; no bookmarkingZ0
Level 1 click ratio (genre)	Continuous variable defined for each product k clicked by customer i. Let j be the category (at Level 1) to which product k belongs. Then, Level 1 click ratio for product, k=(Total number of products clicked by customer i that belong to category j at Level 1)/(Total number of products clicked by customer i)
Level 2 click ratio (specific type)	Continuous variable defined for each product k clicked by customer i. Let j be the category (at Level 2) to which product k belongs. Then, Level 2 click ratio for product, k=(Total number of products clicked by customer i that belong to category j at Level 2)/(Total number of products clicked by customer i)
Basket placement status	Binary variable: basket placement=1; no basket placement=0
Purchase status	Binary variable: purchase=1; no purchase=0

Table 2.4: An example structure of collected data

Case	Customer	CD	Click type	Length of reading time	No. of visits	Level 1 ratio	Level 2 ratio	Basket placement	Purchase
1	1	A	1	49	2	0.67	0.33	1	1
2	1	B	1	15	1	0.67	0.33	1	0
3	1	C	0	4	1	0.33	0.33	0	0
4	2	A	0	6	1	0.75	0.50	0	0
5	2	C	0	8	1	0.75	0.50	0	0
6	2	D	1	12	1	0.25	0.25	1	1
7	2	E	0	6	1	0.25	0.25	0	0

2) Estimate the possibility of adding a clicked product to basket.

a) Estimate the probability p of purchasing a product after basket placement by

$$p = \frac{\text{Total number of cases in which product is purchased}}{\text{Total number of cases in which product is placed in basket}}$$

e.g., total number of cases with products purchased in Table 2.4 is 2, and the total case with basket placement is 3, so $p = 2/3$ in the given example.

b) Estimate the probability b of placing a product after clicking it using decision tree (DT) analysis (Figure 2.2), logistic regression (LR) analysis, or artificial neural network (ANN).

e.g., take case 2 for example to run the decision tree as Figure 2.3, the possibility of basket placement is 19.5%.

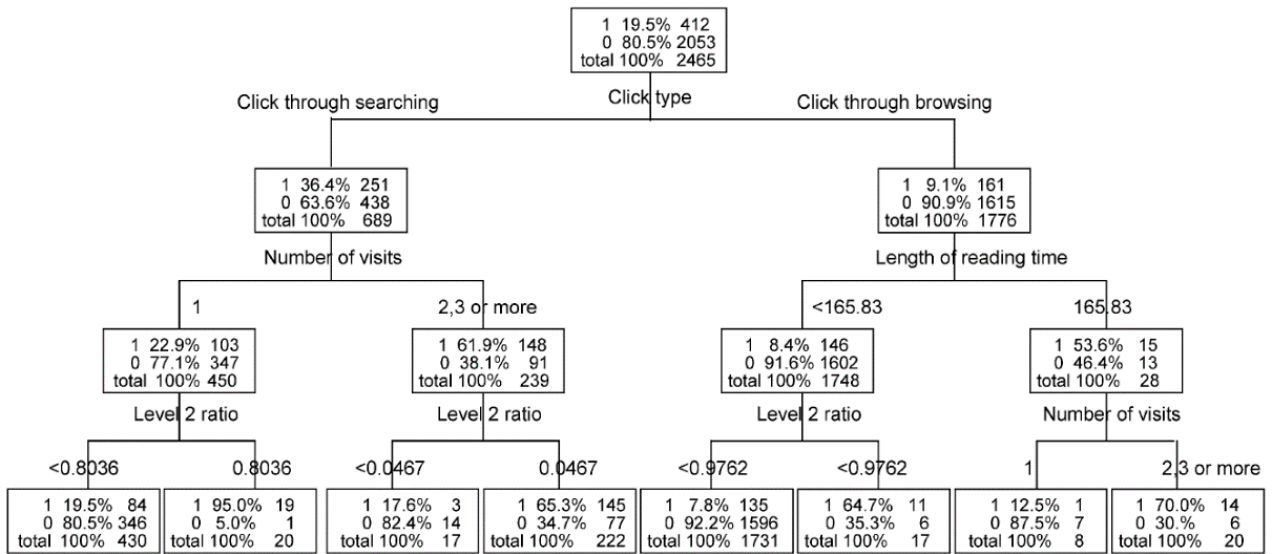


Figure 2.2: Constructed decision tree for reaching basket placement

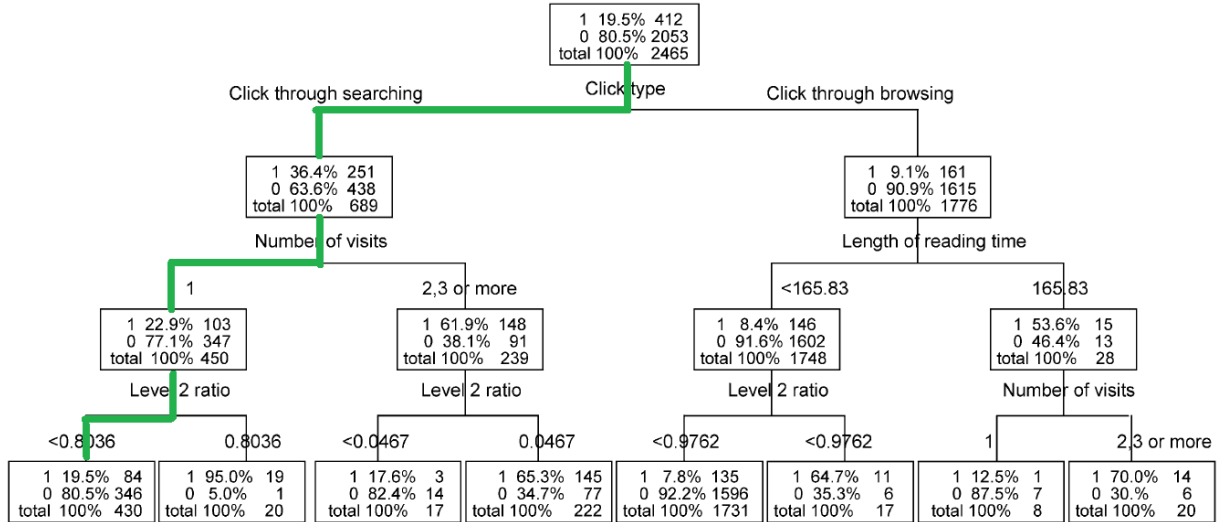


Figure 2.3: An example of using constructed decision

c) The possibility of adding a clicked product to basket is $p \times b$.

e.g., the total of adding a click product to basket in the example is $\frac{2}{3} * 19.5\% = 0.13$.

3) Enrich the user-item matrix with the possibilities from step 3 as in Figure 2.4, then call the conventional collaborative filtering algorithm to make recommendations on the new matrix.

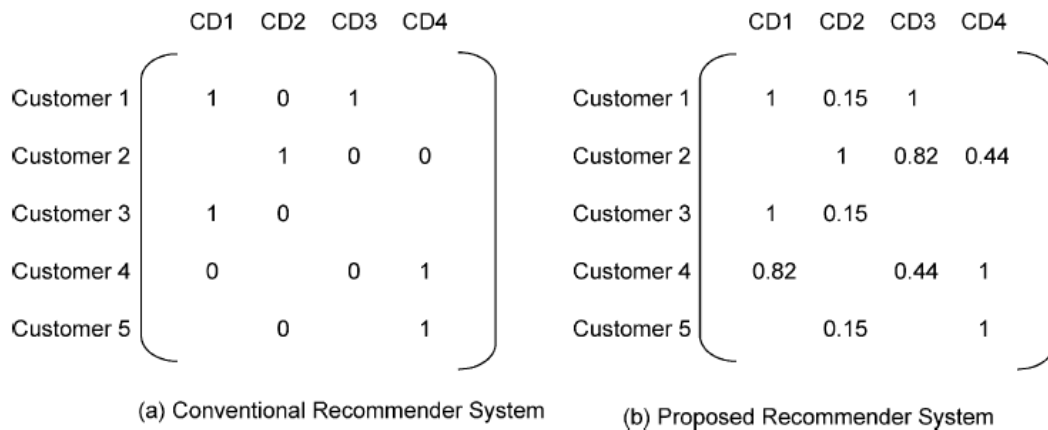


Figure 2.4: A comparison of the original and enriched user-item rating matrix

The proposed algorithm calculates the possibility of placing a product into the basket, and use the result to enrich the user-item rating matrix, the new matrix is more informative and therefore gives better and more accurate recommendations. Nevertheless, it only takes the products which have been previously clicked by the customer as candidates, where using collaborative techniques would make it more general. It also uses the statistical data nesting

in the decision tree to find the possibilities; this method has compromised the relationship information between products by only using the average data as reference.

2.1.2 An Association Rule Approach (Kim & Yum, 2011)

Kim11Rec (Kim & Yum, 2011) integrates with association rule mining calculates confidence between products in different stages like click, basket placement and purchase, then it filters out the top-N products by applying a predefined minimum support. The procedure is as follows:

- 1) Data preparation. Like the preparation in Kim05Rec (Kim, Yum, Song, & Kim, 2005), gather all the data related to purchase, navigational, and behavior patterns, then give the descriptive statics in Table 2.5.

Table 2.5: Data type collected from the experimental E-commerce site

Case	Customer	CD	Click type	Length of reading time	No. of visits	Level 1 ratio	Level 2 ratio	Basket placement	Purchase
1	1	CD _A	1	49	2	0.67	0.33	1	1
2	1	CD _B	1	15	1	0.67	0.33	1	0
3	2	CD _A	0	4	1	0.33	0.33	0	0
4	2	CD _C	0	6	1	0.75	0.50	0	0
5	2	CD _D	0	8	1	0.75	0.50	0	0
6	2	CD _E	1	12	1	0.25	0.25	1	1
7	2	CD _F	0	6	1	0.25	0.25	0	0

Case	Customer	Click CD	Click type	Length of reading time	No. of visits	Level 1 ratio	Level 2 ratio	Basket placement	Purchase
1	1	CD _A	1	High	2	High	Low	1	1
2	1	CD _B	1	Medium	1	High	Low	1	0
3	2	CD _A	0	Low	1	Low	Low	0	0
4	2	CD _C	0	Low	1	High	High	0	0
5	2	CD _D	0	Low	1	High	High	0	0
6	2	CD _E	1	Medium	1	Low	Low	1	1
7	2	CD _F	0	Low	1	Low	Low	0	0

- 2) Association rule mining.
 - a) Identify all pairwise combinations of products that simultaneously appear in a transaction. A transaction consists of the products clicked by a customer. That is, a transaction corresponds to a customer who clicks at least two products. Note that a transaction is said to be made regardless of a purchase. e.g., assume case 1 and 2 happened in one transaction, and there are 50 transactions totally, the number of transactions where CD_A and CD_B are both

clicked is 10, the number of transactions where CD_A is clicked is 13, and 15 for CD_B .

- b) For each pair (e.g., CD_i and CD_j , $i \neq j$) in step (1), calculate the support using following association rule support Equation 2.1 for “ $Click(CD_i), Click(CD_j)$ ”, where $Count(R)$ represents number of transactions in which R occurs, $Count(R, S)$ is the total number of transactions in which both R and S occur, and $Count(All)$ donates to total number of transactions.

Equation 2.1: A formula for calculating support

$$Support = P(U \cap V) = \frac{Count(U, V)}{Count(All)}$$

e.g., product CD_A in case 1 and product CD_B in case 2, $support(CD_A, CD_B)=10/50$.

- c) For each pair whose support is greater or equal than a threshold (e.g., 2%), calculate the lift values using following association rule lift Equation 2.2 for “ $Click(CD_i) \gg Click(CD_j)$ ” and “ $Click(CD_j) \gg Click(CD_i)$ ”:

Equation 2.2: A formula for calculation lift

$$Lift = \frac{P(V|U)}{P(V)} = \frac{P(U \cap V)}{P(U)P(V)} = \frac{Count(U, V) \times Count(All)}{Count(U) \times Count(V)}$$

e.g., lift between CD_A and $CD_B = (10*50)/(13*15)=2.56$.

- d) For each pair “ $Click(CD_i) \gg Click(CD_j)$ ” whose lift is greater than a threshold (e.g., 1), is selected as a candidate for confidence level calculation. As in Table 2.5, in addition to “Click”, “Basket placement”, “Purchase”, each case is associated with other variables such as “Length of reading time”, “Number of visits”, “Level 1 ratio”, and “Level 2 ratio”. Compare the confidence level of CD_i combining different variables such as “ $Click(CD_i) + Length\ of\ reading\ time(High) \gg Click(CD_j)$ ”, “ $Click(CD_i) + Level\ 1\ ratio(Medium) \gg Click(CD_j)$ ”, the highest one is determined as the click confidence level for C_{AB} .

e.g., lift between CD_A and CD_B is greater than 1, there for, the association rule between them is considered significant. Then in all the transactions containing both CD_A and CD_B , find the highest confidence among “Click(CD_A) + length

of reading time(High) >> Click(CD_B)”, “Click(CD_A) + Level 1 ratio(Medium) (High) >> Click(CD_B)”, “Click(CD_A) + Number of visits(>2) >> Click(CD_B)”, etc.

- 3) Calculate confidence. Similar to step 2d), find the confidence level for both basket placement and purchase, if a product CD_G is a qualified associated product for both product CD_A and CD_B , then use the higher confidence level for preference level.
- 4) Top-N recommendation. For each phase (Click, Basket placement, Purchase), find the top-N products ranked by confidence level from 2d) as recommendations.

The approach was proved outperformed the decision tree approach in Kim05Rec (Kim, Yum, Song, & Kim, 2005), but by applying association rule mining which takes all the applicable cases into calculation, it loses the connection between users sharing special interest. E-commerce recommendation systems should care about infrequent users who may have infrequent visiting patterns, this can be improved with collaborative mining techniques.

2.1.3 A Clustering Approach (Chen & Su, 2013)

Chen and Su have been working on clickstream data, most importantly on finding interest patterns using clustering algorithms. Chen13Rec (Chen & Su, 2013) and Chen15Rec (Su & Chen, 2015) proposed algorithms to find the close neighbors who share similar interests by mining clickstream data for top-N recommendations. This approach aims at finding the similarity between two users by measuring indicators like category visiting path, category browsing frequency and category access time.

Given a browsing path $P\{url_1, url_2, url_3, \dots, url_n\}$ which is the sequence of webpages browsed during a session. The procedure is given as following:

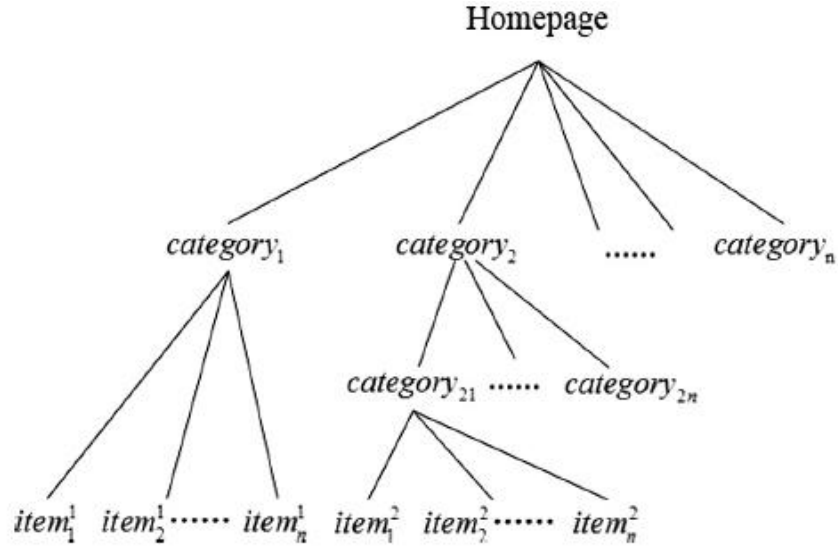


Figure 2.5: Tree of an E-commerce website topology

- 1) Find the category visiting path. Match all the users in path P to the website topology in Figure 2.5.

e.g., $P_i = \{ctg_1, item_1^1, ctg_1, item_2^1, item_1^2\}$ when user i firstly visited category ctg_1 , then $item_1^1$, then category ctg_1 again, $item_2^1$ which belongs to category 1, then $item_1^2$ which is an item in category 2, where $item_k^j$ belongs to ctg_j . Then form a category visiting path $CtgP_i = \{ctg_1, ctg_2\}$ for this case.

- 2) Calculate the visiting frequency. Hit_i^j represents the total number of $user_i$ visits category ctg_j in a session from Equation 2.3, which consist of the number of visits to ctg_j and the length of the visits to product items which belong to ctg_j , and $count(u, v)$ donates the number of visits of page v visited by user u during a session.

Equation 2.3: Calculating the frequency a user spent on a category

$$hit_i^j = count(user_i, ctg_j) + \sum_{k=1}^l count(user_i, item_k^j)$$

Then it finds the frequency of the hits happened on a category during a session following Equation 2.4, where $len(P_i)$ is the length of the visiting sequence.

Equation 2.4: Calculating the frequency ratio a user spent on a category

$$freq_i^j = \frac{hits_i^j}{len(P_i)} \quad (0 \leq freq_i^j \leq 1)$$

e.g., we can calculate $freq_1^1=0.8$, $freq_1^2=0.2$. With all the value calculated for the frequency of a user on a category, form the category-user matrix as in Equation 2.5.

Equation 2.5: A category-user matrix recording the click frequency

$$R_{TKC}^{freq} = \begin{pmatrix} freq_1^1 & freq_1^2 & \dots & freq_1^j & \dots & freq_1^s \\ freq_2^1 & freq_2^2 & \dots & freq_2^j & \dots & freq_2^s \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ freq_i^1 & freq_i^2 & \dots & freq_i^j & \dots & freq_i^s \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ freq_t^1 & freq_t^2 & \dots & freq_t^j & \dots & freq_t^s \end{pmatrix}.$$

- 3) Calculate the relative duration. This paper defines $duration_i^j$ to donate the total time of $user_i$ spend browsing category ctg_j in a session as Equation 2.6, Where $time(u, v, s)$ is the time spent on page/category v by user u for the s th time in a session.

Equation 2.6: Calculating the accumulated time a user spent on a category

$$duration_i^j = \sum_{s=1}^{count(user_i, ctg_j)} time(user_i, ctg_j, s) + \sum_{k=1}^l \sum_{t=1}^{count(user_i, item_k^j)} time(user_i, item_k^j, t)$$

Then it calculates the relative duration by Equation 2.7, where $time(P_i)$ is the total time $user_i$ spent during the session.

Equation 2.7: Calculating the accumulated time ratio a user spent on a category

$$redu_i^j = \frac{duration_i^j}{time(P_i)}$$

Similar to the frequency calculation, it forms another category-user matrix to store all the relative duration results as Equation 2.8:

Equation 2.8: A category-user matrix recording the accumulated time

$$R_{T \times U}^{rtime} = \begin{pmatrix} rtime_1^1 & rtime_1^2 & \dots & rtime_1^j & \dots & rtime_1^s \\ rtime_2^1 & rtime_2^2 & \dots & rtime_2^j & \dots & rtime_2^s \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ rtime_i^1 & rtime_i^2 & \dots & rtime_i^j & \dots & rtime_i^s \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ rtime_t^1 & rtime_t^2 & \dots & rtime_t^j & \dots & rtime_t^s \end{pmatrix}.$$

- 4) Measure the similarity. For two different users p and q , the visiting path similarity can be calculated by Equation 2.9 finding the maximum longest sub common sequence ratio, Equation 2.10 and Equation 2.11 shows using cosine similarity on the matrices from step two and three to measure the visit frequency and relative duration similarity respectively.

Equation 2.9: Measure the similarity between visiting paths

$$sim_{pq}(path) = \max \frac{common(CtgP_p, CtgP_q)}{[len(CtgP_p), len(CtgP_q)]}$$

Equation 2.10: Measure the similarity between visiting frequency

$$sim_{pq}(freq) = \cos(R^{freq}[p, \cdot], R^{freq}[q, \cdot])$$

Equation 2.11: Measure the similarity between visiting accumulated time

$$sim_{pq}(time) = \cos(R^{time}[p, \cdot], R^{time}[q, \cdot])$$

Equation 2.12: Measure the similarity between two users

$$sim_{pq} = \alpha \times sim_{pq}(seq) + \beta \times sim_{pq}(freq) + \gamma \times sim_{pq}(time)$$

- 5) Clustering and recommendation. Based on the total similarity from the step four using Equation 2.12, a k-means like clustering algorithm was introduced to cluster users, and for all the users in a same cluster, a case study proved that they share similar interest. Moreover, top-N most similar users can be filtered out for a given user, and their interests are considered as recommendations for the given user.

These two systems simply measure the interest by browsing paths from the clickstream data. Different from using purchase data, it has proved that clickstream data contains information for cluster users by their browsing path. Nevertheless, the information mined from clickstream data is always confirmed by purchase data, so integrating purchase data

would make a significant difference on the recommendation accuracy. Moreover this system requires specific domain knowledge for categories, and only supports category level recommendations, an enhancement on the generality can make it more useful and integratable.

2.2 E-commerce Recommendation Systems on Transaction Data

Traditional recommendation systems such as Amazon03Rec (Linden, Smith, & York, 2003) uses simple transaction data or the rating matrix, activity data hiding in transaction data were also mined by some algorithms such as Fan14Rec (Fan, Pan, & Jiang, 2014).

2.2.1 Item-Item Collaborative Filtering (Linden, Smith, & York, 2003)

Amazon is the most famous B2C Company crossing the whole world. A good recommendation system helps with promoting business and maintaining customers. Scalability is the most important issue for Amazon given the huge group of customer. Cluster model-based CF was also studied in (Xue, et al., 2005) focusing on fixing the scalability and sparse data issue for traditional collaborative filtering systems, whereas Amazon (Linden, Smith, & York, 2003) developed their item-to-item collaborative filtering based on item-based top-N RS in (Deshpande & Karypis, 2004) and solved the scalability issues and responses in real time. The proposed algorithm seeks for the co-purchased users for a given product as candidate users, therefore reduces the candidate size. The algorithm is explained in Algorithm 2.1:

Algorithm 2.1: Amazon Item-Item Collaborative Filtering

For each item in product catalog, I_1
For each customer C who purchased I_1
For each item I_2 purchased by customer C
Record that a customer purchased I_1 and I_2
For each item I_2
Compute the similarity between I_1 and I_2

To explain the algorithm for wallet with an example using transaction Table 2.6:

Table 2.6: Example Data for Amazon CF

	Category 1	Category 2			Category 3	
	iPhone 7	Wallet	Backpack	iPhone 7 case	Cereal	Peanut butter
Adam	Y	?	?	Y	?	Y
Abby	Y	Y	Y	?	?	?
Ellen	?	?	?	?	Y	?
Daniel	Y	Y	?	Y	?	?

- 1) For wallet in Category2, filter out all the customers who have purchased it as in Table 2.7;

Table 2.7: Associated Customers for Wallet in Amazon CF

	Category 1	Category 2			Category 3	
	iPhone 7	Wallet	Backpack	iPhone 7 case	Cereal	Peanut butter
Abby	Y	Y	Y	?	?	?
Daniel	Y	Y	?	Y	?	?

- 2) From Table 2.7, we can find Abby and Daniel also bought iPhone 7, Backpack and iPhone 7 case, then maintain the candidate table like Table 2.8:

Table 2.8: A Table of Candidate items

Base Item	Candidate Item
Wallet	iPhone 7
Wallet	Backpack
Wallet	iPhone 7 case

- 3) Calculate the similarity between each data set, then find the top-k recommendations for Wallet. If k is 2, then the recommendations would be iPhone 7 and Backpack.

Table 2.9: Result Table of Calculating the similarity

Base Item	Candidate Item	Similarity(Euclidean Distance)
Wallet	iPhone 7	1
Wallet	Backpack	1
Wallet	iPhone 7 case	2

- 4) Then keep finding the recommendations for other items in Category2.

Amazon's item-to-item collaborative filtering calculates offline to produce the similar-item tables, and then gives good suggestions on small-scale data by finding and ranking other items that are purchased by similar customers. Therefore, item-to-item collaborative

filtering reduces the time complexity and keeps the high quality of recommended products. Nevertheless, transaction data is only a small input from users, clickstream data has been studied and proved very useful for recommendation systems, integrating it would improve the accuracy considerably.

2.2.2 Combine Content-based CF and User Activity (Fan, Pan, & Jiang, 2014)

In Fan14Rec (Fan, Pan, & Jiang, 2014), the writers combine users' activities with content-based information to calculate values for the unrated spots in sparse matrixes. Users' actions like clicking a particular type of items, commenting habit, the amount of time they browse the pages, or how frequently they revisit some certain types of items, all this information can be formalized and used in recommender systems. Original content-based filtering algorithm always ignores what users do. Instead, it only cares about those preferences users have in their profiles, which change a lot with time going by. On the contrary, users' activities happen to make monitoring these changes possible, and therefore the hybrid system can automatically adjust users' profiles and make the content-based algorithm more accurate.

Equation 2.13: Activity Rate Formula

$$Act(u, i) = \frac{T_i}{T_{total}}$$

They introduced a new method as Equation 2.13, which calculates the activity value of user u for item i . In addition, T_i is the total rating times of user u on the category of the item, T_{total} represents the total rating times from user u . Hence, the activity value actually is same for each genre of items; it is the ratio of a user's rating time out of the user's rating time. To test the new algorithm, the authors pulled data from move-lens, in the result as Figure 1; it is evident that out of 18 movie genres, 10.5 of them have user activity less than 0.1. This feature shows that many users have similar rating frequency, and then it is possible to find the pattern and group users. The whole purpose of user-based collaborative filtering algorithm is finding similar users and recommend them items the neighbors like. The writers presented another Equation 2.14 to calculate the rating value from user u for item i , it first finds all the users whose activity values are lower than 0.1 and also rated item i , this group is called NALS (Nearest-Activity-Level-Set), adding up all the user activities

for item i in NALS then divide by the number of items in NALS, this average rating value reflects user u 's potential rating to some extent.

Equation 2.14: User-item Rate Calculation Formula

$$PA_{u,i} = \frac{\sum_{v \in \text{NALS}} R_{v,i}}{n}$$

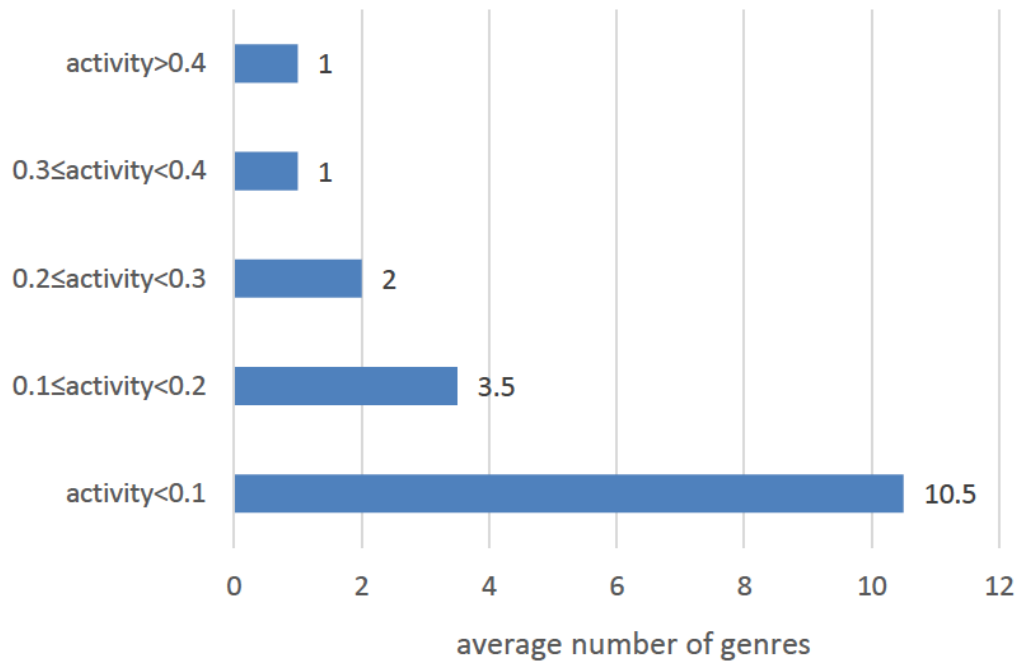


Figure 2.6: Activity Rate Distribution

The whole algorithm was given step by step in the paper, we will explain with an example with Table 2.10 and Table 2.11:

Table 2.10: Example Rating Table for Hybrid RS

	iPhone 7	iPhone 7 case	Cereal	Peanut butter	Wallet	Backpack
Adam	4	5	2	4	5	3
Abby	?	2	3	3	?	4
Ellen	1	5	?	5	2	1
Daniel	4	1	3	2	1	1
Eric	5	5	2	5	4	3

Table 2.11: Example Preference Table for Hybrid RS

	iPhone 7	iPhone 7 case	Cereal	Peanut butter	Wallet	Backpack
Eatable	0	0	1	1	0	0
Office Supply	0	0	0	0	0	1
Electronics	1	1	0	0	0	0
Fashion	1	1	0	0	1	1

- 1) Initiate user-item matrix (Table 2.12) and item-attribute matrix (Table 2.13) by user rating recording and item attributes respectively.

Table 2.12: An example of user-item rating matrix

4	5	2	4	5	3
	2	3	3		4
1	5		5	2	1
4	1	3	2	1	1
5	5	2	5	4	3

Table 2.13: Item attribute matrix

0	0	1	1	0	0
0	0	0	0	0	1
1	1	0	0	0	0
1	1	0	0	1	1

- 2) Use content-based algorithm and Equation 2.15 to formulate preliminary predictive value $PC_{u,i}$ for each empty value in the user-item matrix. Then the new user-item matrix would be like Table 2.14.

Equation 2.15: Preliminary Predictive Rating Formula

$$PC_{u,i} = \frac{\sum_{j \in T_u} sim(i, j) \times R_{u,j}}{\sum_{k \in T_u} |sim(i, j)|}$$

Table 2.14: Preliminary Predictive Rating Table

4	5	2	4	5	3
3.95	2	3	3	3.18	4
1	5	2.67	5	2	1
4	1	3	2	1	1
5	5	2	5	4	3

- 3) Compute the predictive rating value $PA_{u,i}$ based on user activity for every missing rating value by Equation 2.14. Given the small amount of category in sample data, we pick all the items that have Act value less than 0.2 for NALS group. The updated matrix would be like matrix in Table 2.15, because of the proportion of the products in different categories, it copies the same rate that the user has rated for another item in the same category.

Table 2.15: Updated Rating Table

4	5	2	4	5	3
2	2	3	3	4	4
1	5	5	5	2	1
4	1	3	2	1	1
5	5	2	5	4	3

- 4) Fuse the value of $PC_{u,i}$ and $PA_{u,i}$, and the final predictive value can be formulated by the following Equation 2.16:

Equation 2.16: Hybrid RS Final Predict Formula

$$P_{u,i} = \frac{\sum_{j \in T_i} sim(i, j) R_{u,j}}{\sum_{k \in T_i} |sim(i, j)|} \times (1 - \lambda) + \frac{\sum_{v \in NALS} R_{v,i}}{n} \times \lambda$$

Where $P_{u,i}$ is the final predictive value of user u on item i , λ is a weight factor to adjust the relative weight of $PA_{u,i}$ and $PC_{u,i}$. From the experiment in the paper, when λ is 0.1, Mean Absolute Error (MAE) is the smallest which means the precision is the best.

So when λ is 0.1, use Equation 2.15 can get the final predicted rating value for the three missing ratings as Table 2.16.

Table 2.16: Final Predicted Rating

4	5	2	4	5	3
2.76	2	3	3.26	4	4
1	5	5	5	2	1
4	1	2.9	2	1	1
5	5	2	5	4	3

- 5) Fill each missing rating in the user-item matrix with the final predictive value and then use user based collaborative filtering algorithm to generate recommendations.

The improvement of accuracy was proved in the paper with experiments on 100k data from movie-lens. The authors used Mean Absolute Error (MAE) to measure the accuracy for both original content-based filtering algorithm and the improved algorithm combining with user activity, the result showed that the new approach reduced MAE considerably. This paper gave a solid proof to the fact that the activity data has positive relationships with user's purchase data, in addition to the user activity pattern hiding in the rating table, clickstream data was proved to hold most of the implicit activity data, therefore, integrating clickstream data would benefit the accuracy.

2.3 Sequential Similarity Measurement

Sequential pattern mining has been a very hot research area, but exploring the relationships between sequences seems less interesting. But if there is a proper algorithm to measure the likelihood between sequences, then algorithms for well-structured data like k-means clustering can all be used on sequences with different length, therefore make current data mining techniques more general. Sequence similarity has been researched in biology on detecting homology for DNAs, the algorithms are highly domain-oriented which cannot be used in datamining. Edit distance and LCS are two methods for this purpose but not enough. A modified version of edit distance was also proposed in (Bozkaya, Yazdani, & Özsoyoğlu, 1997) which groups sequences by length and makes it less general. S²MP was proposed by Saneifar et al. in 2008 in order to measure the similarity between two sequences composed of sets of items which is not applicable in this case. We will specifically talk about Edit Distance and LCS in this section.

2.3.1 Edit Distance & Modified Edit Distance

Edit distance was first introduced by Levenshtein in 1966; it finds how many steps it requires to change a sequence to another. Therefore the number of the steps can be used as the distance between two sequences. In (Capelle, Masson, & Boulicaut, 2002), edit distance was used as similarity measurement for sequential pattern mining by giving different weights to different operations like insertion, deleting and substitution. We will calculate "bcdab" and "cda" as an example to explain the edit distance calculation without weights.

Levenshtein distance is usually explained by matrix following the dynamic programming formula in Equation 2.17.

Equation 2.17: Edit Distance Equation

$$\begin{aligned}
 d_{i0} &= \sum_{k=1}^i w_{\text{del}}(b_k), & \text{for } 1 \leq i \leq m \\
 d_{0j} &= \sum_{k=1}^j w_{\text{ins}}(a_k), & \text{for } 1 \leq j \leq n \\
 d_{ij} &= \begin{cases} d_{i-1,j-1} & \text{for } a_j = b_i \\ \min \begin{cases} d_{i-1,j} + w_{\text{del}}(b_i) \\ d_{i,j-1} + w_{\text{ins}}(a_j) \\ d_{i-1,j-1} + w_{\text{sub}}(a_j, b_i) \end{cases} & \text{for } a_j \neq b_i \end{cases} & \text{for } 1 \leq i \leq m, 1 \leq j \leq n.
 \end{aligned}$$

- 1) Find variable n equals the length of "bcdab" which is 5, and m as the length of "cda" which is 3. If any of the string is empty, return the length of the other string, which is not the case of the example.
- 2) Construct a matrix and initialize the first row and the first column as Table 2.17:

Table 2.17: Initialized Matrix for Edit Distance Calculation

		c	d	a
	0	1	2	3
b	1			
c	2			
d	3			
a	4			
b	5			

- 3) Define variable s="bcdab", t="cda". Examine each character of s, the index of the character i is from 1 to n, the index of the character for t is j from 1 to m:

- a. If $s[i]$ equals $t[j]$, the cost which is the steps it needs to convert is 0. Otherwise, it's 1.
- b. Find the minimum value from following options for the cell $d[i,j]$:
 - i. The top adjacent cell plus 1: $d[i-1,j]+1$
 - ii. The left adjacent cell plus 1: $d[i,j-1]+1$
 - iii. The cell in the left top plus the cost: $d[i-1,j-1]+cost$
- c. Recursively execute step 3 until all the cells are filled. The resulting matrix would be like Table 2.18.

Table 2.18: Result matrix for Edit Distance

		c	d	a
	0	1	2	3
b	1	1	2	3
c	2	1	2	3
d	3	2	1	2
a	4	3	2	1
b	5	4	3	2

Follow the above steps, the edit distance for all the sequences are in Table 2.19. Edit distance calculated the relationship between sequences to some extent, but it only cares about the relative order of items in one way, does not compromise the reversal at all. Sometimes, the time complexity (MN) is also not acceptable for real-time calculation on large data.

Table 2.19: Edit Distance Result Table

Candidate Sequences	Target Sequence	Edit Distance
abcbcadda	cda	6
bcdab		2
ddcabba		5
cbaadac		4

2.3.2 LCS: Longest Common Subsequences

LLCS (the length of the longest common subsequences) was proposed in (Paterson & Vlado, 1994) as a useful measurement for the similarity of two strings. Since then it has been used for different areas, like the ADMIT application in (Sequeira & Zaki, 2002),

which used LCS as the similarity measurement. LLCS (Equation 2.18) has been dynamically programmed by researchers; We will illustrate the algorithm with string "bcdab" and "cda" as an example:

Equation 2.18: LCS Equation

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \cup x_i & \text{if } x_i = y_j \\ \text{longest}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

- 1) If any of the string is empty, return \emptyset ;
- 2) Construct a matrix and initialize the first row and the first column as Table 2.20:

Table 2.20: Initialized Matrix for LCS

\emptyset	\emptyset	c	d	a
\emptyset				
b				
c				
d				
a				
b				

- 3) Define variable $x="bcdab"$, $y="cda"$. Examine each character of x , the index of the character i is from 1 to n , the index of the character for y is j from 1 to m :
 - a. If $x[i]$ equals $y[j]$, the cell string $LCS(x_i, y_j) = LCS(x_{i-1}, y_{j-1}) \cup x_i$;
 - b. Find the longest string value from following options for the $LCS[x_i, y_j]$:
 - i. The top adjacent cell string: $d[x_{i-1}, y_j]$
 - ii. The left adjacent cell string: $d[x_i, y_{j-1}]$
 - iii. $LCS(x_i, y_j)$ from step a, it is \emptyset when $x[i]$ doesn't equals $y[j]$.
 - c. Recursively execute step 3 until all the cells are filled. The matrices for steps would be like Table 2.21.

Table 2.21: Result matrix for LCS

∅	∅	c	d	a
∅	∅	∅	∅	∅
b	∅	∅	∅	∅
c	∅	c	c	c
d	∅	c	cd	cd
a	∅	c	cd	cda
b	∅	c	cd	cda

Follow the above steps, the LCS for all the sequences are in Table 2.22. LCS also calculates the common sequences from left to right and does not count the distance between items, as in Table 2.22, three sequences turned out having the same LCS but there is more information that can be calculated into the relationship of sequences.

Table 2.22: LCS Result Table

Candidate Sequences	Target Sequence	LCS
abcbcadda	cda	cda
bcdab		cda
ddcabba		da
cbaadac		cda

CHAPTER 3

PROPOSED RECOMMENDATION SYSTEM

In addition to using transaction data for recommendation system, Kim05Rec (Kim, Yum, Song, & Kim, 2005), Kim11Rec (Kim & Yum, 2011) and Chen13Rec (Chen & Su, 2013) have integrated features from clickstream data such as visit times, basket placement rate, association rule, category visit sequence, category visit duration and category visit frequency into recommendation system to make better recommendations. In this paper, we propose a novel recommendation system HPCRec (Algorithm 3.1) integrated with purchase frequencies and the consequential relationship between clicks and purchases. By processing this information, it enhances the user-item rating matrix in both quantity and quality and then improves the recommendations. We use pre-processed consequential table (Table 3.1) and frequency matrix (Table 3.4) as input, and HPCRec returns a matrix with predicted ratings (Table 3.7). There are three functions FN (Frequency Normalization), CSSM and TWFI used in HPCRec; HPCRec was also proven being able to give better recommendations to infrequent users.

3.1 Input Data

HPCRec takes a consequential table and user-item purchase frequency matrix as input, it mines the consequential information from the consequential table to enrich the rating table, normalizes the purchase frequencies to improve the rating quality, finally makes better recommendations.

3.1.1 Consequential Table

We introduce consequential table here, which maintains all the browser sessions happened before, for each session, a user may make some clicks and purchases. To generate a consequential table, we combine the clickstream table (Table 3.2) and transaction (Table 3.3) table and generate a consequential table (Table 3.1) as input which contains all the clicks and purchases happened in each session, for our system HPCRec.

Table 3.1: An example of consequential table

SessionId	UserId	Clicks	Purchases
1	1	1,2	2
2	1	3,5,2,3	2,3
3	2	2,1,4	1,2,4
4	2	4,4,1,2	2,4,4
5	3	1,2,1	1
6	3	3,5,2	

Clickstream table is the electronic record of a user's activity on the Internet (Bucklin & Sismeiro, 2009). We construct the clickstream table for click events as Table 3.2, where sessionId is the primary key; userId is for user id; itemId is the product Id which was clicked in the event, category shows the category id where this product belongs to, and time records when did this click event happen.

Table 3.2: Sample schema of Clickstream table

SessionId	UserId	ItemId	Category	Time
1	1	1	1	2014-04-0211:44:11
1	1	2	1	2014-04-0211:46:37
2	1	3	3	2014-04-0811:15:35
2	1	5	3	2014-04-0811:18:23
2	1	2	1	2014-04-0811:21:12
2	1	3	3	2014-04-0811:23:44
3	2	2	1	2014-04-2511:16:14
3	2	1	1	2014-04-2511:19:47
3	2	4	2	2014-04-2511:23:07
...				

Transaction table records all the transactions with customers, which usually contain the product list and the customer information. We use a traditional table schema to store transaction data in Table 3.3, where sessionId is the primary key, userId is the user id,

purchase holds the product set (1,2,3,4 are the product name in the example) bought by the customer.

Table 3.3: Sample schema of the Transaction table

SessionId	UserId	Purchases
1	1	2
2	1	2,3
3	2	1,2,4
4	2	2,4,4
5	3	1
6	3	

3.1.2 User-item Purchase Frequency Matrix

In E-commerce, we define user-item purchase frequency matrix as in Table 3.4 where each row represents the purchasing amount for userd (1,2,3) respectively, and each column is for products (1,2,3,4) respectively. It is obvious to notice that compared to the binary rating matrix which only shows the fact whether or not a user has bought an item before or not, the purchase frequency matrix is more informative.

Table 3.4: User-item purchase frequency matrix

Customer\Item	1	2	3	4
1	?	2	1	?
2	1	2	?	3
3	1	?	?	?

3.2 Proposed Method: HPCRec (Historical Purchase and Clickstream based Recommendation System)

The proposed HPCRec recommendation system takes a consequential table (Table 3.1) and purchase frequency matrix (Table 3.4) from the previous section as input, generates predicted ratings for unknown ratings, which stand for the unclear interests from users to items. There are five main steps for the HPCRec system (Algorithm 3.1):

Algorithm 1 HPCRec System to Predict Ratings

Input: C , consequential table; F , frequency matrix

Output: P , a rating matrix with predicted ratings

```
1:  $M$ , normalized rating matrix  $\leftarrow$  FN( $F$ ) in Section 3.1;
2: for all  $N$ , session without a purchase  $\in$  consequential table do
3:    $T$ , weighted transaction table  $\leftarrow$  null;
4:   for all  $Y$ , session with purchase  $\in$  consequential table do
5:     similarity  $\leftarrow$  CSSM( $N$ .clicks,  $Y$ .clicks) in Section 3.2;
6:     add (similarity,  $Y$ .purchases) to  $T$ ;
7:   end for
8:    $Is$ , weighted frequent items  $\leftarrow$  TWFI( $T$ ) in Section 3.3;
9:   for all  $I$ , weighted frequent item  $\in Is$  do
10:    if  $M$  does not contain ratings for ( $N$ .user, $I$ .item) then
11:      add ( $N$ .user, $I$ .item, $I$ .weight) to  $M$ ;
12:    end if
13:  end for
14: end for
15:  $P$ , rating matrix with predicted ratings  $\leftarrow$  CF( $M$ );
16: return  $P$ ;
```

Algorithm 3.1: Algorithm for HPCRec recommendation system

- (1) Normalizing user-item purchase frequency matrix to a new user-item rating matrix M , details in section 3.2.1 (Line 1 in Algorithm 3.1).
- (2) In the consequential table, for each session N (Line 2 in Algorithm 3.1) without a purchase belonging to a user N .user, find the top- N similar sessions with purchases (Line 3 in Algorithm 3.1) by comparing the click sequences using function CSSM (Clickstream Sequence Similarity Measurement, Line 5 in Algorithm 3.1) in section 3.2.2. Then use the similarity as weight and assign to the purchases in the selected top- N session (Line 6 in Algorithm 3.1). This step generates a weighted transaction table T where weights are similarities assigned to purchases.
- (3) Use the weighted transaction table T from step 2, call function TWFI (Transaction-based Weighted Frequent Item, Line 8 in Algorithm 3.1) in section 3.2.3 and get a list of items with purchasing possibilities Is .
- (4) For each item I in Is from the previous step (Line 9 in Algorithm 3.1, which are items that have shown to have purchasing possibilities from click analysis), if the

user $N.user$ from step 2 (that is, the users with sessions with no purchase) has not previously purchased the product (Line 10 in Algorithm 3.1), enrich the result matrix from step 1 with the possibility (Line 11 in Algorithm 3.1). Then return to step 2 for the next session without a purchase if possible, and otherwise continue to step 5.

- (5) With the enriched rating matrix, run the CF algorithm and predict ratings. Return the rating matrix with predicted ratings P (Line 15 in Algorithm 3.1).

To explain HPCRec with consequential table (Table 3.1) and purchase frequency matrix (Table 3.4) in steps:

- (1) Normalize the purchase frequency in Table 3.4 for each user on each item, and get a normalized rating matrix (Section 3.2.1) as Table 3.5;

Table 3.5: Normalized user-item purchase frequency matrix

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	?	?	?

- (2) For each session without a purchase, such as session 6 for user 3 in Table 3.1. Calculate the similarity between session 6 and other sessions with purchases (1,2,3,4,5) by comparing the clicks calling CSSM in section 3.2.2, get $CSSM(\langle 3,5,2 \rangle, \langle 1,2 \rangle) = 0.37$, $CSSM(\langle 3,5,2 \rangle, \langle 3,5,2,3 \rangle) = 0.845$, $CSSM(\langle 3,5,2 \rangle, \langle 2,1,4 \rangle) = 0.33$, $CSSM(\langle 3,5,2 \rangle, \langle 4,4,1,2 \rangle) = 0.245$, $CSSM(\langle 3,5,2 \rangle, \langle 1,2,1 \rangle) = 0.295$; form a weighted transaction table using the similarity as weight and purchases as transaction records such as [$\langle (2):0.37 \rangle$, $\langle (2,3):0.845 \rangle$, $\langle (1,2,4):0.33 \rangle$, $\langle (2,4,4):0.245 \rangle$, $\langle (1):0.295 \rangle$];
- (3) Call TWFI in section 3.2.3 with the weighted transaction table from step 2, and get weighted frequent items (2:1, 3:0.189, 4:0.167); for all weighted frequent items, if the user has not purchased it, add the possibility into the normalized frequency matrix such as in Table 3.6.

Table 3.6: Enriched user-item normalized purchase frequency matrix of HPCRec

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	1	0.19	0.167

(4) Return to step 2 if there is more session without a purchase, otherwise, run the CF algorithm using the updated rating matrix (Table 3.6) to get predicted ratings for all of the original unknowns as demonstrated in Table 3.7, return the rating table with predicted ratings. Accuracy also can be calculated for the evaluation purpose. Three Proposed Modules

Table 3.7: User-item rating matrix with predicted ratings for HPCRec

Customer\Item	1	2	3	4
1	0.63	0.89	0.45	0.5
2	0.27	0.53	0.35	0.8
3	1	0.74	0.27	0.3

3.2.1 FN: Frequency Normalization: Step 1 of HPCRec Algorithm

In this module, we take the user-item purchase frequency (Table 3.3) as input, normalize the frequencies into numbers between 0 and 1 using the unit vector formula (Weisstein, 2002) (Equation 3.1). For each user, $\langle x_1, x_2, x_3 \dots x_n \rangle$ is the purchase vector showing the purchase frequency of product $\langle 1, 2, 3, \dots, n \rangle$ respectively. For user 2, the purchase vector is $\langle 1, 2, 0, 3 \rangle$, so the normalized purchase frequency for user 2 on item 2 is $2/\sqrt{1^2 + 2^2 + 0^2 + 3^2} = 0.53$. The normalized frequency matrix is in Table 3.5, from which we can see that for each user, the differences between ratings reflects the different level of interest.

We also tried the feature scaling normalization method (Equation 3.2) to normalize the frequencies, but the unit vector formula was proven more effective.

Equation 3.1: Unit vector normalization

$$x' = \frac{x}{\sqrt{x_1^2 + x_2^2 + x_3^2 + \dots + x_n^2}}$$

Equation 3.2: Feature scaling normalization

$$x' = \frac{x - \min}{\max - \min}$$

3.2.2 CSSM-Clickstream Sequence Similarity Measurement: Step 2 of HPCRec Algorithm

Inspired by the idea of Chen13Rec (Chen & Su, 2013), we introduce CSSM (Clickstream sequence similarity measurement) which takes the frequency and position of items in sequences into consideration to calculate the similarity. Instead of calculating the category visiting sequences and frequencies, CSSM calculates product click sequences and frequencies. We explain this function in steps using two click sequences $\langle 3,5,2 \rangle$ and $\langle 3,5,2,3 \rangle$ in Table 3.1 as an example:

- (1) Calculate the longest common subsequence rate $LCSR(x, y) = LCS(x, y) / \max(|x|, |y|)$, where the longest common subsequence (LCS) (Hunt & MacIlroy, 1976) is defined in Equation 3.3. eg., $LCS(\langle 3,5,2 \rangle, \langle 3,5,2,3 \rangle) = 3$, the maximum sequence size is 4, so $LCSR(\langle 3,5,2 \rangle, \langle 3,5,2,3 \rangle) = 3/4$;

Equation 3.3: Longest common subsequence

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \cup x_i & \text{if } x_i = y_j \\ \text{longest}(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases}$$

- (2) Calculate the item frequency similarity (FS). Firstly form a distinct itemset containing all the items in both sequences, eg., $\langle 2,3,5 \rangle$ in this example. For each sequence, form a vector of frequency for the items in itemset, $\langle 1,1,1 \rangle$ for $\langle 3,5,2 \rangle$, and $\langle 1,2,1 \rangle$ for $\langle 3,5,2,3 \rangle$; then find the cosine similarity between two vectors, which is 0.94 in this case;
- (3) Compute the final similarity $\text{Sim} = \alpha * \text{LCSR} + \beta * \text{FS}$, where $\alpha + \beta = 1$, $0 < \alpha, \beta < 1$, α and β are weight to balance the two indicators from step 1 and 2. In the real procedure, we train our dataset with different α and β to find the best combination for prediction. If set $\alpha = 0.5$, $\beta = 0.5$, the final similarity $\text{Sim}(\langle 3,5,2 \rangle, \langle 3,5,2,3 \rangle) = 0.5 * 3/4 + 0.5 * 0.94 = 0.845$ in the example.

3.2.3 TWFI-Transaction-based Weighted Frequent Item: Step 3 of HPCRec Algorithm

This function takes a weighted transaction table where weights are assigned to each transaction as input, and returns items with weighted support in a given threshold. We explain this with an example [$\langle(2):0.37\rangle$, $\langle(2,3):0.845\rangle$, $\langle(1,2,4):0.33\rangle$, $\langle(2,4,4):0.245\rangle$, $\langle(1):0.295\rangle$], $\text{MinWeightedSupport} = 0.15$, where each unit has pattern and weight in such form $\langle(\text{item ids in a transaction}):weight\rangle$.

- (1) Calculate support. Form a distinct item set from all the transactions, and find the support for each item, e.g., $\langle 1:2,2:4,3:1,4:3\rangle$;
- (2) Compute the average weighted support ($\text{AWS}=\text{AW}*\text{support}$) for each item using the same strategy in (Yun & Leggett, 2005), where average weight ($\text{AW} = \text{sum}(\text{weight})/\text{support}$), which makes $\text{AWS}=\text{sum}(\text{weight})$, e.g., $\text{AWS}(4)=0.33+0.245+0.245=0.82$, $\langle 1:0.625,2:1.79,3:0.845,4:0.82\rangle$; We also tried using maximum weighted support ($\text{MWS}=\text{max}(\text{weight})*\text{support}$), $\text{AWS}(4)=\text{max}(0.33,0.245,0.245)=0.33*3=0.99$, the maximum approach was proven good, but the average approach is better.
- (3) Normalize weighted support using feature scaling (Equation 3.2), so for the average weighted support, $\text{max}=1.79$, $\text{min}=0.625$, then the new average weighted support for item 3 is $(0.845 - 0.625)/(1.79 - 0.625) = 0.189$, all the weighted supports are $\langle 1:0, 2:1, 3:0.189, 4:0.167\rangle$;
- (4) Return all the items with normalized weighted support greater or equal than $\text{MinWeightedSupport}$, e.g., $\langle 2:1, 3:0.189, 4:0.167\rangle$ for using average weighted support;

3.3 An Example Application of Proposed Algorithm

Take the input data as following in Table 3.8, Table 3.9, Table 3.10 and Table 3.11 as input data, we run through Kim05Rec (Kim, Yum, Song, & Kim, 2005), Kim11 (Kim & Yum, 2011), Chen13 (Chen & Su, 2013) and HPCRec respectively to prove that HPCRec is more accurate. To make sure the evaluation is fair, we only select the top 4 most relevant scores from different methods, and give 1 as the rating for the user on the corresponding product to keep the measuring standard consistent.

Table 3.8: Clickstream data for a walk through an example

sid	uid	clickstream	tid	sstart (yyyy.MM.dd.hh.mm.ss)	send (yyyy.MM.dd.hh.mm.ss)
0	0	< 4,2,2,0,2,3,4>	0	2017.09.05.13.23.30	2017.09.05.13.43.00
1	0	<5,3>		2017.06.15.09.00.34	2017.06.15.09.50.20
2	0			2017.03.05.18.53.19	2017.03.05.19.33.14
3	1	< 4,5,1,0,3 >	1	2017.03.05.18.53.19	2017.03.05.19.33.14
4	2	< 0,4,4,0>	2	2017.09.25.15.23.22	2017.09.25.16.23.15
5	2	< 4,3,5,2>			
6	3	< 3,1,3,4,5,2>	3		

Table 3.9: Purchase data for a walk through an example

tid	uid	purchase	time (yyyy.MM.dd.hh.mm.ss)
0	0	{3,2,0,4,4}	2017.09.05.13.23.30
1	1	{1,1,5}	2017.06.15.09.23.34
2	2	{4}	2017.06.15.09.30.34
3	3	{1,4,3,3,3}	2017.03.05.18.59.19

Table 3.10: Rating matrix for a walk through example

User\Item	0	1	2	3	4	5
0	1	?	1	1	1	?
1	?	1	?	?	?	1
2	?	?	?	?	1	?
3	?	1	?	1	1	?

Table 3.11: Product details for a walk through an example

itemId	itemName	price	category
0	a	2	0
1	b	6	1
2	c	3	0
3	d	3	0
4	e	4	2
5	f	7	2

3.3.1 Kim05Rec Method

- 1) Find the valuable indicators for each item, such as the number of visits, length of reading time, basket placement, purchase, etc. In this example, we take the number of visits and basket placement (assuming all the purchased products were placed in the basket before) as our variables, and get following distributed decision table in Table 3.12:

Table 3.12: Decision tree for a walk through an example

itemId	itemName	visits<2	Visits>2	Basket placement when visits<2	Basket placement when visits>=2
0	a	2/3	1/3	1/2	0/1
1	b	2/2	0/2	2/2	0/0
2	c	2/3	1/3	0/2	1/1
3	d	4/5	1/5	1/4	1/1
4	e	3/5	2/5	1/3	2/2
5	f	4/4	0/4	1/4	0/0

- 2) Take the clicked but not purchased session and find the estimated data from Table 3.13. Such as in this case, session 1 has clicks “5,3”, match to basket placement when in Table 3.19.

Table 3.13: Basket placement possibilities for clicked products

sessionId	itemId	visitTime	Basket placement possibility
1	5	1	1/4
	3	1	1/4
5	4	1	1/3
	3	1	1/4
	5	1	1/4
	2	1	0/2

- 3) Remove the estimated ratings where the user has purchased the product, such as shaded item 3 for user 1 in Table 3.14, also remove the basket placement when it is 0, then select the top 4 highest value, in this example, we keep all the 3 records.

Table 3.14: Useful Basket placement possibilities

sessionId	userId	itemId	visitTime	Basket placement possibility
1	0	5	1	1/4
		3	1	1/4
5	2	4	1	1/3
		3	1	1/4
		5	1	1/4
		2	1	0/2

- 4) Modify the original user-item rating matrix with 1 for the qualified ones from the previous step, the enhanced matrix would be like Table 3.15:

Table 3.15: Enriched matrix for kim05Rec

User\Item	0	1	2	3	4	5
0	1	?	1	1	1	0.25
1	?	1	?	?	?	1
2	?	?	?	0.25	1	0.25
3	?	1	?	1	1	?

- 5) Calculate the precision and recall for both of the enriched matrix and the original matrix, and the evaluation data is in Table 3.16.

Table 3.16: Evaluation result for kim05Rec

Method	Precision	Recall
Conventional CF	0.148	0.44
Kim05Rec	0.25	0.75

3.3.2 Kim11 Method

(M=0.3 is the support threshold, N=2.5 is the lift threshold, $\alpha=0.3$, $\beta=0.7$):

- 1) For each user, find all the unpurchased products for this user. Such as for user 2 in Table 3.17, the unpurchased products are (0 1 2 3 5), whereas the clicked products are (4).
- 2) Then for each product P1 in the unclicked list, find the association score:

- i. For each product P2 in the clicked list, find the support of (P1->P2) with $SupportC = P(U \cap V) = \frac{Count(U,V)}{Count(All)}$ in the clickstream data records.

Table 3.17: Support from clickstream for user 2

Purchased\Unpurchased	0	1	2	3	5
4	3/6	2/6	3/6	4/6	3/6

- a) Prune the pairs where support is less than M, all the pairs in the example are qualified.
- b) Calculate $LiftC(P1 \rightarrow P2)$ using $LiftC = \frac{Count(U,V) \times Count(All)}{Count(U) \times Count(V)}$ clickstream data for all the qualified pairs, the result is in Table 3.18:

Table 3.18: Lift from clickstream for user 2

Purchased\Unpurchased	0	1	2	3	5
4	18	12	18	24	18

- ii. For each product P3 in the purchase list, find the support of (P1-> P3) with $SupportP = P(U \cap V) = \frac{Count(U,V)}{Count(All)}$ in the purchase data records, such as Table 3.19 for user 2.

Table 3.19: Support from purchases for user 2

Purchased\Unpurchased	0	1	2	3	5
4	1/4	1/4	1/4	2/4	0

- a) Prune the pairs where support is less than M, then only (4->3) is qualified in this step.
- b) Calculate $LiftP(P1 \rightarrow P2)$ using $LiftP = \frac{Count(U,V) \times Count(All)}{Count(U) \times Count(V)}$ purchase data for all the qualified pairs, the result is in Table 3.20:

Table 3.20: Lift from purchases for user 2

Purchased\Unpurchased	3
4	8

- iii. Use the qualified pair from the previous step, calculate final Lift value for clicked product X and product P1, $Lift(U, V) = \alpha * LiftC(U, V) + \beta * LiftP(U, V)$. In this example, we can get the final lift value in Table 3.21:

Table 3.21: Final lift for unpurchased product for user 2

Purchased\Unpurchased	0	1	2	3	5
4	5.4	4	5.4	12.8	5.4

- iv. For all the qualified product X, find the maximum Lift(X,P1) as the association score for P1. In this example, the qualified scores of user 2 are (5.4, 4, 5.4, 12.8, 5.4) for products (0,1,2,3,5) respectively. By normalizing the scores to numbers between 0 and 1, using feature scaling (Equation 3.2) we can have normalized scores (0.16, 0, 0.16, 1, 0.16).
- 6) Modify the original user-item rating matrix with 1 for the qualified ones from the previous step, the enhanced matrix would be like Table 3.22;

Table 3.22: User-item matrix for Kim11Rec

User\Item	0	1	2	3	4	5
0	1	?	1	1	1	?
1	?	1	?	?	?	1
2	0.16	?	0.16	1	1	0.16
3	?	1	?	1	1	?

- 7) Calculate the precision and recall (Table 3.23) for the enriched matrix.

Table 3.23: Evaluation result for Kim11Rec

Method	Precision	Recall
NAN	0.148	0.44
Kim05Rec	0.25	0.75
Kim11Rec	0.2	0.6

3.3.3 Chen13 Method

- 1) For every two users, find the maximum longest sub common (LSC) category sequence between every two click sequences (one from user U and one from user V).

$$\text{Sim1}(U,V)=\text{Max}\{\text{LSC}(U1,V1), \text{LSC}(U1,V2),\dots\};$$

$\text{LSC}(U1,V1)$ = the longest sub common category sequence
 $(U1,V1)/\text{max}(U1.\text{length},V1.\text{length})$;

For example, we take user 0 and user 1 as an example, we can find the category sequences in Table 3.24 from Table 3.8 and Table 3.11, then $\text{Sim1}(0,1)=\text{max}(\text{LSC}("2, 0, 0, 0, 0, 2", "2, 2, 1, 0, 0")/7; \text{LSC}("2, 0", "2, 2, 1, 0, 0")/5)=\text{max}(0.43,0.43) = 0.43$;

Table 3.24: Category sequences from click sequences

UserId	Click sequence	Category sequence
0	4, 2, 2, 0, 2, 3, 4	2, 0, 0, 0, 0, 0, 2
1	4, 5, 1, 0, 3	2, 2, 1, 0, 0
0	5, 3	2, 0

- 2) The similarity of the category number of visits vector; for user V , form a vector (C_1, C_2, C_3, \dots) , where C_1 donates the number of visits from user V on category 1. For user U and user V , find the cosine similarity between two corresponding vectors as Sim2 , for example, user 0 and user 1 has visited category 0, 1, and 2, the visit times are listed in Table 3.25; And the similarity between the category frequency vectors is 0.89.

Table 3.25: Visit frequency on categories

User\ Category	0	1	2
0	6	0	3
1	2	1	2

- 3) The similarity of the category visit time duration vector; for user V , form a vector (T_1, T_2, T_3, \dots) , where T_1 donates the total time duration from user V on category 1; for user U and user V , find the cosine similarity between two corresponding vectors as Sim3 , in our example, we assume all the users spend 10 seconds on each page. Then we can have category duration visit Table 3.26 for user 0 and user 1.

Table 3.26: Visit duration time on categories

User\ Category	0	1	2
0	6	0	3
1	2	1	2

- 4) Find the similarity score $\text{Similarity}(U, V) = \alpha * \text{Sim1} + \beta * \text{Sim2} + \gamma * \text{Sim3}$ between two users using the previous calculated Sim1, Sim2, and Sim3, where $0 < \alpha, \beta, \gamma < 1$, $\alpha + \beta + \gamma = 1$. Set $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.3$ in our example, then we have following results: $\text{Similarity}(0,1) = 0.4 * 0.43 + 0.3 * 0.89 + 0.3 * 0.89 = 0.53$. Similarly, we calculate the similarity between all the users in Table 3.27 for the user-item rating matrix in Table 3.28:

Table 3.27: New similarity matrix

User\User	0	1	2	3
0		0.53	0.58	0.55
1			0.62	0.59
2				0.64
3				

Table 3.28: User-item rating matrix

User\Item	0	1	2	3	4	5
0	1	?	1	1	1	?
1	?	1	?	?	?	1
2	?	?	?	?	1	?
3	?	1	?	1	1	?

- 8) Select top 4 data such as (0->2, 1->2, 1->3, 2->3) and modify similarity during collaborative if the new similarity is stronger. Then calculate the precision and recall, the result is in Table 3.29.

Table 3.29: Evaluation result for Chen 13

Method	Precision	Recall
NAN	0.148	0.44
Kim05Rec	0.25	0.75
Kim11Rec	0.2	0.6
Chen13Rec	0.148	0.444

3.3.4 HPCRec

We firstly pre-process clickstream (Table 3.8) and transaction (Table 3.9) to get the consequential table (Table 3.30), and transaction (Table 3.9) to get user-item purchase frequency table (Table 3.31). Then use these two processed tables as input to run through Algorithm 3.1:

Table 3.30: Consequential Table

Purchased	SessionId	UserId	Clicks	Purchases
Y	0	0	< 4,2,2,0,2,3,4>	3,2,0,4,4
	3	1	< 4,5,1,0,3>	1,1,5
	4	2	< 0,4,4,0>	4
	6	3	< 3,1,3,4,5,2>	1,4,3,3,3
N	1	0	<5,3>	
	5	2	< 4,3,5,2>	

Table 3.31: User-item purchase frequency Table

User\Item	0	1	2	3	4	5
0	1	?	1	1	2	?
1	?	2	?	?	?	1
2	?	?	?	?	1	?
3	?	1	?	3	1	?

- (1) Normalize the user-item purchase frequency table (Table 3.31) through the function NF in section 3.2.1 , and get normalized purchase frequency table (Table 3.32):

Table 3.32: Normalized purchase frequency table

User\Item	0	1	2	3	4	5
0	0.38	?	0.38	0.38	0.76	?
1	?	0.89	?	?	?	0.45
2	?	?	?	?	1	?
3	?	0.3	?	0.9	0.3	?

- (2) For each sequence without purchases (N) in Table 3.30, calculate the similarity between it and every sequences with purchases (Y) in Table 3.30 using function CSSM

in section 3.2.2, the result is in Table 3.33. To remove the long tail, we prune the record when similarity is less than 0.1 and generate a weighted transaction table (Table 3.34) for the next step;

- Calculate $LCSR("4,3,5,2", "4,2,2,0,2,3,4") = LCS("4,3,5,2", "4,2,2,0,2,3,4") / \max(4,7) = 2/7 = 0.29$.
- Calculate $FS("4,3,5,2", "4,2,2,0,2,3,4") = \cosSim(\langle 001111 \rangle, \langle 110111 \rangle) = 0.77/7 = 0.11$; where 001111 and 110111 are the frequency vectors for product 0, 1, 2, 3, 4 and 5.
- Use α and β as parameters to balance the sub sequence similarity and frequency similarity, where $0 < \alpha, \beta < 1, \alpha + \beta = 1$. α and β will be determined from training dataset. So if set $\alpha = 0.8, \beta = 0.2$, $Sim("4,3,5,2", "4,2,2,0,2,3,4") = 0.8 * 0.29 + 0.2 * 0.11 = 0.25$;
- Similarly, we can have following similarity result in Table 3.33:

Table 3.33: An example of the similarity between clickstream sequences

Clickstream without purchase	Clickstream with purchases	Similarity
< 5,3>	< 4,2,2,0,2,3,4>	0.12
	< 4,5,1,0,3>	0.35
	< 0,4,4,0>	0
	< 3,1,3,4,5,2>	0.16
<4,3,5,2>	< 4,2,2,0,2,3,4>	0.25
	< 4,5,1,0,3>	0.35
	< 0,4,4,0>	0.22
	< 3,1,3,4,5,2>	0.43

Table 3.34: A table of weighted transactions for user 0 and 2

uid	sid	clickstream	purchase	weight
0	0	<4,2,2,0,2,3,4>	3,2,0,4,4	0.12
	3	< 4,5,1,0,3>	1,1,5	0.35
	6	< 3,1,3,4,5,2>	1,4,3,3,3	0.16
2	0	< 4,2,2,0,2,3,4>	3,2,0,4,4	0.25
	3	< 4,5,1,0,3>	1,1,5	0.35
	4	< 0,4,4,0>	4	0.22
	6	< 3,1,3,4,5,2>	1,4,3,3,3	0.43

- (3) Use TWFI function in section 3.2.3 to calculate weighted frequency for items in Table 3.35. Prune the items that have been purchased or weighted support (WS) is less than a minimum weighted support (mws=0.3) (shaded items in Table 3.36) by a user. Eg., $ws(1)$ for user “0” is $(0.35+0.35+0.16)/3 * 3=0.86$, other results are in Table 3.35; and the normalized weighted frequencies are in Table 3.36;

Table 3.35: A result of weighted frequent items

User\item	0	1	2	3	4	5
0	0.12	0.86	0.12	0.6	0.4	0.35
2	0.25	1.13	0.25	1.54	0.93	0.35

Table 3.36: Normalized weighted frequent items

User\item	0	1	2	3	4	5
0	0.18	1	0	0.65	0.38	0.31
2	0	0.68	0	1	0.53	0.08

- (4) Select qualified items (weighted support are 1, 0.31, 0.68, 1) from the previous step and fill the original user-item matrix with them (change all the non-zero values to 1 instead for experimental purpose), the new user-item matrix is as in Table 3.37.

Table 3.37: Enriched user-item matrix from HPCRec

User\Item	0	1	2	3	4	5
0	0.38	1	0.38	0.38	0.76	0.31
1	?	0.89	?	?	?	0.45
2	?	0.68	?	1	1	?
3	?	0.3	?	0.9	0.3	?

(5) Calculate the precision and recall for both of the enriched matrix and the original matrix, the comparison table is Table 3.38.

Table 3.38: Evaluation result of HPCRec

Method	Precision	Recall
NAN	0.148	0.44
Kim05Rec	0.25	0.75
Kim11Rec	0.2	0.6
Chen13Rec	0.148	0.444
HPCRec	0.308	0.923

3.4 An Example Application of Proposed Algorithm

Take the input data as following in Table 3.39, Table 3.40, Table 3.41 and Table 3.42 as input data, we run through Kim05Rec (Kim, Yum, Song, & Kim, 2005), Kim11Rec (Kim & Yum, 2011), Chen13Rec (Chen & Su, 2013) and HPCRec respectively to prove that HPCRec is only one being able to detect the rare cases and give good recommendations for these scenarios.

Table 3.39: Clickstream data for a walk through example

sid	uid	clickstream	tid	sstart (yyyy.MM.dd.hh.mm.ss)	send (yyyy.MM.dd.hh.mm.ss)
1	1	<abcdade>	1	2017.09.05.13.23.30	2017.09.05.13.43.00
2	2	<cdea>	2,3	2017.06.15.09.00.34	2017.06.15.09.50.20
3	3	<ddcabe>	4	2017.03.05.18.53.19	2017.03.05.19.33.14
4-98	4-98	<abce>	5-99	2017.03.05.18.53.19	2017.03.05.19.33.14
99	99	<fgh>	100	2017.09.25.15.23.22	2017.09.25.16.23.15
100	100	<ggg>		2017.10.25.15.23.22	2017.10.25.16.23.15

Table 3.40: Purchase data for a walk through example

tid	uid	purchase	time (yyyy.MM.dd.hh.mm.ss)
1	1	{ace}	2017.09.05.13.23.30
2	2	{cdd}	2017.06.15.09.23.34
3	2	{a}	2017.06.15.09.30.34
4	3	{dae}	2017.03.05.18.59.19
5-99	4-99	{ace}	2017.03.05.18.59.19
100	99	{fg}	2017.09.25.15.56.22

Table 3.41: Rating matrix

User\Item	a	b	c	d	e	f	g	h
1	1		1		1			
2	1		1	1				
3	1			1	1			
4-98	1		1		1			
99						1	1	
100								

Table 3.42: Product metadata Table

itemId	itemName	price	category
1	a	6	1
2	b	3	2
3	c	3	2
4	d	4	1
5	e	7	3
6	f	23	4
7	g	3	1
8	h	4	4

3.4.1 Kim05Rec Method

- 1) Find the valuable indicators for each item, such as number of visits, length of reading time, basket placement, purchase, etc. In this example, we take number of visits and

basket placement (assuming all the purchased products were placed in the basket before) as our variables, and get following distributed decision Table 3.43:

Table 3.43: Decision tree example

itemId	itemName	visits<2	Visits>2	Basket placement when visits<2	Basket placement when visits>=2
1	a	97	1	97/97	1/1
2	b	97	0	0/97	0/0
3	c	98	0	97/98	0/0
4	d	1	2	1/1	1/2
5	e	98	0	97/98	0/0
6	f	1	0	1/1	0/0
7	g	1	0	1/1	0/0
8	h	1	0	0/1	0/0

- 2) Take the clicked but not purchased session and find the estimated data from Table 3.39. Such as in this case, session 100 has clicks “ggg”, match to basket placement when (visits of g >= 2), we get 0/0 which means there is no such case to find an estimated possibility for placing g in to the basket after clicking three times.
- 3) Then the modified user-item rating matrix would be the same as the original one in Table 3.41. For the new user 100, even with all the clicks, this method cannot predict the user’s interests.

3.4.2 Kim11 Method

(M=0.1 is the support threshold, N=2 is the lift threshold, $\alpha+\beta=1$, $0<\alpha$, $\beta<1$):

- 1) For a given user, find all the unclicked products for this user. Such as for user 100 in Table 3.39, the unclicked products are (a,b,c,d,e,f,h), whereas the clicked product is only “g”.
- 2) Then for each product P1 in the unclicked list, find the association score:
 - i. For each product P2 in the clicked list, find the support of (P1->P2) with $SupportC = P(U \cap V) = \frac{Count(U,V)}{Count(All)}$ in the clickstream data records as in Table 3.44.

Table 3.44: Support from clickstream data

clickedItem\UnClickedItem	a	b	c	d	e	f	h
g	0	0	0	0	0	1/100	1/100

- a) Prune the pairs where support is less than M, then there is no qualified pairs for item “g”.
- ii. For each product P2 in the purchase list, find the support of (P1->P2) with $SupportP = P(U \cap V) = \frac{Count(U,V)}{Count(All)}$ in the purchase data records as in Table 3.45.

Table 3.45: Support from purchase data

clickedItem\UnPurchasedItem	a	b	c	d	e	f	h
g	0	0	0	0	0	1/100	0

- a) Prune the pairs where support is less than M, then there is no qualified pairs for item “g”.
- iii. Calculate final Lift value for clicked product X and product P1, $Lift(U, V) = \alpha * LiftC(U, V) + \beta * LiftP(U, V)$. In this example, there are no qualified LiftC and LiftP, so there is no qualified lift value.
- iv. For all the qualified product X, find the maximum Lift(X,P1) as the association score for P1. In this example, there is no qualified scores for user 100.
- 3) Then there will not be any new useful value for user 100 mined from the association rules, so the matrix will be the same as the original user-item matrix.

3.4.3 Chen13 Method

- 1) Find the maximum longest sub common (LSC) category sequence between every two click sequences (one from user U and one from user V).

$$Sim1(U,V)=Max\{LSC(U1,V1), LSC(U1,V2), \dots\};$$

$$LSC(U1,V1)= \text{the longest sub common category sequence} \\ (U1,V1)/\max(U1.length, V1.length);$$

For example, we take user 100 as an example,

$$Sim1(1,100)=\max(LSC(“abcdade”, “ggg”)/\max(7,3))=\max(LSC(1221113,111)/7)=3/7;$$

$$\text{Sim1}(2,100)=\max(\text{LSC}(\text{"cdea"}, \text{"ggg"})/\max(4,3))=\max(\text{LSC}(2113,111)/4)=2/4;$$

$$\text{Sim1}(3,100)=\max(\text{LSC}(\text{"ddcabe"}, \text{"ggg"})/\max(6,3))=\max(\text{LSC}(112123,111)/6)=3/6;$$

$$\text{Sim1}(4-98,100)=\max(\text{LSC}(\text{"ace"}, \text{"ggg"})/\max(3,3))=\max(\text{LSC}(123,111)/3)=1/3;$$

$$\text{Sim1}(99,100)=\max(\text{LSC}(\text{"fgh"}, \text{"ggg"})/\max(3,3))=\max(\text{LSC}(414,111)/3)=1/3;$$

- 2) The similarity of the category number of visits vector; for user V , form a vector (C_1, C_2, C_3, \dots) , where C_1 donates the number of visits from user V on category 1; for user U and user V , find the cosine similarity between two corresponding vectors as Sim2 , for example, for user 100,

$$\text{Sim2}(1,100)=\cos\text{Sim}(421, 300)=0.87;$$

$$\text{Sim2}(2,100)=\cos\text{Sim}(211, 300)=0.82;$$

$$\text{Sim2}(3,100)=\cos\text{Sim}(321, 300)=0.80;$$

$$\text{Sim2}(4-98,100)=\cos\text{Sim}(111, 300)=0.58;$$

$$\text{Sim2}(99,100)=\cos\text{Sim}(1002, 3000)=0.45;$$

- 3) The similarity of the category visit time duration vector; for user V , form a vector (T_1, T_2, T_3, \dots) , where T_1 donates the total time duration from user V on category 1; for user U and user V , find the cosine similarity between two corresponding vectors as Sim3 , in our example, we assume all the users spend 10 seconds on each page. Then we can have following relationship for user 100.

$$\text{Sim3}(1,100)=\cos\text{Sim}(40\ 20\ 10, 30\ 0\ 0)=0.87;$$

$$\text{Sim3}(2,100)=\cos\text{Sim}(20\ 10\ 10, 30\ 0\ 0)=0.82;$$

$$\text{Sim3}(3,100)=\cos\text{Sim}(30\ 20\ 10, 30\ 0\ 0)=0.80;$$

$$\text{Sim3}(4-98,100)=\cos\text{Sim}(10\ 10\ 10, 30\ 0\ 0)=0.58;$$

$$\text{Sim3}(99,100)=\cos\text{Sim}(10\ 0\ 0\ 20, 30\ 0\ 0\ 0)=0.45;$$

- 4) Find the similarity score $\text{Similarity}(U, V) = \alpha * \text{Sim1} + \beta * \text{Sim2} + \gamma * \text{Sim3}$ between two users using the previous calculated Sim1 , Sim2 , and Sim3 , where $0 < \alpha, \beta, \gamma < 1$, $\alpha + \beta + \gamma = 1$. Set $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.3$ in our example, then we have following results:

$$\text{Similarity}(1,100)=0.4*3/7+0.3*0.87+0.3*0.87=0.693;$$

$$\text{Similarity}(2,100)= 0.4*2/4+0.3*0.82+0.3*0.82=0.0.692;$$

$$\text{Similarity}(3,100)= 0.4*3/6+0.3*0.8+0.3*0.8=0.68;$$

$$\text{Similarity}(4-98,100)= 0.4*1/3+0.3*0.58+0.3*0.58=0.48;$$

$$\text{Similarity}(99,100)= 0.4*1/3+0.3*0.45+0.3*0.45=0.4;$$

- 5) In the conventional collaborative filtering algorithm, we can use this similarity between users to detect the potential interest instead using the similarity between purchase vectors, therefore we can narrow the matrix down the chosen similar users. But in this example, the most possible product for user 100 to buy is product “g”, then product “f”. But from this method, we can notice that user 99 the least similar to user 100, which will be filtered out for neighbor selection, so instead, the user is mostly likely to get products previously purchased by user 1, 2, 3 as recommendations.

3.4.4 HPCRec

We firstly pre-process clickstream (Table 3.39) and transaction (Table 3.40) to get the consequential table (Table 3.46), and transaction (Table 3.40) to get user-item purchase frequency table (Table 3.47). Then use these two processed tables as input to run through Algorithm 3.1:

Table 3.46: Consequential Table

Purchased	SessionId	UserId	Clicks	Purchases
Y	1	1	<abcdade>	{ace}
	2	2	<cdea>	{acdd}
	3	3	<ddcabe>	{dae}
	4-98	4-98	<abce>	{ace}
	99	99	<fgh>	{fg}
N	100	100	<ggg>	

Table 3.47: User-item purchase frequency table

User\Item	a	b	c	d	e	f	g	h
1	1		1		1			
2	1		1	2				
3	1			1	1			
4-98	1		1		1			
99						1	1	
100								

- 1) Normalize the user-item purchase frequency table (Table 3.47) through the function NF in section 3.2.1 , and get normalized purchase frequency table (Table 3.48):

Table 3.48: Normalized user-item purchase frequency table

User\Item	a	b	c	d	e	f	g	h
1	1		1		1			
2	0.41		0.41	0.82				
3	1			1	1			
4-98	1		1		1			
99						1	1	
100								

- 2) For each sequence without purchases (N) in Table 3.46, calculate the similarity between it and every sequence with purchases (Y) using function CSSM in section 3.2.2. In this example, we calculate the similarity between “ggg” and others, following steps are for sequences “ggg” and “abcdade”. And we only can only form a weighted transaction for user 100 which is <fg:0.506>;
- Calculate $LCSR(\text{“ggg”}, \text{“abcdade”}) = LCS(\text{“ggg”}, \text{“abcdade”}) / \max(3, 7) = 0$, where 3 is the length of “ggg”, and 7 is the length of “abcdade”.
 - Calculate $FS(\text{“ggg”}, \text{“abcdade”}) = \cosSim(\langle 000001 \rangle, \langle 211210 \rangle) = 0$; where 000001 and 211210 are the frequency vectors for product a, b, c, d, e and g.
 - Use α and β as parameters to balance the sub sequence similarity and frequency similarity, where $0 < \alpha, \beta < 1, \alpha + \beta = 1$. α and β will be determined from training dataset. So if set $\alpha = 0.3, \beta = 0.7, Sim(\text{“ggg”}, \text{“abcdade”}) = 0.3 * 0 + 0.6 * 0 = 0$;
 - Similarly, we can have following similarity result:

$$\text{Sim}(\text{"ggg"}, \text{"cdea"})=0;$$

$$\text{Sim}(\text{"ggg"}, \text{"ddcabe"})=0;$$

$$\text{Sim}(\text{"ggg"}, \text{"cdea"})=0;$$

$$\text{Sim}(\text{"ggg"}, \text{"abce"})=0;$$

$$\text{Sim}(\text{"ggg"}, \text{"fgh"})=0.3*\text{LCS}(\text{"ggg"}, \text{"fgh"})/3 + 0.7*\text{cosSim}(\langle 030 \rangle, \langle 111 \rangle) = 0.3*1/3 + 0.7*0.58=0.506;$$

- 3) Use TWFI function in section 3.2.3 to find weighted frequent items for the weighted transaction $\langle fg:0.506 \rangle$, which are $\langle f:0.506 \rangle$ and $\langle g:0.506 \rangle$.
- 4) We successfully found the interest from user 100 on product g after viewing three times, then fill the original user-item matrix with the weighted frequency from the previous step for further purpose; the new user-item matrix is as in Table 3.49.

Table 3.49: Enriched user-item matrix

User\Item	a	b	c	d	e	f	g	h
1	1		1		1			
2	1		1	1				
3	1			1	1			
4-98	1		1		1			
99						1	1	
100						0.506	0.506	

CHAPTER 4

EXPERIMENTS EVALUATION AND ANALYSIS

We have implemented Kim05Rec (Kim, Yum, Song, & Kim, 2005), Kim11Rec (Kim & Yum, 2011) and Chen13Rec (Chen & Su, 2013) and our recommendation system HPCRec. To make sure the evaluation is fair, we only select the top N (N is productNumber/10, this is also a variable with different results in section 4.3) scores from different approaches, and give 1 as the rating for the user on the corresponding product to keep the measuring standard consistent. Then we feed the new rating matrix to an evaluation method of an existing recommendation library [Librec](#) (Guo, Zhang, Sun, & Yorke-Smith, 2015) to test all the approaches. For Chen13Rec (Chen & Su, 2013), we use the measured relationship to enhance the similarity table during the evaluation. For HPCRec, we ran through using both average weighted and maximum weighted support strategies in module TWFI (section 3.2.3).

4.1 Dataset and Sample Selection

We use the dataset provided by YOOCHOOSE GmbH for [ACM RecSys 2015](#) (Ben-Shimon, et al., 2015), which is from an online retailer in Europe. There are two files recording 33,040,175 clicks and 1,177,769 purchase events respectively; all the events happened in 9,512,786 unique sessions, the total amount of product is 52,739 belonging to 339 categories. For sample selection, we randomly select a certain amount of session 10 times and use the average value. For each time, given the lack of user information in the dataset, we generate a reasonable number of user and assign to the sessions randomly then use the average value of 10 times attempts. It has been proven that regardless of how users are distributed in sessions, our methods are better.

4.2 Evaluation Metrics

We evaluate with both user-based and item-based recommendations. Itemknn selections and PCC similarity method are used for item-based evaluation, userknn and cosine similarity method are for user-based evaluation. Different evaluation measurements in Librec are used: AUC (Area Under the Curve); AP (Average Precision); Precision; Recall; RR (Reciprocal Rank); NDCG (Normalized DCG). In this thesis, we use evaluation measurements AP (Average Precision), Precision and Recall.

Precision. The output of HPCRec system are a list of selected top-N recommendations, which can be formed as (user, item) representing recommending product “item” to user “user”. Among all these recommendations which are referred as positive recommendation (tp means the number of recommendations that are preferred by the users, fp represents the amount of the ones are not appreciated by the users) in Table 4.1, precision measures the proportion of preferred recommendations (Equation 4.1). For example, if a recommendation system recommend products (1,2,3,4,5) to users (a,b,c,d,e) respectively, and only user a,b,c preferred the recommendations, so in this case tp (true-positive) is 3, and fp (false-positive) is 2, therefore, the precision equals to $3/(3+2)$ which is $3/5$.

Equation 4.1: Precision Formula

$$Precision = \frac{tp}{tp + fp}$$

Table 4.1: Confusion Matrix

	Recommended (Positive)	Not recommended (Negative)
Preferred (True)	tp	fn
Not preferred (False)	fp	tn

Recall. A user may have not purchased but is interested in some products, if a recommendation system successfully recommend some of these products in its top-N recommendations, then we use tp (true-positive in Table 4.1) to represent the number of these product, for the products a user prefers but the recommendation system failed to recommend, we use fn (false-negative in Table 4.1) to hold the amount of preferred but not recommended products. In these potential interesting products users like, recall represents the proportion of products that the recommendations can predict in Equation 4.2. For example, users (a,b,c,d,e) prefer unpurchased products (1,2,3,4,5) respectively, if a recommendation system recommend products (1,2,3) to users (a,b,c) respectively and failed to recommend products (4,5) to users (d,e), so in this case tp (true-positive) is 3, and fn (false-negative) is 2, therefore, the recall equals to $3/(3+2)$ which is $3/5$.

Equation 4.2: Recall Formula

$$Recall = \frac{tp}{tp + fn}$$

Average Precision. Precision measures the accuracy for a selected group of recommendations from recommendation system by applying a threshold or top-N parameter, whereas average precision measures the precision with different thresholds and calculate the average precision. In Equation 4.3, where relevance(i) is 1 if relevant, and 0 if not. For instance, if with different top-N (1,2,3,4,5), the precisions are (0.5,0.6,0.3,0.85, 0) respectively, then the average precision is $(0.5 + 0.6 + 0.3 + 0.85 + 0) * 1/5 = 0.45$.

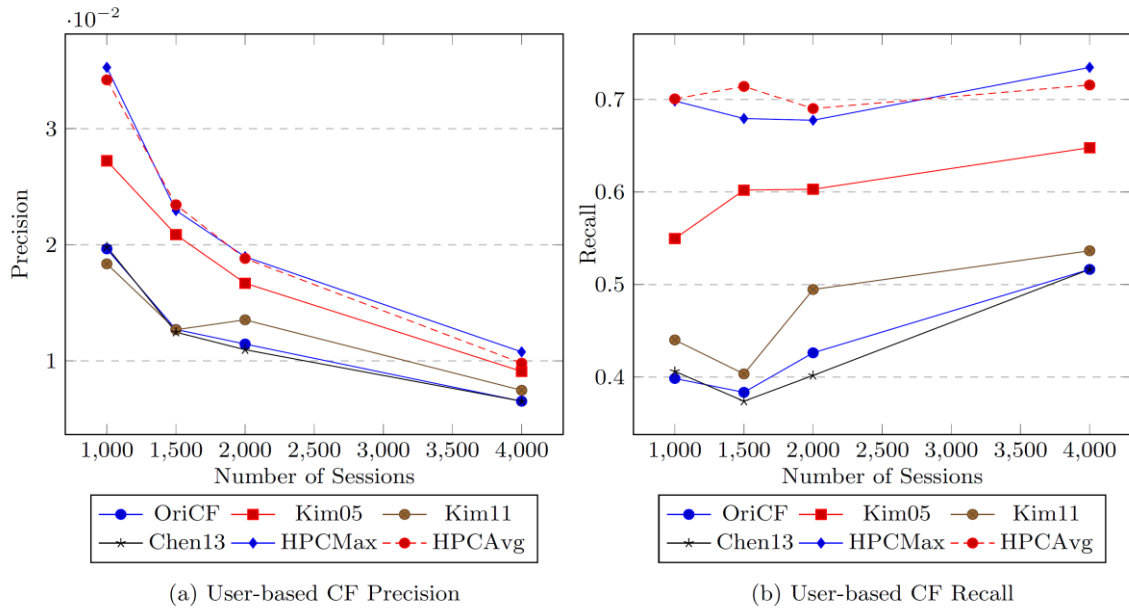
Equation 4.3: Average Precision Formula

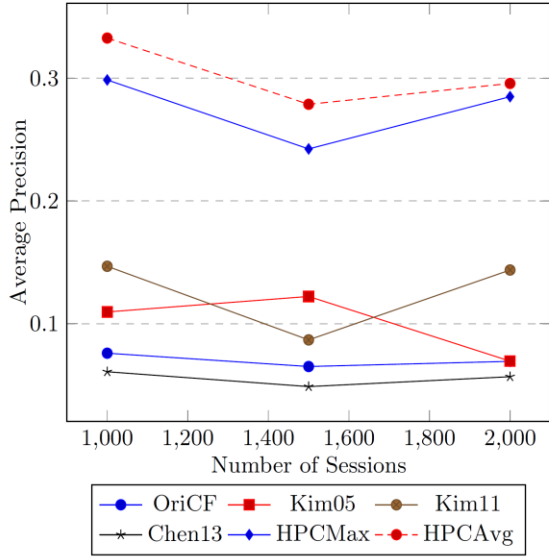
$$AP(\text{Average Precision}) = \frac{1}{|R|} \cdot \sum_{i=1}^n \text{precision}(i) \cdot \text{relevance}(i);$$

4.3 Evaluation Result and Analysis

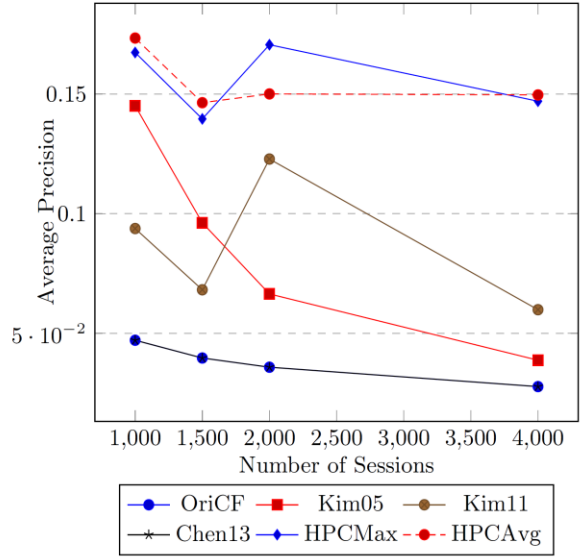
From both user-based and item-based CF evaluation results on varying numbers of sessions (Figure 4.1), we can see the accuracy keeps dropping as the amount of sessions increases; our approaches are still better in this respect. For average accuracy and recall, our methods significantly beat others.

We select a different number of top-N scores from all of the methods for calculation and evaluation (Figure 4.2). Both user-based CF and item-based CF are still the best, which proves the high quality of our scores. Kim05Rec (Kim, Im, & Atluri, 2005) also demonstrates good performance.

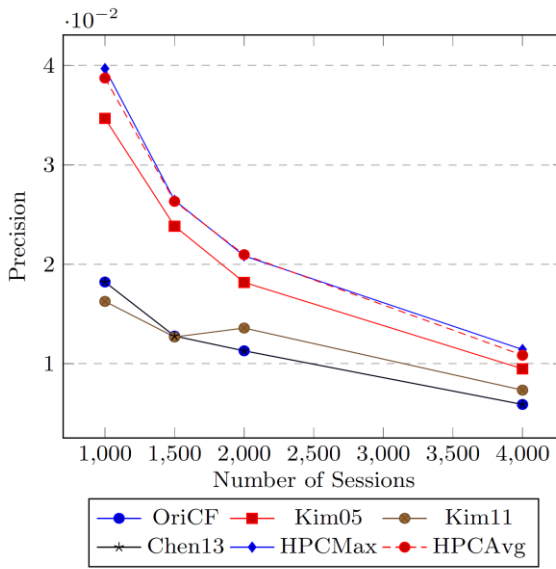




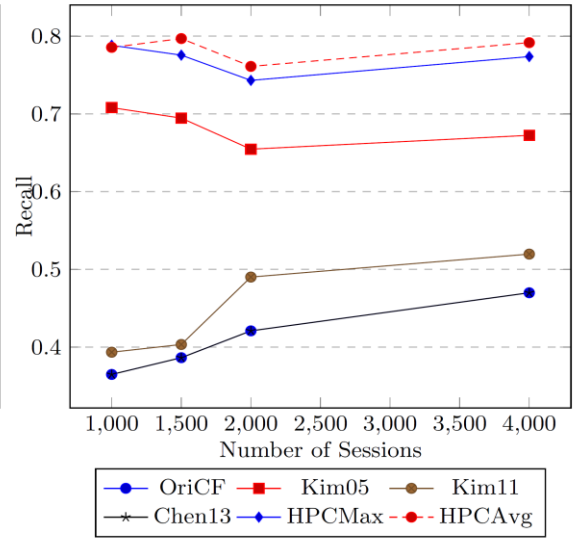
(c) User-based CF Average Precision



(d) Item-based CF Average Precision

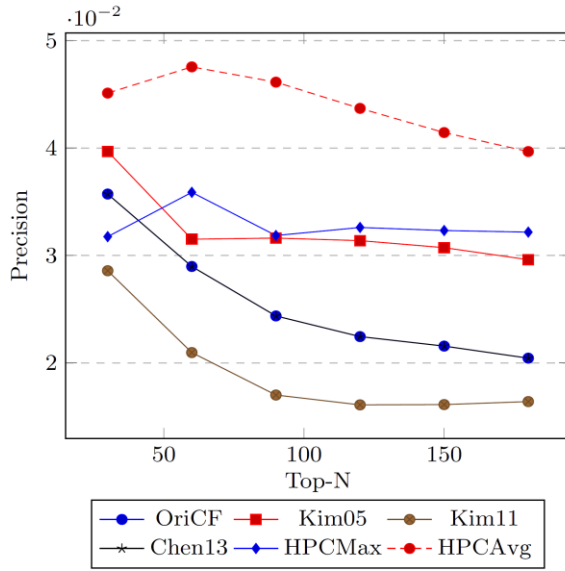


(e) Item-based CF Precision

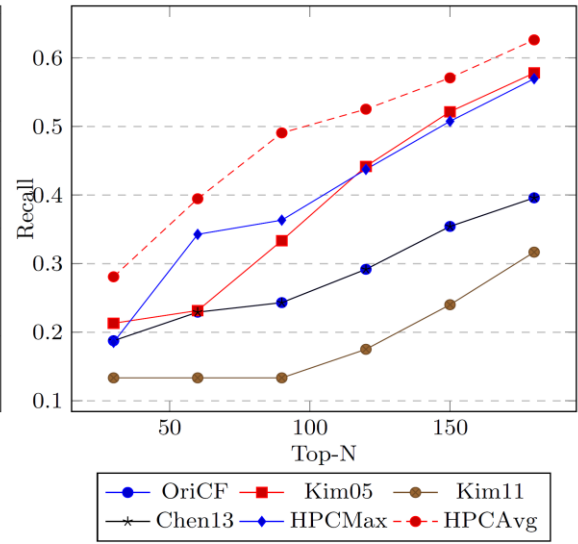


(f) Item-based CF Recall

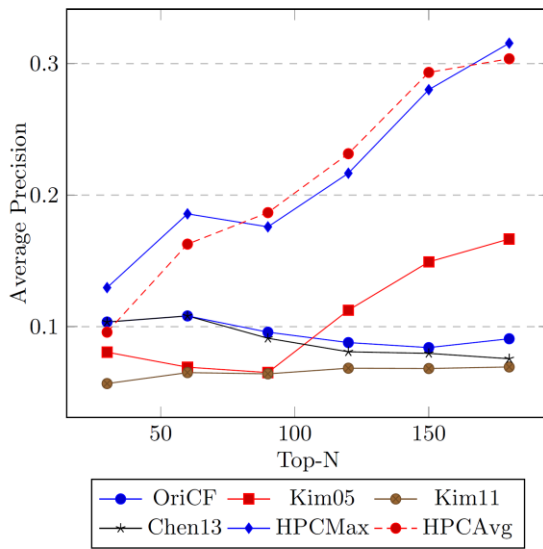
Figure 4.1: Evaluation on different number of sessions



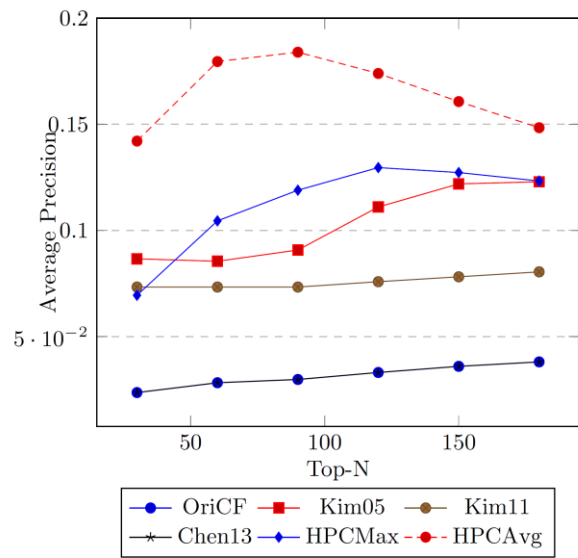
(a) User-based CF Precision



(b) User-based CF Recall



(c) User-based CF Average Precision



(d) Item-based CF Average Precision

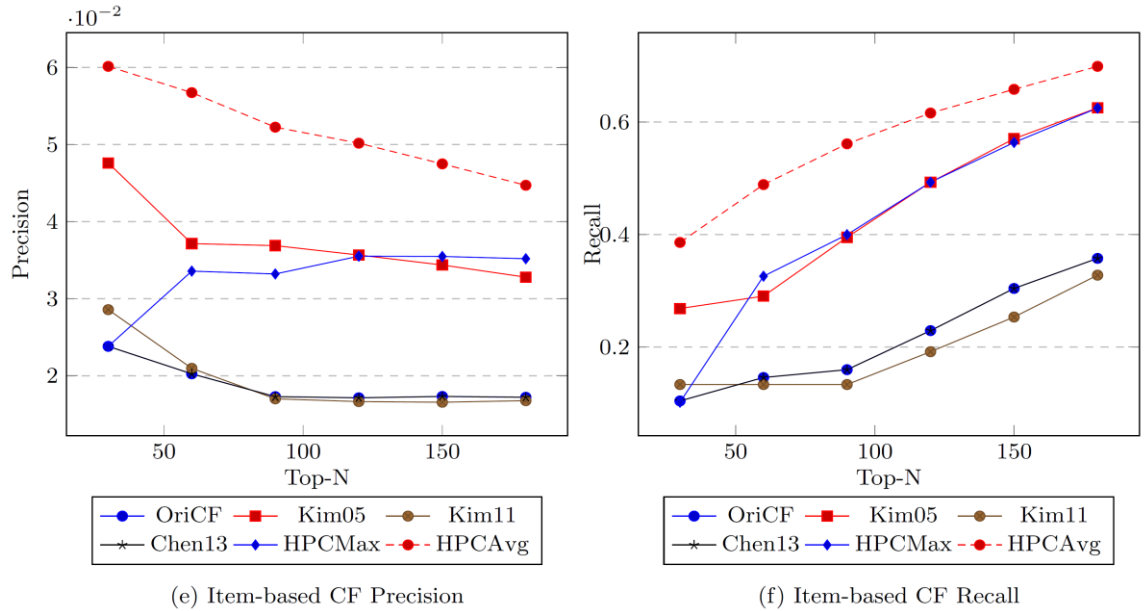


Figure 4.2: Evaluation on different number of top-N scores

4.4 Implementation and Code

4.4.1 Develop Environment and Tools

- Operation system: Windows 10 Pro
 - RAM: 16 GB
 - CPU: 3.6 GHz
 - System type: 64-bit Operating System, x64 based processor
- Develop software: Eclipse Java EE IDE for Web Developers
 - Version: Oxygen.1a Release (4.7.1a)
 - Build id: 20171005-1200
- Platform: Java SE Development Kit
 - Version: 1.8.0_65
- Project manage tool: Apache Maven
 - Version: 3.5.3

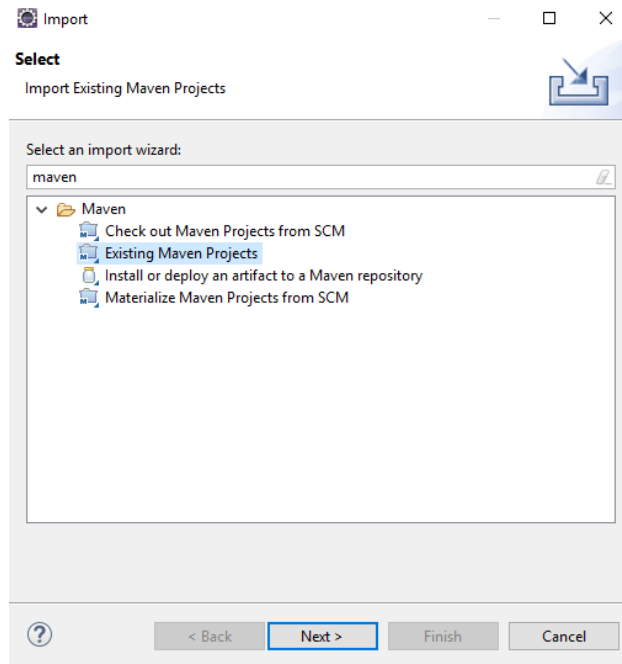
4.4.2 Deploy Environment and Tools

- Operation system: Linux
 - Architecture: x86_64
 - CPU op-mode(s): 32-bit, 64-bit
 - Byte Order: Little Endian

- CPU(s): 16
- Model name: AMD Opteron(tm) Processor 6320
- CPU MHz: 1400.000
- Platform: Java SE Development Kit
 - Version: 1.8.0_65
- Project manage tool: Apache Maven
 - Version: 3.5.3

4.4.3 Setup Development

- (1) Download folder “librec-librec-src-v2.0” onto your local pc from “/home/woddlab/Ying” on woddlab Linux server; test data is in folder “librec-librec-src-v2.0/yoochoose-data”
- (2) Install Jdk and eclipse;
- (3) In your eclipse, click File->Import->Existing Maven Projects, and browse to the downloaded folder “librec-librec-src-v2.0”, click finish and you will see the project in your Project Explorer.



- (4) Find the Junit test cases in folder “librec-librec-src-v2.0\core\src\test\java\generatedataRefactored” and right click->run as->Junit test.

4.4.4 Run on Linux Server

- (1) Check Maven and Java version using cmd command lines “Java –version” and “Mvn -version”, if got exceptions, then run following command to make sure Java and Maven are installed:

```
M2_HOME="/home/woddlab/Ying/Maven/apache-maven-3.5.3"
```

```
PATH=/home/woddlab/Ying/Maven/apache-maven-3.5.3/bin:$PATH
```

```
JAVA_HOME="/home/woddlab/Ying/Java/jdk1.8.0_111"
```

```
PATH=/home/woddlab/Ying/Java/jdk1.8.0_111/bin:$PATH
```

- (2) Change directory to “/home/woddlab/Ying/librec-librec-src-v2.0”, and run Junit tests with Maven:

- a. For a single test case: `mvn -Dtest=generatedataRefactored.YooseTest2 test`
- b. For all the test cases: `mvn test`

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this thesis, we have proposed HPCRec recommendation system, which enhances the known rating quality in the matrix by integrating historical purchases to change rating values from 1 to a reasonable number which can reflect how much a user likes a product relatively; also by mining the consequential relationship between session-based clicks and purchases, it predicts potential purchasing possibilities for products a user has clicked but not purchased, then through enriching the user-item rating matrix with the possibilities for the unknown ratings, HPCRec improves the quantity of ratings for the input matrix. HPCRec is capable of generating recommendations for infrequent users after above improvements whereas Kim05Rec, Kim11Rec and Chen13Rec can not. By performing a session-based collaborative purchasing interest probability calculation, HPCRec improved recommendation accuracy and proved the session-based consequential bond is stronger. Our experimental results show that HPCRec outperform above existing systems referred in this thesis.

We give some ideas and directions of potential extensions for future work:

- I. Mine more information out of the historical data to improve recommendations such as how long ago a user purchased an item and the frequent sequential purchase patterns.
- II. Incorporate multiple data sources with different data schema, and make recommendations based on the overall data set.
- III. Integrate HPCRec to the real online recommendation system, and use the recommendation accept rate to automatically tune the parameters and optimize the recommendation accuracy.
- IV. Given the fact that the input user-item rating matrix is not sparse anymore after processing with our method, use techniques such as dimensionality reduction (eg., Singular Value Decomposition), remove insignificant users or items to reduce the dimensions directly, or other additional techniques for collaborative filtering.

REFERENCES/BIBLIOGRAPHY

- Aggarwal, C. C. (2016). An introduction to recommender systems. In *Recommender Systems* (pp. 1-28). Springer.
- Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, (pp. 3-14).
- Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *Acm sigmod record*, 22, pp. 207-216.
- Agrawal, R., Srikant, R., & others. (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB, 1215*, pp. 487-499.
- Aguiar, L., & Martens, B. (2016). Digital music consumption on the internet: evidence from clickstream data. *Information Economics and Policy*, 34, 27-43.
- Anyanwu, M. N., & Shiva, S. G. (2009). Comparative analysis of serial decision tree classification algorithms. *International Journal of Computer Science and Security*, 3, 230-240.
- Babin, B. J., & Darden, W. R. (1995). Consumer self-regulation in a retail environment. *Journal of retailing*, 71, 47-70.
- Babin, B. J., Darden, W. R., & Griffin, M. (1994). Work and/or fun: measuring hedonic and utilitarian shopping value. *Journal of consumer research*, 20, 644-656.
- Ben-Shimon, D., Tsikinovsky, A., Friedmann, M., Shapira, B., Rokach, L., & Hoerle, J. (2015). Recsys challenge 2015 and the yoochoose dataset. *Proceedings of the 9th ACM Conference on Recommender Systems*, (pp. 357-358).
- Berry, M. J., & Linoff, G. (1997). *Data mining techniques: for marketing, sales, and customer support*. John Wiley & Sons, Inc.
- Bozkaya, T., Yazdani, N., & Özsoyoğlu, M. (1997). Matching and indexing sequences of different lengths. *Proceedings of the sixth international conference on Information and knowledge management*, (pp. 128-135).
- Bucklin, R. E., & Sismeiro, C. (2003). A model of web site browsing behavior estimated on clickstream data. *Journal of marketing research*, 40, 249-267.
- Bucklin, R. E., & Sismeiro, C. (2009). Click here for Internet insight: Advances in clickstream data analysis in marketing. *Journal of Interactive Marketing*, 23, 35-48.

- Bucklin, R. E., Lattin, J. M., Ansari, A., Gupta, S., Bell, D., Coupey, E., . . . Steckel, J. (2002). Choice and the Internet: From clickstream to research stream. *Marketing Letters*, 13, 245-258.
- Capelle, M., Masson, C., & Boulicaut, J.-F. (2002). Mining frequent sequential patterns under a similarity constraint. *International Conference on Intelligent Data Engineering and Automated Learning*, (pp. 1-6).
- Chen, L., & Su, Q. (2013). Discovering user's interest at E-commerce site using clickstream data. *Service systems and service management (ICSSSM), 2013 10th international conference on*, (pp. 124-129).
- Chen, M.-S., Han, J., & Yu, P. S. (1996). Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering*, 8, 866-883.
- Chiang, R.-D., Wang, Y.-H., & Chu, H.-C. (2013). Prediction of members' return visit rates using a time factor. *Electronic Commerce Research and Applications*, 12, 362-371.
- Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22, 143-177.
- Fan, J., Pan, W., & Jiang, L. (2014). An improved collaborative filtering algorithm combining content-based algorithm and user activity. *Big Data and Smart Computing (BIGCOMP), 2014 International Conference on*, (pp. 88-91).
- Gündüz, Ş., & Özsu, M. T. (2003). A web page prediction model based on click-stream tree representation of user behavior. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 535-540).
- Guo, G., Zhang, J., Sun, Z., & Yorke-Smith, N. (2015). LibRec: A Java Library for Recommender Systems. *UMAP Workshops*, 4.
- Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. *ACM sigmod record*, 29, pp. 1-12.
- Hartigan, J. A., & Wong, M. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28, 100-108.
- Hipp, J., Güntzer, U., & Nakhaeizadeh, G. (2000). Algorithms for association rule mining—a general survey and comparison. *ACM sigkdd explorations newsletter*, 2, 58-64.

- Hunt, J. W., & MacIlroy, M. D. (1976). *An algorithm for differential file comparison*. Bell Laboratories Murray Hill.
- Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM computing surveys (CSUR)*, 31, 264-323.
- Jones, M. A., Reynolds, K. E., & Arnold, M. J. (2006). Hedonic and utilitarian shopping value: Investigating differential effects on retail outcomes. *Journal of Business Research*, 59, 974-981.
- Kim, D.-H., Im, I., & Atluri, V. (2005). A clickstream-based collaborative filtering recommendation model for e-commerce. *E-Commerce Technology, 2005. CEC 2005. Seventh IEEE International Conference on*, (pp. 84-91).
- Kim, Y. S., & Yum, B.-J. (2011). Recommender system based on click stream data using association rule mining. *Expert Systems with Applications*, 38, 13320-13327.
- Kim, Y. S., Yum, B.-J., Song, J., & Kim, S. M. (2005). Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. *Expert Systems with Applications*, 28, 381-393.
- Kumbaroska, V., & Mitrevski, P. (2017). Behavioural-based modelling and analysis of Navigation Patterns across Information Networks. *arXiv preprint arXiv:1701.01639*.
- Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7, 76-80.
- Moe, W. W. (2003). Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *Journal of consumer psychology*, 13, 29-39.
- Moe, W. W., & Fader, P. S. (2004). Dynamic conversion behavior at e-commerce sites. *Management Science*, 50, 326-335.
- Park, Y.-J., & Chang, K.-N. (2009). Individual and group behavior-based customer profile model for personalized product recommendation. *Expert Systems with Applications*, 36, 1932-1939.
- Paterson, M., & Vlado, D. (1994). Longest common subsequences. *International Symposium on Mathematical Foundations of Computer Science*, (pp. 127-142).
- Picot-Clémente, R., Bothorel, C., & Lenca, P. (2014). Towards Intention, Contextual and Social Based Recommender System. In *Advanced Approaches to Intelligent Information and Database Systems* (pp. 3-13). Springer.

- Rathipriya, R., & Thangavel, K. (2011). A fuzzy co-clustering approach for clickstream data pattern. *arXiv preprint arXiv:1109.6726*.
- Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1-35). Springer.
- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Analysis of recommendation algorithms for e-commerce. *Proceedings of the 2nd ACM conference on Electronic commerce*, (pp. 158-167).
- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th international conference on World Wide Web*, (pp. 285-295).
- Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. *Data mining and knowledge discovery*, 5, 115-153.
- Senecal, S., Kalczynski, P. J., & Nantel, J. (2005). Consumers' decision-making process and their online shopping behavior: a clickstream analysis. *Journal of Business Research*, 58, 1599-1608.
- Sequeira, K., & Zaki, M. (2002). ADMIT: anomaly-based data mining for intrusions. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, (pp. 386-395).
- Shardanand, U., & Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth". *Proceedings of the SIGCHI conference on Human factors in computing systems*, (pp. 210-217).
- Shi, F., & Ghedira, C. (2016). Intention-based online consumer classification for recommendation and personalization. *Hot Topics in Web Systems and Technologies (HotWeb), 2016 Fourth IEEE Workshop on*, (pp. 36-41).
- Shi, F., & Ghedira, C. (2017). Improving recommender systems with an intention-based algorithm switching strategy. *Proceedings of the Symposium on Applied Computing*, (pp. 1668-1673).
- Sismeiro, C., & Bucklin, R. E. (2004). Modeling purchase behavior at an e-commerce web site: A task-completion approach. *Journal of marketing research*, 41, 306-323.
- Sivapalan, S., Sadeghian, A., Rahnama, H., & Madni, A. M. (2014). Recommender systems in e-commerce. *World Automation Congress (WAC), 2014*, (pp. 179-184).

- Su, Q., & Chen, L. (2015). A method for discovering clusters of e-commerce interest patterns using click-stream data. *electronic commerce research and applications*, 14, 1-13.
- Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 4.
- Ungar, L. H., & Foster, D. P. (1998). Clustering methods for collaborative filtering. *AAAI workshop on recommendation systems*, 1, pp. 114-129.
- Van den Poel, D., & Buckinx, W. (2005). Predicting online-purchasing behaviour. *European journal of operational research*, 166, 557-575.
- Volk, M., Shareef, A. E., Jamous, N., & Turowski, K. (2017). New E-Commerce User Interest Patterns. *Big Data (BigData Congress), 2017 IEEE International Congress on*, (pp. 406-413).
- Wang, G., Zhang, X., Tang, S., Wilson, C., Zheng, H., & Zhao, B. Y. (2017). Clickstream user behavior models. *ACM Transactions on the Web (TWEB)*, 11, 21.
- Wei, K., Huang, J., & Fu, S. (2007). A survey of e-commerce recommender systems. *Service systems and service management, 2007 international conference on*, (pp. 1-5).
- Weisstein, E. W. (2002). Normal Vector.
- Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., & Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, (pp. 114-121).
- Yun, U., & Leggett, J. J. (2005). WFIM: weighted frequent itemset mining with a weight range and a minimum weight. *Proceedings of the 2005 SIAM International Conference on Data Mining*, (pp. 636-640).
- Yun, U., & Ryu, K. H. (2011). Approximate weighted frequent pattern mining with/without noisy environments. *Knowledge-Based Systems*, 24, 73-82.
- Yun, U., Shin, H., Ryu, K. H., & Yoon, E. (2012). An efficient mining algorithm for maximal weighted frequent patterns in transactional databases. *Knowledge-Based Systems*, 33, 53-64.
- Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE transactions on knowledge and data engineering*, 12, 372-390.

Zhao, C.-t., & Chun-e, M. (2011). BWorld Robot Control Software.

Zheng, L., Cui, S., Yue, D., & Zhao, X. (2010). User interest modeling based on browsing behavior. *Advanced Computer Theory and Engineering (ICACTE), 2010 3rd International Conference on*, 5, pp. V5--455.

VITA AUCTORIS

NAME: Ying Xiao
PLACE OF BIRTH: Shaanxi, China
YEAR OF BIRTH: 1988
EDUCATION: Lujing High School, Shaanxi, China, 2004
Xi'an University of Posts and Telecommunications, B.E.,
Shaanxi, China, 2007
University of Windsor, M.Sc., Windsor, ON, 2016