8-27-2019

E-commerce Recommendation by an Ensemble of Purchase Matrices with Sequential Patterns

Mehdi Naseri
University of Windsor

E-commerce Recommendation by an Ensemble of Purchase Matrices with Sequential Patterns

By

Mehdi Naseri

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

2019

E-commerce Recommendation by an Ensemble of Purchase Matrices with Sequential Patterns

by

Mehdi Naseri

APPROVED BY:

Z. Hu
Department of Mathematics & Statistics

S. Samet
School of Computer Science

C.I. Ezeife, Advisor
School of Computer Science

August 27, 2019

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 recommendation systems, integrating collaborative filtering (CF) and sequential pattern mining (SPM) of purchase history data will improve the accuracy of recommendations and mitigate limitations of using only explicit user ratings for recommendations. Existing E-commerce recommendation systems which have combined CF with some form of sequences from purchase history are those referred to as LiuRec09, ChioRec12 and HPCRec18. ChoiRec12 system, HOPE first derives implicit ratings from purchase frequency of users in transaction data which it uses to create user item rating matrix input to CF. Then, it computes the CFPP, the CF-based predicted preference of each target $user_u$ on an $item_i$ as its output from the CF process. Similarly, it derives sequential patterns from the historical purchase database from which it obtains the second output matrix of SPAPP, sequential pattern analysis predicted preference of each user for each item. The final predicted preference of each user for each item FPP is obtained by integrating these two matrices by giving 90% to SPAPP and 10% to CFPP so it can recommend items with highest ratings to users. A limitation of HOPE system is that in user item matrix of CF, it does not distinguish between purchase frequency and ratings used for CF. Also in SPM, it recommends items, regardless of whether user has purchased that item before or not.

This thesis proposes an E-commerce recommendation system, SEERs (Stacking Ensemble E-commerce Recommendation system), which improves on HOPE system to make better recommendations in the following two ways: i) Learning the best minimum support for SPA, best k similar users for CF and the best weights for integrating the four used matrices. ii) Separating their two intermediate matrices of CFPP and SPAPP into four intermediate matrices of $CF\_notpurchased$, $SPM\_purchased$, $SPM\_notpurchased$ and $purchasehistory$ matrix which are obtained and merged with the better-learned parameters from (i) above. Experimental results show that by using best weights discovered in training phase, and also separating purchased and not purchased items in CF and sequential pattern mining methods, SEERS provides better precision, recall, F1 score, and accuracy compared to tested systems.

Keywords: Recommendation System, Collaborative Filtering, Frequent Sequential Pattern Mining, E-Commerce, Data Mining, Historical Purchase.

DEDICATION

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

ACKNOWLEDGEMENTS

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

I also thank my internal reader, Dr. Saeed Samet and my external reader, Dr. Zhiguo Hu, for their time, effort, and valuable opinions.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Recommendation systems are tools and techniques, suggesting items to users such as what items to buy, what music to listen, what movie to watch, or what news to read (Ricci, Rokach, & Shapira, 2011). Many Recommendation systems have been developed in various domains such as Movies (Netflix), News (Google), Image (Tumblr), Video (YouTube), Friend (Facebook), Travel (TripAdvisor), Music (Spotify), E-commerce (Amazon). Selling or buying products or services online are termed as e-commerce. Each e-commerce platform usually has many products in its repository, and customers have to find their favorite products one by one. To make shopping process more convenient and efficient, most e-commerce portals use recommendation systems to offer appropriate products to a customer. So, they have become an important component of e-commerce platforms. Recommendation systems are also beneficial for sellers too because they can sell more products by recommending needed products to more customers, and the likelihood of more purchases will be increased.

1.1 E-commerce Recommendation System Input and Output

Before using any recommendation method, we should gather information from e-commerce databases such as users and products. We can categorize e-commerce recommendation systems' inputs in four categories: user profile (e.g. customer age or gender), products features (e.g. product name, category, brand), community data (e.g. products bought by similar customers, bestselling products), and knowledge

models (e.g. frequent purchase sequences already found in customers historical purchases). This information might be personalized data about a specific customer (such as their age, interests or previous transaction) or it might be non-personalized data (such as best sellers, market sales trends, shoppers advises, public statistics, market research) which are similar for all users.

There are many methods to gather these inputs. We can categorize information gathering methods in two categories: explicit information gathering and implicit information gathering. Explicit information gathering includes collecting rating of products by users, registration form, asking for interest and preferences. Implicit information gathering includes collecting the history of purchases, navigation history, time spent on specific pages, links followed by a user, button clicks, user data from social network platforms.

The output of a recommendation system is a list of top-N recommended items to each user. These items are ranked based on their ratings. For example recommendation system might recommend top 3 products {Item_2, Item_1, Item_3} to user_1 which indicates for this user, item_2 has the highest rating, then item_1 and item_3. In a similar way, three items will be recommended to each user.

1.2 Collaborative Filtering

Collaborative filtering is the most commonly researched recommendation system technique. This method receives user item rating matrix as input and predicts unknown values in this matrix and return it as collaborative filtering output. A sample user item purchase frequency matrix is displayed in Table 1. For example this table shows $user_1$ has purchased $item_1$, three times. As we can see in Table 1, many ratings are unknown in this input matrix. For example we do not know if $user_1$ likes $item_3$ or not.

Table 1: User Item rating matrix

	$\mathbf{Item_1}$	$\mathbf{Item_2}$	$\mathbf{Item_3}$	$\mathbf{Item_4}$	$\mathbf{Item_5}$
$\mathbf{User_1}$	3	1	?	1	3
$\mathbf{User_2}$	?	1	4	1	2
$\mathbf{User_3}$	2	2	4	2	1
$\mathbf{User_4}$	3	3	3	1	?

In this research, neighborhood-based collaborative filtering is used. This method receives the user item rating matrix as input. Then finds other users who have similar ratings in this matrix, by using similarity methods such as Pearson or cosine similarity. Then predict unknown ratings by calculating weighted ratings of similar users which have a rating for that item. Therefore collaborative filtering recommendation systems can fill up more ratings in user item matrix.

Problem Definition: Receive user item rating matrix as input (e.g. Table 1), the goal is to find unknown ratings in this matrix and return new user item rating matrix with less unknown ratings (e.g. Table 3) as output.

Input: Collaborative filtering recommendation system receives user item rating matrix as input. This matrix shows ratings of users for items. The ratings can be discovered from explicit data (e.g. user explicitly rated products) or by using implicit data (e.g. discover rating based on the historical purchase of user). For example, in e-commerce, each rating shows the frequency of purchasing an item by a user. For example sample input which is displayed in Table 1 indicates that $user_1$ has purchased $item_1$ three times.

Output: User Item matrix with predicted ratings (e.g. Table 1) which has less or no unknown ratings compare to input user item matrix.

Algorithm: User-based neighborhood-based collaborative filtering system discovers unknown ratings of user item rating matrix (Table 1) in the following four steps:

Step 1. Compute mean rating of each user. For each user, we find average

rating of items for that user (Equation 1). In this Equation, r_{ui} denote rating of $user_u$ for $item_i$. $\bar{r_i}$ denote mean rating of $user_i$. $|I_u|$ denote number of items which $user_u$ have a rating for them, for example $user_1$ has rating for four items therefor $|I_1| = 4$.

$$\bar{r_i} = \frac{\sum_{i \in I_u} r_{ui}}{|I_u|} \tag{1}$$

For example, in Table 1, $user_1$ has rated four items which are 3,1,1,3, so $\sum_{i \in I_u} r_{ui} = 3 + 1 + 1 + 3$ and $|I_1| = 4$ therefore mean rating of $user_1$ is 1.75.

$$\bar{r_1} = \frac{3 + 1 + 1 + 3}{4} = 2$$

Therefore, users mean ratings are: $\bar{r_1} = 2$, $\bar{r_2} = 2$, $\bar{r_3} = 2.2$, $\bar{r_4} = 2.5$.

Step 2. Compute similarity between users. In this step, similarity function is used to find similarity between the target user and all other users (Table 2). In this section, we have used Pearson similarity to estimate the similarity between two users (Equation 2). In this equation, u and v are two users who we want to find similarity between them. r_{ui} denote rating of $item_i$ by $user_u$. I_u denote item indices which $user_u$ have a rating for them.

$$PearsonSimilarity(u, v) = \frac{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in I_u \cap I_v} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{i \in I_u \cap I_v} (r_{vi} - \bar{r_u})^2}} \tag{2}$$

For example in Table 1, $user_1$ has rated $item_1$, $item_2$, $item_4$, $item_5$ so I_1 is $\{1, 2, 4, 5\}$ and $user_2$ rated $item_2$, $item_3$, $item_4$, $item_5$ so I_2 is $\{2, 3, 4, 5\}$ and $I_u \cap I_v$ is 2,4,5. Now by using similarity equation 2 we find similarity between $user_1$ and $user_2$.

$$Similarity(u_1, u_2) = \frac{(1-2)*(1-2)+(1-2)*(1-2)+(3-2)*(2-2)}{\sqrt{(1-2)^2+(1-2)^2+(3-2)^2}\sqrt{(1-2)^2+(1-2)^2+(2-2)^2}} = 0.82$$

In a similar way, we find similarity between all users in the input user item rating matrix (Table 1). Calculated similarities are displayed in Table 2.

Table 2: Similarity between users

	User$_1$	**User$_2$**	**User$_3$**	**User$_4$**
User$_1$	-	0.82	-0.4	0.52
User$_2$	-	-	0.89	0.13
User$_3$	-	-	-	0.31
User$_4$	-	-	-	-

Step 3. Compute user$_u$'s pear group for item$_j$. $P_u(j)$ denote pear group of $user_u$ for $item_j$ which is the set of users who have the highest similarity with $user_u$ and have a rating for $item_i$. For example based on Table 2, $P_{user_1}(item_3)$ are $user_4$ and $user_2$ because they have the most similarity with $user_1$ and both of them have a rating for $item_3$.

Step 4. Compute unknown ratings in user item matrix. To find unknown rating of $user_1$ for $item_1$ we use Equation 3. in this equation, $\hat{r}_{ui}$ denote unknown rating of $user_u$ for $item_i$. r_{ui} denote unknown rating of $user_u$ for $item_i$. $Similarity(u, v)$ is calculated similarity between $user_u$ and $user_v$ in step 2. $P_u(j)$ denote pear group of $user_u$ for $item_j$ which is found in step 3 and shows most similar users to $user_u$ who have rating for $item_j$.

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum\limits_{v \in P_{\mathrm{u}}(j)} Similarity(u, v).(r_{\mathrm{vi}} - \bar{r_v})}{\sum\limits_{v \in P_{\mathrm{u}}(j)} |Similarity(u, v)|} \tag{3}$$

For example two most similar users to $user_1$ are $user_4$ and $user_2$ also both of them have rating for $item_3$. So $P_{user_1}(item_3)$ are $user_2$ and $user_4$. Rating of $user_2$ for $item_3$ is 4 and rating of $user_4$ for this item is 3. By using equation 3 we have:

$$\hat{r}_{user_1, item_3} = 2 + \frac{0.82 * (4 - 2) + 0.52 * (3 - 2.5)}{0.82 + 0.52} = 2 + \frac{1.64 + 0.26}{1.34} = 3.42 \tag{4}$$

Other unknown ratings of input user item matrix (Table 1) are calculated using

Equation 3 and displayed in Table 3.

Table 3: User item matrix in with discovered unknown ratings

	Item$_1$	**Item$_2$**	**Item$_3$**	**Item$_4$**	**Item$_5$**
User$_1$	3	1	3.42	1	3
User$_2$	2.72	1	4	1	2
User$_3$	2	2	4	2	1
User$_4$	3	3	3	1	2.18

After discovering unknown ratings in the user item matrix, we can use it to recommend items to users by selecting items which have the highest ratings. For example, consider user item matrix in Table 3. If we want to recommend three items to each user, then for $users_1$, $item_5$, $item_3$ and $item_4$ will be recommended because these items have highest ratings in the user item matrix. Similarly recommended items to other users will be selected.

1.2.1 Sequential Pattern Mining

Sequential pattern mining receives input sequential database (e.g. Table 4) and minimum support (e.g. 50 percent) as input and discovers frequent subsequences in sequential database with frequency higher than a user-specified threshold (e.g. 50 percent) (Aggarwal, Bhuiyan, & Al Hasan, 2014). Each input sequence consists of a list of itemsets (e.g. purchase transactions). For example, a sequence in input sequential database (Table 4) is <(A)(BD)(C)(E)> which consist of four itemsets and indicates user, purchase item A then items B and C together, later buy item C and finally purchase item E. Also, each itemset is a list of items (e.g. purchased products), for example (BC) is an itemset which consists of two items which are B and C. The output of sequential pattern mining is a list of frequent itemsets. For example <(B)(C)> is a frequent sequence with support of 50 percent because two users out of four have purchased B and then C in their input sequence (Table 4). Sequential pattern mining can be used to predict what users will purchase in the

future. For example, it can predict a customer who purchases a digital camera, later will buy a memory card. The problem of frequent sequential pattern mining has been widely studied because of its numerous applications to a variety of data mining problems such as mining e-commerce customers purchase pattern. As a result, many algorithms have been developed for mining frequent sequential patterns such as GSP (Srikant & Agrawal, 1996), Spade (Zaki, 2001), Spam (Ayres, Flannick, Gehrke, & Yiu, 2002), Prefixspan (Pei et al., 2004), Lapin-Spam (Yang & Kitsuregawa, 2005), freespan (Han et al., 2000) and PLWAP (Ezeife, Lu, & Liu, 2005).

1.2.2 Generalized Sequential Pattern (GSP) Algorithm

Generalized Sequential Pattern (GSP) method proposed in (Srikant & Agrawal, 1996), is much faster than ArioriAll (Agrawal & Srikant, 1995). This algorithm finds sequential patterns in a sequential database as opposed to a non-sequential transactional database in (Agrawal & Srikant, 1995). This algorithm contribution is consisting of using candidate generation, pruning, and support counting algorithms for extracting frequent sequential patterns. GSP algorithm receive input sequential database (Table 4) and minimum support (e.g. 50 percent). This method first discovers 1-item candidates (C_1) then count support of C_1 candidates and those candidates which have a *support* higher than *minimum support* are considered 1-item frequent sequences (L_1). Then in a loop, generate candidate C_k by using $L_{\text{k-1}} \bowtie_{\text{GSP join}} L_{\text{k-1}}$ then in candidate pruning step remove any candidates which have a non-frequent sub-sequence and finally in counting step select candidate with *support* higher than *minimum support* as frequent sequence L_k. Continue this loop until there is not any new frequent sequence. This algorithm return $F1 \cup F2 \cup .. \cup F_n$ as output frequent sequences. In this section, all steps of this algorithm are described in detail with examples.

Input: List of sequences in sequential database and user specified minimum support. for example, input sequences can be <(A)(BD)(C)(E)>, <(B)(C)(A)>, <(A)(CD)(B)>, <(AB)(E)(BD)> and minimum support of 50 percent.

Table 4: Sample input sequential database

Id	Sequence
1	<(A)(BD)(C)(E)>
2	<(B)(C)(A)>
3	<(A)(CD)(B)>
4	<(AB)(E)(BD)>

Output: All sequences that have *support* greater or equal to minimum support (e.g. <(A)(C)(B)>).

Algorithm: GSP algorithm receives a sequential database and discovers frequent sequential patterns by using the following steps.

Step 1. Discovering all 1-item candidates (C_1). In this step, read all sequences to determine all items (C_1). For example, items in the previous example (1-itemset candidates) are C_1 = (A), (B), (C), (D), (E).

Step 2. Find 1-item frequent sequences (L_1). Counting support of candidates and return frequent candidates. We pass through input sequences to count the number of sequences that include that item. In previous example, support of items are <(A),4>, <(B),4>, <(C),3> ,<(D),3>, <(E),2>. Then select candidates which have a support value higher than user-specified minimum support as frequent 1-sequences. For example, If minimum support is 50% then each item must be in at least two input sequences. In this example, support of all items is more than two. As a result, 1-item frequent sequential patterns are L1 = <(A),4>, <(B),4>, <(C),3> ,<(D),3>, <(E),2>

Step 3. Generate 2-element candidates (C_2). First, like Apriori algorithm, create all combinations of two items (e.g. (A)(A), (A)(B), (A)(C), (A)(D), ...). In this example, there are 25 combinations. For example, <(A)(B)> indicates that user has bought item A and then later buy item B. Also, we should consider cases that user buys two items at the same period. In this example there are 10 possible combinations that customer buys two items together such as <(AB)>, <(AC)>, <(AD)>, etc.. For

example (AB) indicates that user purchase item A and B together.

Step 4. Find frequent L_2. Counting support of 2-element candidates and return frequent candidates. In this step first count number of sequences which contain candidates from the previous step. Then save the candidates that have a support higher than minimum support. In this example there are 7 candidates which have a support higher than minimum support which are: <(A)(B),3>, <(A)(C),2>, <(A)(D),3>, <(A)(E),2>, <(B)(C),2>, <(BD),2>, <(B)(E),2>.

Repeat step 5 to 7 until there is not any new frequent itemset. In each loop we use previous L_{k-1} sequences to create new candidates (C_k) by using GSP join. Then after pruning and support counting, we find the next frequent sequences (L_{k+1}).

Step 5. Candidate Generation. Use ($L_{k-1} \bowtie_{\text{GSP join}} L_{k-1}$) to generate candidates. For generating k-sequence candidates in GSP join, we receive two L_{k-1} sequences. Then remove the first and last item in the sequences. Then, we select sequences that their *sequence-1* is equal to *sequence-Last*. Then we add the last item of the second sequence to the end of the first sequence. If the last item is part of the last element of the second sequence, then we add it to the last itemset of the first sequence. If it is a separate element in sequence_2, then we add it to the first sequence as a separate element (Table 5). For example, if we merge <(A)(B)> and <(B)(C)> the candidate is <(A)(B)(C)> but if we combine <(A)(B)> and <(BD)> the candidate is <(A)(BD)>.

Table 5: 3-sequence candidate generation

Sequence 1	**Sequence1-1st**	**Sequence 2**	**Sequence2-Last**	**New Candidates**
(A)(B)	B	(B)(C)	B	(A)(B)(C)
(A)(B)	B	(B)(E)	B	(A)(B)(E)
(A)(B)	B	(BD)	B	(A)(BD)
(A)(D)	D	(BD)	D	(A)(BD)
(BD)	B	(B)(C)	B	(BD)(C)
(BD)	B	(B)(E)	B	(BD)(E)

Step 6. Candidate Pruning. Before reading all sequenced to find the support of each candidate, it is better to prune candidates. We know if a candidate is frequent, then all of its sub-sequences must be frequent too. For example, if <(A)(B)(C)> is frequent then <(A)(B)>, <(A)(C)> and <(B)(C)> are frequent too. Therefore we remove any candidate which has a non-frequent subsequence.

Step 7. Return frequent itemsets (L_k) by counting support of candidates. First, we read all sequences and counting support of each pruned candidate. For example in previous step we had three pruned candidates and their support are <(A)(B)(C),1>, <(A)(B)(E),1> , <(A)(BD),2>. We keep candidates that their support is higher than minimum support, as new frequent sequences. In the previous example, there is only one candidate with support greater than minimum support <A(BD)>.

Step 8. Repeat step 5 to step 7 until there is not any new frequent sequence. In each step, we found new frequent sequences and return $F_1 \cup F_2 \cup .. \cup F_n$ as final frequent sequences. For example, in the previous sequential dataset, some frequent itemsets and their support are <(A),100%>, <(B)(C),50%> , <(A)(BD),50%>

1.3 E-commerce Recommendation Systems Evaluation

There are some criteria for evaluating each recommendation system, such as accuracy, diversity, coverage, confidence and trust, novelty, serendipity, robustness and stability, and scalability (Aggarwal, 2016). For example, Recommended items must be diversified; otherwise, if a user doesn't like the first recommended item, he would probably reject all recommended items. We can find diversity by finding similarity between all recommended items. Higher similarity means less diversity, and lower similarity means more diversity. But accuracy is the most important metric for evaluating recommendation systems, so we describe accuracy thoroughly in this section. Later in chapter four, it is used to evaluate e-commerce recommendation systems.

1.3.1 Accuracy

Accuracy is the most important factor for evaluation of e-commerce recommendation systems. Three metrics are widely used for evaluating recommendation accuracy, which are Precision, Recall, and F1 Score. E-commerce recommendation system suggests some products to each user. Some of these products might be actually good, and user buys them. Also, the user might not like some of the recommended products. In general, we can categorize all products into four groups. True Positive (TP) represent purchased products that were also recommended correctly. False Positive (FP) represent products not purchased but which were recommended falsely. False Negative (FN) represent products purchased but which were not recommended falsely. True Negative (TN) represent products not purchased and not recommended correctly.

Table 6: Categories of product recommendation

		Purchase Reality	
		Purchased	Not Purchased
Recommendation	Recommended	True Positive (TP)	False Positive (FP)
	Not Recommended	False Negative (FN)	True Negative (TN)

Precision is calculating by dividing the number of items correctly recommended by the number of all recommendations (Equation 5). Precision shows the percentage of items that are recommended correctly to users. In other words, it is the percentage of recommended items that have been purchased by users.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

For example, if we recommend 10 items to a user and he buys 3 of them, then precision is 0.3 because $\frac{3}{10} = 0.3$.

Recall is the number of items correctly recommended (i.e. recommended and purchased), divided by all items purchased (Equation 6). Recall shows the percentage

of items that are recommended and users have purchased them to the number of all items purchased by users.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

For example, if a user buys 10 items and we recommend 4 of them to the user correctly, then recall is 0.4 because $\frac{4}{10} = 0.4$.

F1 score is the weighted average of the precision and recall, where the best score is 1, and the worst score is 0 (Equation 7). F1-score combines both *precision* and *recall* so it can be used as an overall utility of the recommendation system.

$$F1score = 2 * \frac{Precision.Recal}{Precision + Recal} \tag{7}$$

For example if precision was 0.3 and recall was 0.4, then $F1score$ is 0.34 because $2 * \frac{0.3*0.4}{0.3+0.4} = 0.34$.

1.4 Existing Hybrid E-commerce Recommendation Systems of Collaborative Filtering and Sequential Pattern Mining

One of the methods used in recommendation systems is collaborative filtering. This method finds similar users' ratings to recommend a product to a user, but collaborative filtering does not consider customers preference change over time. Sequential rule mining discovers customer's purchase patterns, so it can predict next purchase behavior based on previous purchase patterns. As a result, by combining these two methods, the accuracy of e-commerce recommendation can be improved. Authors in (D.-R. Liu, Lai, & Lee, 2009) research proposed a hybrid recommendation system which combines segment-based sequential rule mining with segment-based KNN collaborative filtering. In another research, (Choi, Yoo, Kim, & Suh, 2012) used implicit rating to create user item matrix of collaborative filtering. Also, they used sequential pattern mining to find the frequent pattern. In the end, they integrate collaborative

filtering and sequential pattern mining and recommend items with the highest ratings to the users. Collaborative filtering recommendation system uses the matrix of user-item for predicting next item. But usually, this matrix is very sparse because for many user-items we do not have any purchase information. Historical purchase with clickstream recommendation system (HPCRec) method (Xiao & Ezeife, 2018) uses consequential bond information to compare clickstream and purchase sequences of users for purposes of predicting items for recommendation. Historical Sequential Pattern Recommendation System (HSPRec) (Bhatta, Ezeife, & Butt, 2019) extract sequential patterns of customers' clickstreams and purchases and uses them to enrich user item matrix. Then, it uses this enriched matrix in collaborative ltering to discover unknown ratings and recommend items. In this section, we shortly explain these methods, and their limitations compared to our proposed method, and in the next chapter, we describe them thoroughly with examples.

1.4.1 Hybrid sequential rules and collaborative filtering and collaborative filtering (D.-R. Liu, Lai, & Lee, 2009)

In this method, first customers are segmented to 8 groups based on their recency, frequency and monetary. Then combine clusters which have the same recency, frequency and monetary patterns. To find segment-based sequential rules, first cluster transactions in separate groups based on purchased products. Then finds transaction cluster changes over period of times and select items in predicted transaction cluster as sequential rule recommendations. On the other hand, it uses KNN collaborative filtering to discover unknown ratings in user item matrix and recommend products in current period. Next, normalize results of these two methods and give a weight to each one. Finally, combine the results with $ProductRating = (1 - \alpha) * SequentialRule + \alpha * CollaborativeFiltering$ formula and recommend the items with highest ratings.

1.4.2 Hybrid Online Product rEcommendation (HOPE) System (Choi, Yoo, Kim, & Suh, 2012)

When the explicit rating is not available, collaborative filtering recommendations can not predict users' preferences properly. Authors in this paper improve the performance of the recommendation system in two ways. First, by using implicit rating because there would be more data about products and users as a result user product matrix of collaborative filtering is less sparse. When a user buys a product, it is considered that he like that product, and when a user buys that product more frequently, it implies that he likes that item more. This way, implicit ratings in e-commerce data can be extracted even when users are not rating products explicitly. This research extract implicit rating from purchase frequency of users. For example, if $user_1$ buys $item_1$ three times, then the rating value in user item matrix will be three.

HOPE first derives implicit ratings from purchase frequency of users in transaction data which it uses to create user item rating matrix input to CF. Then, it computes the CFPP, the CF-based predicted preference of each target $user_u$ on an $item_i$ as its output from the CF process. Similarly, it derives sequential patterns from the historical purchase database from which it obtains the second output matrix of SPAPP, sequential pattern analysis predicted preference of each user for each item. The final predicted preference of each user for each item FPP is obtained by integrating these two matrices by giving 90% to SPAPP and 10% to CFPP so it can recommend items with highest ratings to users. A limitation of HOPE system is that in user item matrix of CF, it does not distinguish between purchase frequency and ratings used for CF. Also in SPM, it recommends items, regardless of whether user has purchased that item before or not.

1.4.3 Historical Purchase with with Clickstream based Recommendation System (HPCRec) (Xiao & Ezeife, 2018)

Also, this method finds the relationship between click-stream and historical purchase and uses the predicted purchases to fill up more fields in the user item matrix (solving

sparsity problem). This way, in a situation that there is not enough information about a user, we can predict his preference (solving cold start problem). HPCRec algorithm first creates the user item matrix. In this matrix, each rating is the frequency of $item_i$, purchased by the $user_u$, which are normalized. Next, for each user which we do not have purchase data, find similarity of click stream data with all other users by using click-stream sequence similarity measurement (CSSM). Then use the transaction-based weighted frequent item (TWFI) to find the items that he would purchase with their average weight support. If a user has not purchased this item, add this rating to the user item matrix. Finally, use collaborative filtering method to fill up unknown ratings in the user item matrix and recommend items which have the highest ratings.

In Table 7, we have summarized the best current systems which are using collaborative filtering with some sort of sequential pattern mining to recommend items to users in e-commerce dataset.

Year	**Paper Title**	**Input Data**	**Major Contribution**	**Limitation**
LiuRec09 (D.-R. Liu, Lai, & Lee, 2009)	A hybrid of sequential rules and collaborative filtering for product recommendation	Historical purchase	This method combines segment-based sequential rule mining with segment-based KNN collaborative filtering to create a hybrid recommendation system	Only find transaction cluster changes not all of the sequential rules. In collaborative filtering, ratings based on purchase frequency and rating calculated based on similar users have the same weight.

HOPE (Choi, Yoo, Kim, & Suh, 2012)	A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis	Historical purchase	This research proposed a method for extracting implicit rating from purchase data. Also combined collaborative filtering with sequential pattern analysis method	In collaborative filtering, ratings based on purchase frequency and ratings calculated based on similar users have the same weight. In sequential pattern mining, recommends item regardless of if that item has already purchased by that user or not.
HPCRec18 (Xiao & Ezeife, 2018)	E-Commerce product recommendation using historical purchases and clickstream data	Historical purchase, Click-stream data	Improves quality and quantity of rating by finding relationship between click stream data and historical purchase and filling up more fields in user-item matrix	In sequential pattern mining only add items to user-item matrix which user has not purchased them. Needs clickstream data. In collaborative filtering, ratings based on purchase frequency and rating calculated based on similar users have the same weight.
HSPCRec18 (Bhatta, Ezeife, & Butt, 2019)	Mining sequential patterns of historical purchases for e-commerce recommendation	Historical purchase, Click-stream data	Enrich user item matrix with sequential patterns of customer clicks and purchases to capture better customer behavior	In sequential pattern mining only add items to user-item matrix which user has not purchased them. Needs clickstream data. In collaborative filtering, ratings based on purchase frequency and rating calculated based on similar users have the same weight.

SEERS (proposed thesis system)	E-commerce recommendation by an ensemble of purchase matrices with sequential patterns	Historical purchase	In collaborative filtering, separate purchase frequency from ratings calculated by collaborative filtering. Use sequential rule mining and separate purchased and not purchased items. Use training phase to discover the best minimum support and confidence, best number of similar neighbors, and best weights for each four intermediate user item matrices	Training time. Lack of consideration of other user item interactions (eg. Clickstream).

Table 7: Comparative features of related works

1.4.4 HOPE versus SEERS

In this section, the overall framework of HOPE and SEERS recommendation systems are mentioned. In section 3.4, both methods are compared in details with example. Hope method first, uses collaborative filtering to discover the rating of items for users. On the other hand, uses sequential pattern mining to find the rating of items. After normalizing the output of each method, integrate the results by gives 90 percent weight to sequential pattern mining ratings and 10 percent to collaborative ratings. Overview of hope system is displayed in figure 1. But HOPE method does not distinguish between purchase frequency and ratings calculated by collaborative filtering. Also, HOPE uses sequential pattern mining to recommend items, regardless of if the user has already purchased the item before or not.

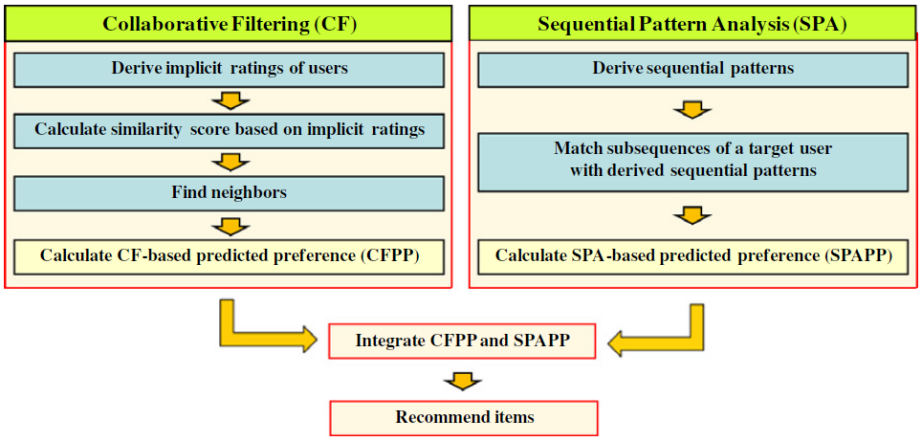

Fig. 1: HOPE Framework

SEERS method consist of two phases: training and recommendation. We also split input purchase history dataset to 3 parts: training dataset, verification dataset, and test dataset. In the training phase, SEERS receives training dataset and finds the best minimum support and minimum confidence for sequential rule mining. Then it finds the best number of similar user for collaborative filtering. Next, it uses discovered best minimum support and confidence in sequential rule mining to create $SPMpurchased$ and $SPMnotpurchased$ user item matrices. Also, use best k similar users in collaborative filtering to create $CFnotpurchased$. SEERS also create $SPMnotpurchased$ from the purchase frequency of users. In the last step of the training phase, it finds the best weight for each one of these four user item matrices. To discover the best weights, it ensembles four intermediate user item matrices by giving various weight (from 0 to 100 with steps of 5) to each method. Then we evaluate the F1 score of recommendation over verification dataset. Set of four weights which give us the best F1 score will be selected as the best weight.

In recommendation phase, It receives training and verification dataset as input and uses discovered best minimum support and confidence in sequential rule mining and discovered k similar users in collaborative filtering to create four intermediate user item matrices. After we normalized each intermediate user item matrix, we give each one, the best weights we discovered in the training phase and ensemble the results. The output is a user item matrix which is used to recommend items with the

highest ratings to each user. Overview of SEERS method is displayed in figure 2.

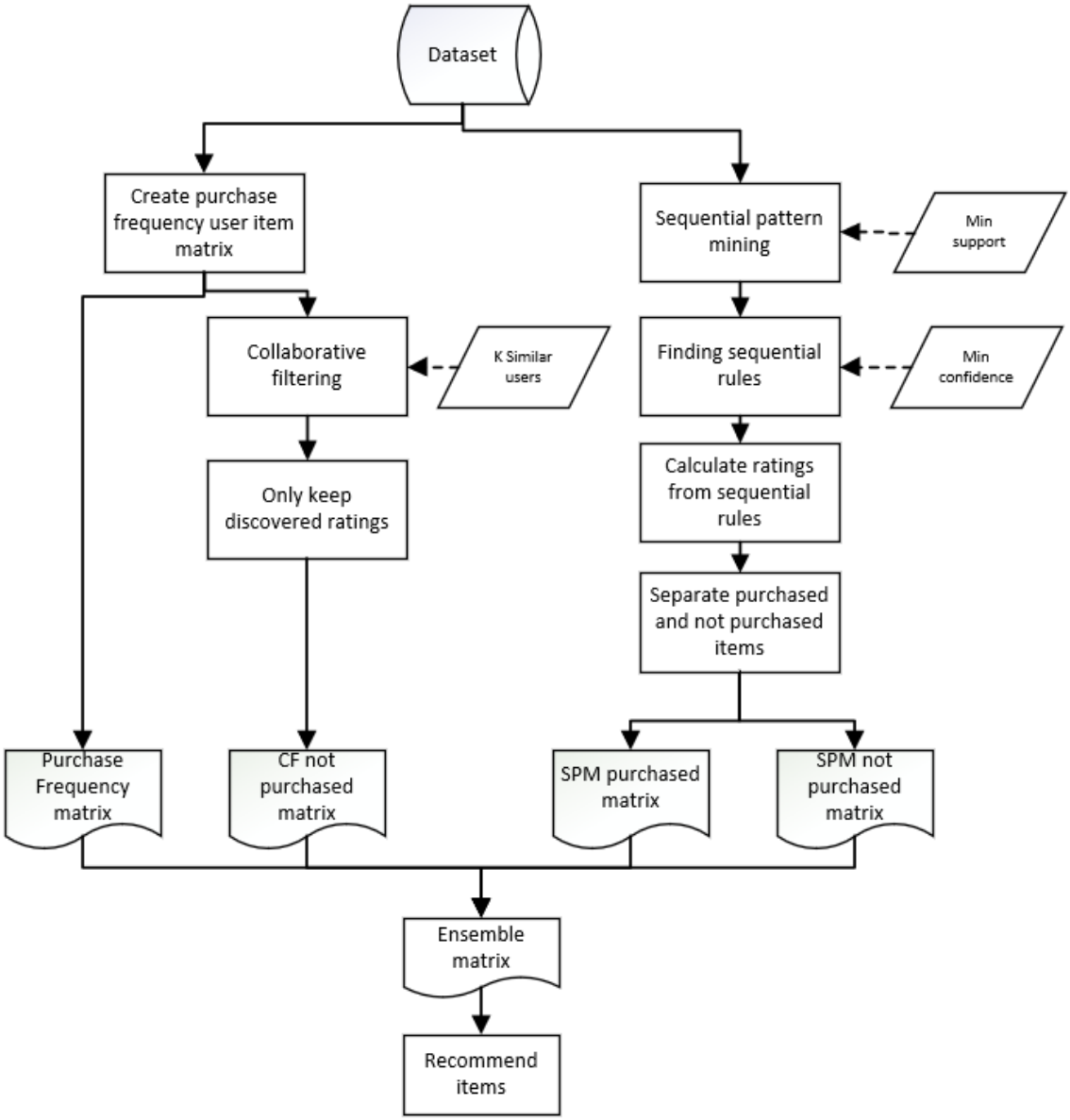

Fig. 2: Overview of SEERS recommendation phase

Figure 3 shows side by side comparison of HOPE and SEERS recommendation systems. HOPE system only uses sequential pattern mining, but SEERS also uses sequential rules in addition. Also, SEERS split calculated ratings into two parts (SPM purchased and SPM not purchased) based on if a user has purchased consequent item of the rule or not. Also when SEERS uses collaborative filtering, it only keeps discovered ratings (cf not purchased) and purchased frequencies are saved separately.

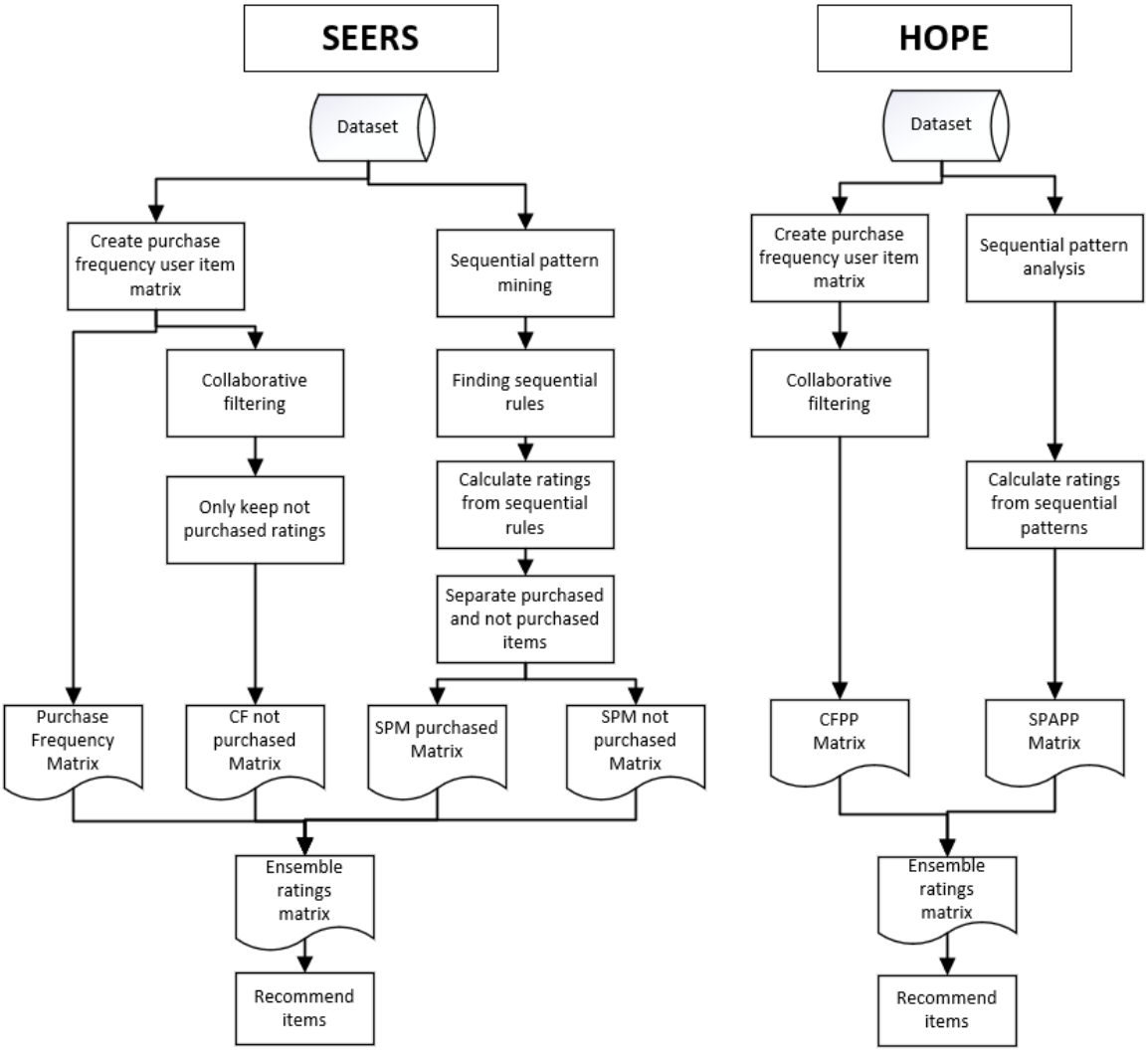

Fig. 3: Comparison of SEERS and HOPE

1.5 Observation and Thesis Hypothesis

Table 8 shows a user item matrix. As it is explained in section 1.2 we can use collaborative filtering to find unknown ratings (Table 9). For example, in this matrix, rating of $user_1$ for $item_3$ is unknown, but by using collaborative filtering have discovered 3.42 as the rating.

Table 8: User item matrix with unknown ratings

	$\textbf{Item}_1$	$\textbf{Item}_2$	$\textbf{Item}_3$	$\textbf{Item}_4$	$\textbf{Item}_5$
$\textbf{User}_1$	3	1	?	1	3
$\textbf{User}_2$	?	1	4	1	2
$\textbf{User}_3$	2	2	4	2	1
$\textbf{User}_4$	3	3	3	1	?

Table 9: User item matrix with discovered ratings

	$\textbf{Item}_1$	$\textbf{Item}_2$	$\textbf{Item}_3$	$\textbf{Item}_4$	$\textbf{Item}_5$
$\textbf{User}_1$	3	1	(3.42)	1	3
$\textbf{User}_2$	(2.72)	1	4	1	2
$\textbf{User}_3$	2	2	4	2	1
$\textbf{User}_4$	3	3	3	1	(2.18)

We can use this user item matrix to recommend items to the users by selecting items which have the highest ratings. For example if we want to recommend one item to each user, then $item_3$ will be recommended to $user_1$. As you can see in Table 9, for $user_1$, $item_3$ has the highest rating therefore it will be selected as the best recommendation. This user has already purchased all other items, but collaborative filtering is recommending the only item which the user has never purchased before. In e-commerce, users mostly prefer to buy products they have already paid money for them and purchased before compared to products they have never purchased. Current researches including (D.-R. Liu, Lai, & Lee, 2009), (Choi, Yoo, Kim, & Suh, 2012) and (Xiao & Ezeife, 2018) are ignoring this problem and treat all the ratings in user item matrix similarly, regardless of if they are calculated by collaborative filtering or they are actual purchase frequency of the user.

Thesis hypothesis: Improve rating quality. In this research, we give less weight to ratings in user item matrix which user has never purchased before and give higher weight to the rating of items which the user has purchased before. This way, we recommend more items which are purchased before compare to not purchased items. For example, if we give weight of 0.8 to purchased items and 0.2 to not purchased items and only use collaborative filtering method, then we have the user item matrix in Table 10. Based on this matrix if we want to recommend two items to $user_1$, then $item_1$ and $item_5$ will be selected. In this example, $user_1$ has already purchased both of them before.

Table 10: Add weights to user item matrix with discovered unknown ratings in collaborative filtering

	Item$_1$	**Item$_2$**	**Item$_3$**	**Item$_4$**	**Item$_5$**
User$_1$	2.4	0.8	(0.68)	0.8	2.4
User$_2$	(0.54)	0.8	3.24	0.8	1.62
User$_3$	1.6	1.6	3.2	1.6	0.8
User$_4$	2.4	2.4	2.4	0.8	(0.44)

On the other hand, we use sequential rule mining to create user item matrix and recommend items. Sequential rules are in the format of A→B, which indicates if a user has purchased item A then later will buy item B. In some rules, the user has purchased the consequent item, and in other rules, the user has not purchased the consequent item. Similar to the problem of user item matrix of collaborative filtering, which explained before, these items have different values for the user, and we should not treat them equally. In this research, we separate these items and give them proper weights which we find it in the training phase.

1.6 Thesis Contributions

In this research, we introduce a novel Stacking Ensemble E-commerce Recommendation System (SEERS), which is receiving historical purchases as input and integrate collaborative filtering and sequential rule mining to predict next purchase of users.

1.6.1 Method Contribution

Current studies in (D.-R. Liu, Lai, & Lee, 2009), (Choi, Yoo, Kim, & Suh, 2012) and (Xiao & Ezeife, 2018) are integrating collaborative filtering with some form of sequential pattern mining to improve the accuracy of e-commerce recommendation system. In this research, we improve HOPE system ((Choi, Yoo, Kim, & Suh, 2012)) by adding the following contributions:

1. **Separate the two integrated matrices of CFPP and SPAPP in HOPE to four matrices.** Current researches (including HOPE) create the user item matrix based on purchase frequency of users. Then they use collaborative filtering to find unknown ratings in this user item matrix. In these methods, ratings based on purchase frequency and rating calculated by collaborative filtering have the same weight. But when we create user item matrix from purchase history, rating extracted from purchase frequency of users are much more important. By separating them and giving more weight to items that the user has purchased before, we recommend more purchased items to the user.

   Dividing sequential rules based on if the user has already bought the consequent item or not. A sequential rule ($A \rightarrow B$) indicates that if user purchase item A, later would buy item B. But in discovered sequential rules, sometimes a user has already bought the consequent item. By separating these items from other consequent items that the user has not purchased yet, and discovering the best weight for each group, we improve the accuracy of the recommendation system.

2. **Finding best parameters.** We use stacking ensemble learning method (Wolpert, 1992) for finding the best weights for each intermediate user item matrix. HOPE gives static weights to each recommendation method (90% to sequential pattern mining and 10% to collaborative filtering) (Choi, Yoo, Kim, & Suh, 2012), but these weights might be different on various datasets. We discover the best four weights for intermediate user item matrices in each dataset. Also we find the best support and confidence for sequential rules, best number of similar users in collaborative filtering before sing these methods.

1.6.2 Feature Contribution

The proposed method in this research combines the collaborative filtering method with sequential rule mining and purchase history then uses ensemble learning to improve the following features of the recommendation system compare to HOPE system (Choi, Yoo, Kim, & Suh, 2012).

1. **Improved recommendation accuracy.** In the Training phase, SEERS, first, trains sequential rule mining to find best minimum support and confidence which leads to the highest accuracy (best F1 score). Then it trains collaborative filtering to find the best number of similar users, which generate the highest F1 score. Also, it trains four intermediate user item matrices to discover four weights to reach the highest accuracy as a hybrid method. Therefore it has better accuracy compared to each method individually.

   In the recommendation phase, when SEERS uses collaborative filtering, it separates purchase frequency from ratings calculated by collaborative filtering because these two ratings have different importance for users. Also in sequential rule mining, it distinguishes between items which user has purchased consequent items with items which user has never purchased consequent items before. By giving the best weights (which are discovered in the training phase) to each intermediate matrix, SEERS recommends items with higher accuracy.

1.6.3 Thesis Outline

In the first chapter, we introduce recommendation system methods, collaborative filtering, and sequential pattern mining. In the second chapter, we explain current researches that are combining collaborative filtering and some sort of sequential pattern mining to create e-commerce recommendation systems. In the third chapter, we propose our novel model for e-commerce recommendation system. In chapter 4, we analyze and compare our system with current recommendation systems. Finally, in chapter five, we proposed some ideas for future work.

CHAPTER 2

Related Work

Because of increasing e-commerce businesses, the importance of e-commerce recommendation systems is rapidly growing too. E-commerce recommendation systems allow users to find what they want very fast and also enable sellers to recommend products to users that are more likely to buy. In this research, sequential pattern mining and collaborative filtering methods are used to recommend best products in e-commerce domain. In hybrid e-commerce recommendation system, we combine two previous methods to improve e-commerce recommendations accuracy. This way, we can use the benefits of these methods and also solve or mitigate the disadvantages of each method as much as possible. Therefore, hybrid recommendation systems accuracy is higher than each one of the previous methods separately.

Collaborative filtering creates a user item matrix from ratings of users, but sometimes explicit rating is not available. When explicit ratings are not available, collaborative filtering recommendations can not predict users preferences directly, and it should extract implicit ratings to create user item matrix. Quality of recommendation system can be improved by using implicit rating in collaborative filtering. Also in collaborative filtering recommendation systems, it is difficult to recommend items to a user who has not rated any items before (new user problem), and it is difficult to recommend items which have never been rated before by any user (new item problem) and it makes poor recommendations when rating information is insufficient (sparsity problem) (Kim & Yum, 2011). On the other hand, sequential pattern mining has been used to create recommendation systems such as (Huang & Huang, 2009) but a limitation of this method is that it is difficult to recommend items that do not appear

in sequential patterns. Hybrid recommendation systems have been developed to overcome, or at least to mitigate, the limitations of content-based filtering, collaborative filtering, and rule-based recommendation systems. Changes in customers behavior are studied in (Cho, Cho, & Kim, 2005) and they use sequential rule recommendation to improve performance of collaborative filtering method, but collaborative filtering methods including the mentioned research do not consider changes in customers preference or frequent sequences in customers purchase patterns. Authors in (D.-R. Liu, Lai, & Lee, 2009) proposed a hybrid recommendation system which combines segment-based sequential rule mining with segment-based KNN collaborative filtering. First, customers are clustered in segments based on their recency, frequency, and monetary. Then, transactions are clustered, and by using sequential rule mining method, transaction cluster changes over periods are found. Also, collaborative filtering method is used to find similar customers to recommend products in the current period. Finally, linear algebra is used to combine the results of these two methods and return items with the highest rating. Another hybrid recommendation system of collaborative filtering and sequential pattern mining was proposed by (Choi, Yoo, Kim, & Suh, 2012). Authors in this paper improve performance of recommendation systems in two ways. First, quality of recommendation system is improved by adding implicit rating to explicit rating because there would be more data about products and users as a result user-product matrix of collaborative filtering is less sparse. On the other hand, authors combined collaborative filtering with sequential pattern analysis. To create a hybrid recommendation system, authors first run collaborative filtering method and sequential pattern analysis method separately. Then they normalize each one of them because they are in different ranges. Finally, integrate them by giving 90 percent weight to sequential pattern mining and 10 percent to collaborative filtering results and recommend items which have the highest rating for the user.

User's preferences and interests can be used to improve the performance of recommendation systems too. The pages customers visited, frequency of page visits or time spent on each page can be used to extract user preferences. Users browsing time has been used to calculate user interest in (Zheng, Cui, Yue, & Zhao, 2010). A more

comprehensive method for extracting users preference is proposed in (Kim & Yum, 2011), which is using purchase data and time spent on each page. Association rules mining for discovering frequent products has been used for analyzing click stream, baskets, and purchase data (Kim & Yum, 2011). In association rules mining method, which is used in (Kim & Yum, 2011), it only finds popular products. Also, it can not recommend proper products to users when there is not enough information about them (e.g. infrequent users). On the other hand, (Chen & Su, 2013) receives clickstream data and find other users who were visiting the same category of products. The proposed method in (Chen & Su, 2013) only works on the category of each visit, and this method is not efficient for mining whole dataset. Furthermore, these two methods did not use session-based clickstream and purchases. In (Su & Chen, 2015), authors proposed a method for extraction of user's preference by using customers visited pages sequence, frequency of visiting each category, and time spend on each category. The proposed algorithm put users in different groups based on their browsing patterns. Then they extract the user page visiting activities to find patterns in that group. Also, they used their method on clickstream data to extract users' preferences. On the other hand, the implicit rating can be used in collaborative filtering recommendation system to create the user item matrix. For example, when a user buys an item value of one will be saved in the related user item field. But usually, this matrix is very sparse because, for many user items, we do not have any purchase information. Also, this might be a binary matrix because if a user has bought that item, the field will be one and otherwise it will be zero. Historical purchase with clickstream recommendation system (HPCRec) method proposed by (Xiao & Ezeife, 2018) used historical purchase data to fill user item rating after normalizing the values. Also, this method finds the relationship between clickstream and historical purchase and uses the predicted purchases to fill up more fields in the user item matrix (solving sparsity problem). In this section current three researches are explained which are (D.-R. Liu, Lai, & Lee, 2009) , (Choi, Yoo, Kim, & Suh, 2012) and (Xiao & Ezeife, 2018).

2.1 E-commerce Recommendation Systems based on Collaborative Filtering and Sequential Pattern Mining

A hybrid of collaborative filtering and sequential patterns for product recommendation is proposed in (D.-R. Liu, Lai, & Lee, 2009), but this system can only extract purchase cluster changes in sequential pattern mining. In another research, a hybrid online-product recommendation system by combining implicit rating-based collaborative filtering and sequential pattern analysis proposed in (Choi, Yoo, Kim, & Suh, 2012). In (Choi, Yoo, Kim, & Suh, 2012), authors extract implicit ratings from purchase history, which can be used in collaborative filtering even when the explicit rating is not available. When a user buys a product, it is considered that he like that item and when a user buys that item more frequently, it implies that he likes that item more. This way, implicit ratings in e-commerce data can be extracted even when users are not rating products explicitly. Also, the authors claim they have improved recommendation quality by integrating collaborative filtering and sequential pattern analysis. To create a hybrid recommendation system, authors first run collaborative filtering method and sequential pattern analysis method separately. Then they normalize each one of them to become in the same range. Finally, they give weight to each one of these two recommendation methods (10% for collaborative filtering and 90% for sequential pattern analysis) to integrate the result and recommend items. In this section me explain these methods with example.

2.1.1 Hybrid Sequential Rules and Collaborative Filtering for Product Recommendation (D.-R. Liu, Lai, & Lee, 2009)

Authors in this paper proposed a hybrid recommendation system which combines segment-based sequential rule mining with segment-based KNN collaborative filtering.

Input: Historical purchase data in e-commerce dataset including purchase items, frequency of purchase, price, and time of transaction.

Output: Recommended products to each user.

Algorithm: The proposed method in this paper uses a combination of collaborative filtering and sequential rule mining. This method is explained in the following five steps:

Step 1: Customer clustering. Customers who had similar historical purchase patterns, usually have similar RFM (Recency, Frequency, Monetary) values. In this step, customers are segmented to different groups based on their RFM, so later collaborative filtering method only works on similar customers. To accomplish this task, first, RFM value for each customer is calculated. Then all the values are normalized. K-mean clustering method is used to segment all customers based on their normalized RFM. User's RFM is consists of values which are recency, frequency, and monetary. Then for each one of these three values, if it is higher than average, then "up" is assigned to its pattern and if it is less than average then "down" is assigned to its pattern. There are two possible results for each value; therefore, there are eight possible groups based on three parameters. Each customer is assigned to one of these groups. An example of clustering customers based on RFM is demonstrated in Table 11.

Table 11: Cluster of customers based on FRM

	Customers	**Recency**	**Frequency**	**Monetary**	**Recency Pattern**	**Frequency Pattern**	**Monetary Pattern**
Cluster 1	104	72	19	40797	Up	Up	Up
Cluster 2	43	119	3	7342	Up	Down	Down
Cluster 3	17	64	67	147315	Down	Up	Up
Cluster 4	214	56	19	40279	Down	Up	Up
Cluster 5	78	57	37	74045	Down	Up	Up
Cluster 6	367	58	9	18677	Down	Down	Down
Cluster 7	126	92	7	14853	Up	Down	Down
Cluster 8	24	73	8	16106	Up	Down	Down
Average		68	14	28638			

Some of these clusters can be combined. For example, clusters 3, 4, and 5 have

the same RFM patterns, because in all of them, recency is lower than average, and frequency and monetary are higher than average. So, they can be combined to form a new segmentation. In Table 12, customer segmentation in four groups is shown.

Table 12: Final segmentation of customers based of RFM patterns

Segments	**Customers**	**Recency Pattern**	**Frequency Pattern**	**Monetary Pattern**	**Recency**	**Frequency**	**Monetary**
Loyal	309	Down	Up	Up	57	26	54691
Potential	104	Up	Up	Up	72	19	40797
Uncertain	367	Down	Down	Down	58	9	18677
Valueless	409	Up	Down	Down	84	7	14801

Step 2: Segmentation-based sequential rule mining. Cluster transactions to different groups. This clustering is based on similar product purchased. First, for each customer make a bit vector. For example, if $user_1$ buys $product_1$ and $product_3$ but did not buy $product_2$ and $product_4$ then it's bit vector is (1,0,1,0). A sample bit vector for all customers is shown in Table 13. In this Table, products are displayed as Pro1 to Pro8. Then customers' transactions have clustered in groups by using the clustering method.

Table 13: Customers bit vector and clustering of customers

Customer	**Date**	**Pro1**	**Pro2**	**Pro3**	**Pro4**	**Pro5**	**Pro6**	**Pro7**	**Pro8**	**Cluster**
C1	20040416	1	0	0	0	0	0	0	0	C
C2	20031127	0	0	1	0	0	0	0	0	A
C2	20031127	0	1	0	0	0	0	1	1	B
C2	20040202	0	0	0	1	1	1	0	0	E
C3	20030820	0	1	0	0	0	1	0	1	B

The same method is used in every period to find customer's cluster in that period. For example, a customer might be in cluster A in the first period but in another cluster (e.g. cluster B) in the next period. Also, a customer might be in multiple transaction cluster in a period because he might have different transactions in that period. A

sample change of customers transactions in three periods are displayed in Table 14.

Table 14: Changes of customers transactions in multiple periods

	Period 1	**Period 2**	**Period 3**
Customer 1			C
Customer 2		AB	E
Customer 3	B		D
Customer 4		A	E
Customer 5	F		

Now sequential rule mining method is used to predict the next pattern in the next period. For example, in Table 14 we can discover $(A_{p2} \rightarrow E_{p3})$ rule with support of 40 percent and confidence of 100 percent. It means when a customer transaction pattern in period 2 is A, then his transaction pattern might be E in period 3 with support of 0.4 and confidence of 1. In the same way, if a customer transaction pattern in period 2 is B, then his transaction pattern might be E in period 3 with support of 0.2 and confidence of 1. Finally, similarity method is used to find sequential rules that are more similar to users' transactions. For example, if transaction behavior of a user is in cluster A in the first period, and it is in cluster B in the second period, then we find similarity of $(A \Rightarrow B)$ with all sequential rules that belong to that user segment. Then two rules with the highest similarity will be returned to recommend products. For example, when these two rules are $(A \Rightarrow E)$ and $(B \Rightarrow D)$ then we predict user's next purchase will be in transaction cluster of D or E. Finally, we count the frequency of each item in predicted transaction cluster (e.g. transaction cluster E) and return top N items with the highest frequency count.

Step 3: KNN collaborative filtering. In this step for each customer, Pearson similarity method is used to find the distance to other customers. K customers who have the highest similarity in the same segment are selected. Then collaborative filtering method is used to find products that the customer would buy. To do this task, first, count number of times that k-nearest neighbors have bought each product

and then recommend products that have the highest purchase frequency.

Step 4: Linear combination of sequential rule mining and KNN collaborative filtering. For each product, we have a sequential rule rating and a collaborative filtering rating. Now we combine them linearly. Give weight of α to collaborative filtering method and (1- α) to sequential rule method. This value defines the importance of each method. Then final value can be calculated by giving weight to each method in Equation 1.

$$ProductRating = (1 - \alpha) * SequentialRule + \alpha * CollaborativeFiltering \quad (1)$$

Step 5: Recommend top N products. In the previous step, ratings for each product are already calculated. Now we select top N products which have the highest ratings and recommend them to the customer.

LiuRec09 method limitations: This method does not recommend previously purchased items by that user although there is a high chance, he bought that item again. Also, this method only finds transaction cluster changes, which is usually different from items discovered using sequential rule mining. When this method uses collaborative filtering, ratings based on purchase frequency, and ratings calculated based on similar users have the same weight.

2.1.2 Hybrid Online Product rEcommendation (HOPE) System (Choi, Yoo, Kim, & Suh, 2012)

This research extract implicit ratings based on purchase history, which can be used in collaborative filtering even when the explicit rating is not available. Also, this proposed hybrid recommendation system uses a combination of collaborative filtering and sequential pattern mining in a way that its accuracy outperforms each one of collaborative filtering and sequential pattern mining recommendation methods individually in all measurements of precision, recall, and F1. The overall framework of hope system is displayed in Figure 4.

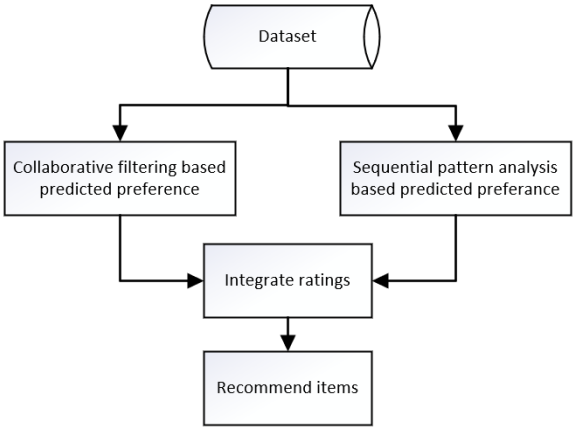

Fig. 4: HOPE Framework

Input: Historical purchase data in e-commerce dataset.

Output: Recommended products to each user.

Algorithm: This algorithm is explained in the following five steps..

Step 1: Calculate collaborative filtering predictions. First, extract ratings of users item matrix from the purchased frequency in historical purchase data. Sometimes there are not enough explicit data, or it is difficult to receive explicit ratings from users. This method uses historical purchase data of customers to extract implicit ratings. Then we use collaborative filtering in the next five steps to discover rating for each user item.

Step 1.1: Calculate absolute preference. To find the implicit rating for user item, count number of transactions that the user has purchased that item. Then divide that to the number of user's transactions and add one to the final result. So, for $user_u$ and $product_i$ following formula can be used for calculating absolute preference.

$$AP(u,i) = \frac{\text{Number of tranactions } user_u \text{ has purchased } item_i}{\text{Total number of } user_u \text{ transactions}} + 1 \qquad (2)$$

For example, if a user purchases a product 4 times in his 10 transactions, then absolute preference is 1.4 because by using Equation 2 we have: $AP(u,i) = \frac{4}{10} + 1 = 1.4$

Step 1.2: Calculate relative preference. In the previous example, the user purchases $item_i$ 4 times in his 10 transactions. If another user buys that item in eight

transactions out of total ten transactions, then we might think that this user does not like that product enough. Since the frequency of each item is different for users, it is better to normalize the values by using relative preference. For $user_u$ and $item_i$, relative preference is calculated in Equation 3. In this equation, U denote users who purchase item$_i$. AP(u,i) denote absolute preference of $user_u$ for $item_i$ which is calculated in previous step.

$$RP(u,i) = \frac{AP(u,i)}{\max\limits_{c \epsilon U} AP(c,i)} \tag{3}$$

For example, in the previous step, $user_1$ has purchased $item_1$ 4 time out of 10 transactions. If no other user buys that product in more than 40 percent of his transactions, then the relative preference of $user_1$ and $item_1$ is equal to one because $RP(user_1, item_1) = \frac{1.4}{1.4} = 1$. Similarly, if a user buys that item in 10 percent of his transactions, then its relative preference is 0.785 because $RP(user_2, item_1) = \frac{1.1}{1.4} = 0.785$.

Step 1.3: Multiply rating by five. In explicit ratings, users usually give a rate between 1 to 5 to products. To produce similar values, the relative value from the previous step is multiplied by 5, and then it is round up. For example, in previous step $RP(user_1, item_1) = 1$ and $RP(user_1, item_1) = 0.0.785$ so if we multiply them by 5 then we have $ImplicitRating(user_1, item_1) = Roundup(5 * RP(user_1, item_1)) = Roundup(5{*}1) = 5$ and $ImplicitRating(user_2, item_1) = Roundup(5{*}RP(user_2, item_1)) = Roundup(5 * 0.785) = 4$.

Step 1.4:. Find k nearest neighbors of each user by using Pearson similarity method (Equation 4). In this equation, u and v are two users that we want to find their similarity. $\bar{R}_u$ and $\bar{R}_v$ are average ratings of user u and v.

$$PearsonSimilarity(u,v) = \frac{\sum\limits_{i=1}^{n}(R_{ui} - \bar{R}_u)(R_{vi} - \bar{R}_v)}{\sqrt{\sum\limits_{i=1}^{n}(R_{ui} - \bar{R}_u)^2}\sqrt{\sum\limits_{i=1}^{n}(R_{vi} - \bar{R}_v)^2}} \tag{4}$$

Step 1.5: Discover unknown ratings in user item matrix from ratings of k-nearest

neighbors. Use Equation 5 to calculate unknown ratings for each user item. New rating for each user item, is only based on K most similar users that found in this step.

$$CF(a,i) = \bar{R}_a + \frac{\sum_{b=1}^{n} Sim(a,b) * (R_{bi} - \bar{R}_b)}{\sum_{b=1}^{n} |Sim(a,b)|} \tag{5}$$

Step 2: Calculate sequential pattern analysis-based predictions. Performs the following three sub-steps to find the rating of each user item by using sequential pattern mining.

Step 2.1. Find sequential pattern from all users' transaction sequences except target user.

Step 2.2. Find all subsequences of the user transactions. For example, when user purchase sequence is <item1, item2, item3> then its subsequences are <item1>, <item2>, <item3>, <item1, item2>, <item1, item3>, <item2, item3>, <item1, item2, item3>.

Step 2.3. Compare subsequences from step 2.2 with discovered frequent sequential patterns of all users in step 2.1. If it is matched with starting part of a frequent sequence, then next item in the frequent sequence is a candidate for recommendation. Finally, calculate support value of the recommended product which is total support of the item in all subsequences of user's transactions (Equation 6) and return those who have a support value, higher than predefined minimum support.

$$SPAPrediction(a,i) = \sum_{s \epsilon SUB} Support_s^i \tag{6}$$

Step 3: Normalize results of collaborative filtering and sequential pattern analysis. Range of ratings in sequential pattern mining and collaborative filtering are different. For example, extracted implicit rating in collaborative filtering is between one to five, so before combining two methods, it should be normalized to the same range.

Step 4: Integrating collaborative filtering and sequential pattern mining predictions. For each user item, two ratings are calculated in sequential pattern mining

and collaborative filtering method. To combine these two values, give a weight (α) to sequential pattern analysis value and $(1 - \alpha)$ weight to collaborative filtering rating. (α) has a range between zero and one. Integration of two methods' prediction is displayed in Table 15.

Table 15: Integrating collaborative filtering and sequential pattern mining

Item	**Collaborative Filtering**	**Sequential Pattern**	**Normalized Collaborative Filtering**	**Normalized Sequential Pattern**	**Fianl Rating**
Item_1	4.7	0	0	0	0.5
Item_2	3.5	0	0.5463	0	0.273
Item_3	3.2	1.25	0.4504	0.833	0.641
Item_4	2	1.5	1	1	0.5
Item_5	3	0.5	0.3333	0.33	0.348

Step 5: Recommend items. For each user, the HOPE system, recommends those items which have the highest rating in the previous step. For example, in Table 15, if we want to recommend two items, then $item_3$ and $item_4$ will be recommended because they have the highest values.

ChoiRec12 method limitations: In collaborative filtering, ratings based on purchase frequency and rating calculated based on similar users have the same weight. When this method uses sequential pattern mining, it recommends item regardless of if that item has already purchased by that user or not.

2.1.3 Historical Purchase with Clickstream based Recommendation System (HPCRec) (Xiao & Ezeife, 2018)

HPCRec uses historical purchase data to fill user item rating after normalizing the values. Also, this method finds the relationship between clickstream and historical purchase and uses the predicted purchases to fill up more fields in the user item matrix

(solving sparsity problem).

Input: Consequential matrix (Table 16a) which shows items that user clicked on them plus product that he had bought. Also, user item frequency matrix (Table 16b) which shows the frequency of buying each item by each user. This matrix is created based on purchase frequencies in table 16a.

Table 16: (a) Consequential matrix (b) user-item purchase frequency matrix

SessionId	UserId	Clicks	Purchase
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	

	Item 1	Item 2	Item 3	Item 4
User 1	?	2	1	?
User 2	1	2	?	3
User 3	1	?	?	?

Output: User item rating matrix with predicted ratings (Table 17).

Table 17: User-item rating with predicted rarings

	Item 1	Item 2	Item 3	Item 4
User 1	0.63	0.89	0.45	0.49
User 2	0.27	0.53	0.35	0.8
User 3	1	0.74	0.27	0.33

Algorithm:

Step 1. Normalizing user item purchase frequency matrix. In Table 16b we have a user item matrix which shows frequency of buying each item by a user. In this step, we use unit vector formula to normalize all ratings in Tables 16b to values between 0 and 1. Equation 7 shows the normalization formula. In this equation, x_{ui} shows frequency of buying $item_i$ by the $user_u$ and x'_{ui} is the normalized rating. For each user we create purchase vector (x_{u1}, x_{u2}, x_{u3},. . . ,x_{un}) which x_{ui} is frequency of buying $item_i$ by the $user_u$. Normalized user-item matrix is displayed in Table 18.

$$x'_{ui} = \frac{x_{ui}}{\sqrt{x_{u1}^2 + x_{u2}^2 + x_{u3}^2 + ... + x_{un}^2}} \tag{7}$$

For example, in Table 16, $user_1$ purchased items are $(2, 3)$ and purchase frequencies are $(2, 1)$. Therefore, normalized rating for $item_2$ is calculating by Equation 7:

$$x'_{u1i2} = \frac{2}{\sqrt{2^2 + 1^2}} = \frac{2}{\sqrt{5}} = \frac{2}{2.2360} = 0.89 \tag{8}$$

Table 18: Normalized user-item purchased frequency

	Item 1	**Item 2**	**Item 3**	**Item 4**
User 1	?	0.89	0.45	?
User 2	0.27	0.53	?	0.8
User 3	1	?	?	?

Step 2: For each session in the consequential matrix (Table 16a) which does not have a purchase value, we use clickstream sequence similarity measurement (CSSM) to find similar sessions with purchase value. For example, in Table 16a, session six does not have purchase value, so we compare its clickstream with sessions 1,2,3,4,5. CSSM method receives two clickstreams to find similarity between them in three steps.

Step 2.1: For two sequence, we find the longest common subsequence rate (LCSR). This value is calculated by diving the longest common subsequence(LCS) divided by the maximum length of two sequences (Equation 9). For example, longest common subsequence between (3,5,2) and (3,5,2,3) is 3. Length of the first sequence is 3, and the second one is 4, so the maximum length is 4. Now we use the equation 9 to find longest common subsequence rate (LCSR) of two sequences.

$$LCSR(x, y) = \frac{LCS(x, y)}{max(x, y)} \tag{9}$$

$$LCSR(< 3, 5, 2 >, < 3, 5, 2, 3 >) = \frac{LCS(< 3, 5, 2 >, < 3, 5, 2, 3 >)}{max(3, 4)} = \frac{3}{4} = 0.75$$

Step 2.2: First, create an itemset which contains all distinct items in both sequences (in the previous example <2,3,5>). Then create a vector from the frequency of items in itemset. In previous example, vector for first sequence<3,5,2> is <1,1,1> and for second sequence <3,5,2,3> is <1,2,1>. Now we find cosine similarity between these two vectors to find frequency similarity. In this example, frequency similarity (FS) is 0.94.

Step 2.3: Use $Similarity = \alpha * LCSR + \beta * FS$ equation to find final similarity between to sequences. Values of α and β are used for giving weight to each one of similarity and longest common subsequence. These two weights must have a value between 0 and 1. Also, their sum must be 1. For example, if we set each one of them 0.5 then for previous example, we have $0.5 * 0.75 + 0.5 + 0.94 = 0.845$. Similarly if we compare (3,5,2) with first sequence then we have CSSM((3, 5, 2),(1, 2)) = 0.37. Now we can create weighted transaction record, which is <(2):0.37>. Other clickstreams comparisons are demonstrated in Table 19.

Table 19: Click stream similarity with session <3,5,2>

Clickstream	**CSSM**	**Purchase**
1,2	0.37	2
3,5,2,3	0.845	2,3
2,1,4	0.33	1,2,4
4,4,1,2	0.245	2,4,4
1,2,1	0.295	1

Step 3: Generating purchase predictions. After receiving weighted transaction values (Table 19) we use transaction-based weighted frequent item (TWFI) method to produce weighted frequent items (E.g. <2:1><3:0.189><4:0.167>). This method has three steps:

Step 3.1: Creating a list of distinct items from weighted transactions (Table 19) and calculate their support. (e.g. (1:2),(2:4),(3:1),(4:3))

Step 3.2: For each item support from the previous step, use Equation 10 to

calculate average weight support (AWS).

$$AWS = sum(weight) \tag{10}$$

For example, $item_4$ has been purchased one time in the third sequence and two times in the fourth sequence, so the sum of the weights is 0.82:

$$AWS(4) = 0.33 + 0.245 + 0.245 = 0.82$$

Step 3.3: Using feature scaling formula to normalize weighted support (Equation 11).

$$x' = \frac{x - min}{max - min} \tag{11}$$

for example, normalized rating for $item_3$ is 0.189:

$$x' = \frac{x - min}{max - min} = \frac{0.845 - 0.625}{1.79 - 0.625} = 0.189$$

Step 3.4: Returning all of these items that have a normalized weighted support greater than minimum weighted support (e.g. (2:1),(3:0.189),(4:0.167)). In this example, minimum weighted support is 0.15. Then for each one of these items, if user has not purchased it, add the weight into the normalized user-item matrix (Table 19) to generate new user-item matrix (Table 20) which is less sparse now.

Table 20: Normalized user-item matrix with predicted weights

	Item 1	**Item 2**	**Item 3**	**Item 4**
User 1	?	0.89	0.45	?
User 2	0.27	0.53	?	0.8
User 3	1	1	0.189	0.167

Repeat step 2 and 3 until there is not any session without purchase.

Step 4: In this step, we receive the normalized user item matrix with predicted ratings (Table 20) and using collaborative filtering to predict all remaining ratings.

(Table 17)

Table 21: Final user matrix with all of predictions

	Item 1	**Item 2**	**Item 3**	**Item 4**
User 1	0.63	0.89	0.45	0.49
User 2	0.27	0.53	0.35	0.8
User 3	1	0.74	0.27	0.33

HPCRec18 method limitations: In sequential pattern mining only add items which user has not purchased them, to the user item matrix. It needs clickstream data. In collaborative filtering, ratings based on purchase frequency and rating calculated based on similar users have the same weight.

CHAPTER 3

The Proposed Stacking Ensemble E-commerce Recommendation System (SEERS)

In this chapter, we introduce stacking ensemble e-commerce recommendation system (SEERS) which receives historical purchases of customers and uses a combination of collaborative filtering and sequential rule mining to recommend the best product to each user in an e-commerce dataset. All of the contributions, including steps and tasks required in each part will be explained in this section. Then we compare this proposed method with HOPE (Choi, Yoo, Kim, & Suh, 2012) e-commerce recommendation system through example. In the next chapter, precision, recall, and f1-score of the proposed method and its comparison with previous methods, including HOPE, are analyzed.

3.1 Input

The input of this method is a list of $<CustomerId, PurchasedItem, Transactiondate>$ (Table 22) which each row shows that a customer has purchased one or multiple items at a specific date. Our proposed algorithm analyses this historical purchase of customers to recommend the most relevant products. There might be many fields in e-commerce databases, but we only need customer transactions with these three fields in this research. For example, a sample purchase transaction database is demonstrated

in Table 22. In these input purchase transactions, User_1 has purchased products A and B together in 14^{th} of January 2018.

Table 22: Purchase transactions

Customer Id	**Transaction Date**	**Purchased Items**
User_3	10/1/2018	C
User_1	14/1/2018	AB
User_3	20/1/2018	AC
User_2	7/2/2018	B
User_1	5/3/2018	B
User_1	14/3/2018	AB
User_2	18/3/2018	CD
User_1	24/4/2018	BC
User_3	4/5/2018	C
User_3	19/5/2018	C
User_1	3/9/2018	D
User_2	28/11/2018	D

3.2 Output

The output of recommendation system is a list of top N recommended items to each user. These items are ranked based on their ratings. Sample output is demonstrated in Table 23. In this example, three products $\{D, B, C\}$ are recommended to user_1 which indicates for this user, products D, B and C have the highest probability to be purchased by the user in the future. Similarly, two items are recommended to other users too.

Table 23: Sample e-commerce recommendation system output

	Recommendation 1	**Recommendation 2**	**Recommendation 3**
User$_1$	B	D	C
User$_2$	D	C	B
User$_3$	C	A	D

3.2.1 Problem Definition

Given the customer purchase transaction data (e.g. Table 22) as input. The task of the e-commerce recommendation system is to recommend the best items with the highest F1 score to each user (e.g. Table 23).

3.3 Proposed Method

The proposed method in this thesis comprises of two phases, training and recommendation. In the first phase, we discover best support and confidence vales for sequential pattern rule mining, best number of similar users in collaborative filtering, and best weights for four intermediate user item matrices which are (a) sequential pattern mining rules for purchased items (SPM purchased) user item matrix, (b) sequential pattern mining rules for not purchased items (SPM not purchased) user item matrix, (c) collaborative filtering for not purchased (CF not purchased) user item matrix and purchase frequency user item matrix. But before explaining these two steps, we describe how we create intermediate user item matrices with sequential pattern mining and collaborative filtering.

3.3.1 Using Collaborative Filtering to Create User Item Matrices

In this step, first, we create the user item purchase frequency matrix based on the historical purchase of customers (Table 24). In this matrix, each rating indicates

purchased frequency of an item by a user. Then we normalize values (Table 25) and return it as user item purchase frequency matrix. Then we use collaborative filtering to find unknown ratings in this matrix and return these discovered ratings as another user item matrix for items that user has not purchased before (CF not purchased user item matrix). A detailed description of this process is explained in the following five steps.

1) Create the user item purchase frequency matrix. In this research, we extract implicit rating from the purchase frequency of users. In this method, any time a user purchased an item, we add one to the value of relative rating in the user item matrix. Table 24 shows the purchase frequency user item matrix, which is created from input purchase transactions of Table 22. For example this Table indicates that user_1 has purchased item_2 four times.

Table 24: Sample user item purchase frequency matrix

	Item$_1$	**Item$_2$**	**Item$_3$**	**Item$_4$**
User$_1$	2	4	1	1
User$_2$	0	1	1	2
User$_3$	1	0	4	0

2) Normalize user item purchase frequency matrix. We use normalization method to convert purchase frequencies to values between zero and one. We use vector normalization method in equation 1 to normalize the matrix values.

$$x'_{ui} = \frac{x_{ui}}{\sqrt{\sum_{i=1}^{n} x_{ui}^2}} \tag{1}$$

In this equation, x'_{ui} denote the normalized value, x_{ui} denote the rating we want to normalize, n denote number of ratings of the $user_u$, and x_{ui} denote rating of $user_u$ for $item_i$. For example normalized rating of user_1 for item_2 in user item matrix 0.85

because by using Equation 1 we have:

$$x'_{(user_1,item_2)} = \frac{4}{\sqrt{2^2+4^2+1^1+1^1}} = \frac{4}{\sqrt{22}} = \frac{4}{4.69} = 0.85$$

By using this Equation, all ratings in Table 24 are normalized, and the result is displayed in Table 25. We save it as the first intermediate user item matrix, which is purchased frequency matrix.

Table 25: Normalized user item purchase matrix

	Item$_1$	**Item$_2$**	**Item$_3$**	**Item$_4$**
User$_1$	0.43	0.85	0.21	0.21
User$_2$	-	0.41	0.41	0.82
User$_3$	0.24	-	0.97	-

3) Find similarity between users by using cosine similarity. This step receives the normalized user item purchase frequency matrix (Table 25) and finds the similarity between users by using cosine similarity (Equation 2). In this equation r_ui denote rating of $user_u$ for $item_i$ in user item matrix. I_u denote the index of items which $user_u$ has a rating for them.

$$CosineSimilarity(u,v) = \frac{\sum\limits_{i \in I_u \cap I_v} r_{ui}.r_{vi}}{\sqrt{\sum\limits_{i \in I_u} r_{ui}^2}\sqrt{\sum\limits_{i \in I_v} r_{vi}^2}} \tag{2}$$

Now we have a triangular user-user matrix which each field indicate similarity between two users. The similarities between all users are displayed in Table 26. For example similarity between user$_1$ and user$_2$ is calculate as follow:

$$Similarity(user_1, user_2) = \frac{(0.85*0.41)+(0.21*0.41)+(0.21*0.82)}{\sqrt{0.43^2+0.85^2+0.21^2+0.21^2}\sqrt{0.41^2+0.41^2+0.82^2}} = 0.61 \tag{3}$$

Table 26: Similarity between users

	$\mathbf{User_1}$	$\mathbf{User_2}$	$\mathbf{User_3}$
$\mathbf{User_1}$	-	0.61	0.31
$\mathbf{User_2}$	-	-	0.40
$\mathbf{User_3}$	-	-	-

4) Discover unknown ratings in the user item rating matrix based on the purchased frequency of similar users. User item purchase frequency matrix is very scarce because many users only buy a few items. In collaborative filtering method, we use the purchase frequency of items by similar users to discover and fill up more ratings in the user item matrix. We use equation 4 to discover unknown rating for $user_u$ and $item_i$.

$$rating_{\mathrm{ui}} = \frac{\sum\limits_{n \in P_{\mathrm{u}}(j)} Similarity(u,v).rating_{\mathrm{vi}}}{\sum\limits_{n \in P_{\mathrm{u}}(j)} Similarity(u,v)} \tag{4}$$

In this Equation, $P_u(i)$ denotes the set of k-most similar users who have a rating for $item_i$. $rating_{ui}$ is the rating of $user_u$ for $item_i$, and $Similarity(uv)$ is the similarity between $user_u$ and $user_v$. For example, if we want to recommend items by utilizing the purchase frequency of two similar users (k=2), then rating in user item matrix for $user_2$ and $item_1$ is calculated with the following formula:

$$rating_{\mathrm{u_2i_1}} = \frac{0.61 * 0.43 + 0.40 * 0.24}{0.61 + 0.40} = \frac{0.3583}{1.01} = 0.35 \tag{5}$$

In a similar way, if we calculate rating based on one similar user (k=1), then the rating will be 0.43 because:

$$rating_{\mathrm{u_2i_1}} = \frac{0.61 * 0.43}{0.61} = 0.43 \tag{6}$$

In Table 27 k=1 is used in collaborative filtering to calculate unknown ratings. In the rest of examples in this chapter, we use k=1 as number of similar users in

collaborative filtering.

Table 27: User item matrix of collaborative filtering with k=1

	Item$_1$	**Item$_2$**	**Item$_3$**	**Item$_4$**
User$_1$	0.43	0.85	0.21	0.21
User$_2$	0.43	0.41	0.41	0.82
User$_3$	0.24	0.41	0.97	0.82

5) Only save ratings discovered by collaborative filtering. In step two, normalized purchase frequency of items by all users are found. In this step, we remove them from the user item matrix and only keep ratings calculated by collaborative filtering. User has not purchased any of these items before. For example, in Table 27 three values are calculated with collaborative filtering, and the rest of the values are purchased frequency of user. Therefore we remove all purchase frequencies and keep three values calculated by collaborative filtering. The remaining values in this example are displayed in Table 28.

Table 28: User item matrix without purchased item of each customer

	Item$_1$	**Item$_2$**	**Item$_3$**	**Item$_4$**
User$_1$	-	-	-	-
User$_2$	0.43	-	-	-
User$_3$	-	0.41	-	0.82

6) Normalize the result and save it as *CFnotpurchased* user item matrix. Normalize all ratings in user item matrix (Table 27) and save it as the intermediate matrix. Table 29 displays the normalized matrix of Table 28.

Table 29: Normalized collaborative filtering user item matrix without purchased items of each user.

	$\mathbf{Item_1}$	$\mathbf{Item_2}$	$\mathbf{Item_3}$	$\mathbf{Item_4}$
$\mathbf{User_1}$	-	-	-	-
$\mathbf{User_2}$	1	-	-	-
$\mathbf{User_3}$	-	0.45	-	0.89

3.3.2 Create User Item Matrices by Mining Sequential Rules

Sequential rule mining can be used to discover what items will be purchased by users in the future. In this method, we extract the sequential database (Table 31) from purchase transactions (Table 22), then mining sequential rules from them. Finally, we create two intermediate user item matrices, which are $SPMpurchased$ and $SPMnotpurchased$ and normalize them. In this section, we explain step by step of this method in details.

1) Convert input transactions to customer sequences. We usually receive transactions data ($CustomerId, ProductId, PurchaseDate$) (Table 22) instead of sequential dataset (e.g. $< (B)(CD)(D) >$) (Table 31), therefore before running any sequential pattern mining method, we should convert input data to a list of sequences (sequential dataset). In the beginning, we should convert *transactions date* (e.g. 17/1/2019) to *transaction period* (e.g. Month 1) based on the interval (e.g. monthly, weekly, or daily). For example, if we use sequential pattern mining to find patterns in *montly* periods, then any purchase date in the first month is converted to purchase period of 1, any purchase date in the second month is converted to purchase period of 2. The same way we convert all purchase dates to purchase periods. Transaction periods are integer values (e.g. 1, 2, 3, etc.). Finally, we order all transactions based on the transaction period and customerId. The result of this step is displayed in Table 30.

Table 30: Sample transactions (Table 22) sorted by transaction period and CustomerId

Customer Id	**Transaction Period**	**Purchased Items**
$User_1$	1	AB
$User_1$	3	AB
$User_1$	4	BC
$User_1$	9	D
$User_2$	2	B
$User_2$	3	CD
$User_2$	11	D
$User_3$	1	AC
$User_3$	5	C

Now we can create the purchase sequential database (Table 31) for each customer in Table 30 by combining all purchased item of a customer in a transaction period. To do this, we categorize transactions by Transaction period and customer Id. For example, in Table 31 transaction sequences for each customer are displayed. As you can see, all purchased items are grouped by transaction period and customer id. For example, in Table 30 $user_1$ has purchased (AB) in the first month, (AB) in the third month, (BC) in the fourth month and (D) in the ninth month, therefore his transaction sequence in the sequential database (Table 31) is $< (AB)(AB)(BC)(D) >$.

Table 31: Sequential database of Table 22 and Table 30

Customer Id	**Sequences**
$User_1$	<(AB)(AB)(BC)(D)>
$User_2$	<(B)(CD)(D)>
$User_3$	<(AC)(C)>

2) Find Frequent Sequences. In this step, we receive minimum support (e.g.

50 percent) and find all frequent sequences with a support higher than the minimum support. Output of this step is list of frequent sequences with their support (e.g. Table 32). For example, in Table 31 item C exist in three user sequences, $< (A)(C) >$ exist in two sequences and $< (C)(A) >$ exist in one sequence. $< (A)(C) >$ means customers buy item A and in next months buy item C. We only keep frequent sequences, for example if we receive minimum support of 50 percent, then each sequence must exist in 50 percent of input sequences. As a results, $< (A)(C) >$ is a frequent sequence, but $< (C)(A) >$ is not frequent because it exists in only one input sequences, so its support is 33 percent which is less than the 50 percent minimum support. Table 32 shows the list of frequent sequences in the sequential database (Table 31) with support higher than 50 percent.

Table 32: Frequent sequences of input sequential dataset

Sequence	**Support**	**Support%**
(A)	2	66
(B)	2	66
(C)	3	100
(D)	2	66
(A)(C)	2	66
(B)(C)	2	66
(B)(D)	2	66
(C)(D)	2	66
(B)(C)(D)	2	66

3) Find frequent sequential rules. Frequent sequences with their support are calculated in the previous step (Table 32). Now we find sequential rules and calculate their confidence. We only keep rules that have confidence higher than the predefined minimum confidence (e.g. 50 percent). From frequent sequence $< (A)(B) >$ we can create rule $A \rightarrow B$. This rule indicates that if a user buys item A then he might

buy item B with a probability equal to this rule's confidence. Equation 7 is used to calculate the confidence of each rule. In this equation to calculate confidence of $A \rightarrow B$, we divide support of sequence $< (A)(B) >$ by the support of sequence $< (A) >$.

$$Confidence(A => B) = P(B|A) = \frac{support(A \cup B)}{support(A)} \tag{7}$$

For example, by using equation 7 confidence of sequential rule $A \rightarrow C$ is 100 percent because the support of sequence $< (A)(C) >$ is 66 percent and the support of $< (A) >$ is 66 percent. If we divide 0.66 by 0.66, the result is equal to 1.

$$Confidence(A => C) = P(C|A) = \frac{0.66}{0.66} = 1$$

Table 33 shows frequent rules with minimum support and confidence higher than 50 percent.

Table 33: Frequent sequential rules of input sequential dataset

Antecedent	**Consequent**	**Support%**	**Confidence%**
(A)	(C)	66	100
(B)	(C)	66	100
(B)	(D)	66	100
(C)	(D)	66	66
(B)(C)	(D)	66	100
(B)	(C)(D)	66	100

4) Calculate ratings. To find $Rating(u, i)$, select all proper rules (PR) from input frequent sequential rules (Table 33) which satisfy two conditions: first, rule's consequent is equal to $item_i$, second, rule's antecedent is a subsequence of $user_u$ purchase sequence. Sum of these rule's support is the rating for $user_u$ and $item_i$ in user item matrix. Equation 8 is used to calculate the rating. In this Equation, $Rating_{ui}$ denote calculated rating of $user_u$ for $item_i$ in the user item matrix (Table

34). R denotes all rules calculated in the previous step (Table 33). PR denote proper rules which satisfy two mentioned condition in this step.

$$Rating_{ui} = \sum_{R \epsilon PR} Support_R \tag{8}$$

For example in Table 33 there are two rule that C is the consequent item which are $A \rightarrow C$ and $B \rightarrow C$. Also $user_1$ has already purchased antecedent items (A in $A \rightarrow C$ rule and B in $B \rightarrow C$ rule), so both of them satisfy the two condition. As a results rating of $user_1$ for $item_C$ is 1.32 because sum of the support for two rules is $0.66 + 0.66 = 1.32$. In a similar way all ratings are calculated and saved in user item matrix.

Table 34: User item rating in sequential pattern mining

	A	**B**	**C**	**D**
User$_1$	-	-	1.32	1.98
User$_2$	-	-	0.66	1.98
User$_3$	-	-	0.66	0.66

5) Save ratings in two user item matrices. If user$_u$ has purchased item$_i$, save $Rating(u, i)$ inside $SPMpurchased$ user item matrix. If user$_u$ has not purchased item$_i$, save $Rating(u, i)$ inside $SPMnotpurchased$ user item matrices.

Table 35: User item rating in sequential pattern mining for purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	1.32	1.98
User$_2$	-	-	0.66	1.98
User$_3$	-	-	0.66	-

Table 36: User item rating in sequential pattern mining for not purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	-	-
User$_2$	-	-	-	-
User$_3$	-	-	-	0.66

6) Normalize user item matrix. We use normalization method to convert ratings in user item matrix to values between zero and one. We use vector normalization method in Equation 1 to normalize the matrix values.

Table 37: Normalized user item rating in sequential pattern mining for purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	0.55	0.83
User$_2$	-	-	0.32	0.95
User$_3$	-	-	1	-

Table 38: Normalized user item rating in sequential pattern mining fro not purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	-	-
User$_2$	-	-	-	-
User$_3$	-	-	-	1

3.3.3 Phase 1: Training

In this research, we use stacking ensemble learning method to train the recommendation system. For e-commerce recommendation systems, this proposed hybrid method has better accuracy and consistency, and less bias and overfitting problem compare to

other ensemble learning methods. This method consists of two phases: training and recommendation. We also divide input purchase frequency dataset into three parts, which are training, verification, and test dataset. In training phase, we use training dataset as input and test it over verification dataset and use supervised learning to discover best values for support and confidence of sequential pattern mining, k similar user in collaborative filtering and best weights for each intermediate user item matrices which leads to the highest $F1score$. Overview of the training phase is demonstrated in Figure 5. In this section, three steps of the training phase are explained.

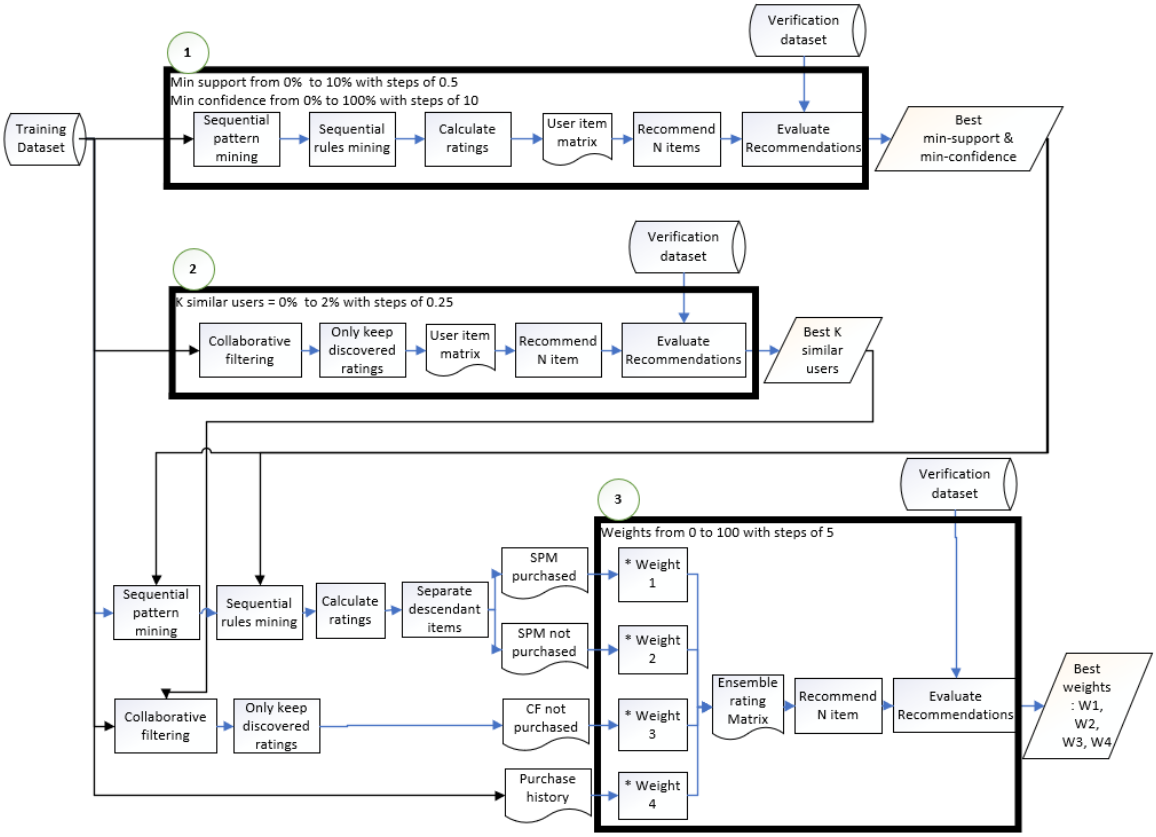

Fig. 5: Overview of training in Stacking Ensemble E-commerce Recommendation System (SEERS)

Step 1.1) Receive training dataset as input, in a loop, use sequential rule mining method explained in section 3.3.2 with various minimum support and confidence value. We use minimum support from 1 percent to 10 percent with the step of 0.5 and minimum confidence from 0 to 100 percent with step of 10. Finally create the user item matrix (section 3.3.2), then recommend top N items with highest ratings

in the user item matrix to each user and calculate the accuracy of recommended items over verification dataset. Then the minimum support and confidence which generate the highest $F1score$ is selected. Table 39 shows the best minimum support and confidence of sequential rule mining when we recommend 20 items to each user. In this example, the best minimum support is 0.5 percent and minimum confidence is 30 percent.

Table 39: Best support and confidence of sequential rule mining with their F1 score

Support	**Confidence**	**F1Score Rec 5**	**F1Score Rec 10**	**F1Score Rec 20**	**F1Score Rec 50**
0.5	30	0.0691	0.0949	0.1350	0.1543
0.5	40	0.0750	0.1031	0.1344	0.1537
1	30	0.0641	0.0892	0.1279	0.1559
2.5	30	0.0656	0.0912	0.1267	0.1563
2	30	0.0630	0.0877	0.1262	0.1606
3	30	0.0630	0.0890	0.1251	0.1490
1.5	30	0.0638	0.0894	0.1251	0.1587
0.5	50	0.0558	0.0855	0.1230	0.1506
3.5	30	0.0606	0.0922	0.1221	0.1372

Step 1.2) In a loop use collaborative filtering method with various K similar neighbors to create user item matrix for not purchased items as explained in section 3.3.1. This method receives training dataset as input and generates the user item matrix as output. Then recommends top N items with the highest ratings in the user item matrix to each user. Finally, calculate the F1 score of recommended items over verification dataset. The K value, which gives the highest F1 score, is selected (e.g: k = 0.25). Table 40 shows F1 score of recommended items when 5, 10, 20 and 50 items are recommended. in this example, k=0.25 percent is the best one because it leads to the highest F1 score.

Table 40: F1 score of collaborative filtering for not purchased item with various number of similar users

	0.25	**0.5**	**0.75**	**1**	**1.25**	**1.5**	**1.75**	**2**
5	0.0023	0.0014	0.0009	0.0005	0.0005	0.0005	0.0005	0.0005
10	0.0028	0.0012	0.0020	0.0008	0.0008	0.0012	0.0012	0.0012
20	0.0059	0.0053	0.0046	0.0033	0.0033	0.0030	0.0026	0.0030
50	0.0122	0.0109	0.0099	0.0088	0.0078	0.0067	0.0065	0.0059

Step 1.3) Receive four intermediate user item matrices and give each one of them a weight, for example, $w1$ for $SPM\ purchased$, $w2$ for $SPM\ not\ purchased$, $w3$ for $collaborative\ filtering\ not\ purchased$ and $w4$ for $purchase\ frequency$. Then create the ensemble user item matrix by calculating integrated ratings, which are the weighted sum of ratings in all four intermediate user item matrices : $Ensemble = w1 * SPMpurchased + w2 * SPMnotpurchased + w3 * CFnotpurchased + w4 * Purchasefrequency$. We run the training with various weights (from 0 to 100 with steps of 5) for $w1$, $w2$, $w3$ and $w4$. Finally, recommend N items from the created ensemble user item matrix over verification dataset and select the best four weights, which leads to the highest F1 score. For example, Table 41 shows the best discovered weights in the input purchase dataset.

Table 41: Best weights for each user item matrix with their F1 score

PurchaseWeight	**CF NotPurcased Weight**	**SpmPurchased Weight**	**Spm NotPurchased Weight**	**F1Score Rec 5**	**F1Score Rec 10**	**F1Score Rec 20**	**F1Score Rec 50**
95	5	60	10	0.0697	0.1147	0.1525	0.1695
95	10	60	10	0.0697	0.1147	0.1525	0.1695
95	5	60	5	0.0697	0.1147	0.1525	0.1692
95	15	60	10	0.0697	0.1147	0.1522	0.1694
95	15	60	15	0.0697	0.1147	0.1522	0.1694
95	20	60	15	0.0697	0.1147	0.1522	0.1694
95	5	60	15	0.0697	0.1147	0.1522	0.1692
95	10	60	15	0.0697	0.1147	0.1522	0.1692
85	20	55	20	0.0708	0.1147	0.1520	0.1683

3.3.4 Phase 2: Recommend Items

In training phase (section 3.3.3), we discovered best minimum support and minimum confidence for sequential rule mining, best k similar users in collaborative filtering and best weights for four intermediate user item matrices which are created by using *SPM purchased*, *SPM notpurchased* (section 3.3.2), collaborative filtering not purchased (section 3.3.1) and *purchase frequency*. In this step, we receive training and verification dataset as input and use discovered best *minimum support* (e.g. 0.5%), *minimum confidence* (e.g. 30%), k similar users (e.g. 0.25%) and four best weights (e.g. w1=60, w2=10, w3=5, w4=95) to create the ensemble user item matrix (Table 46) and recommend best N items to each user (Table 47). Overview diagram of this process is displayed in Figure 6.

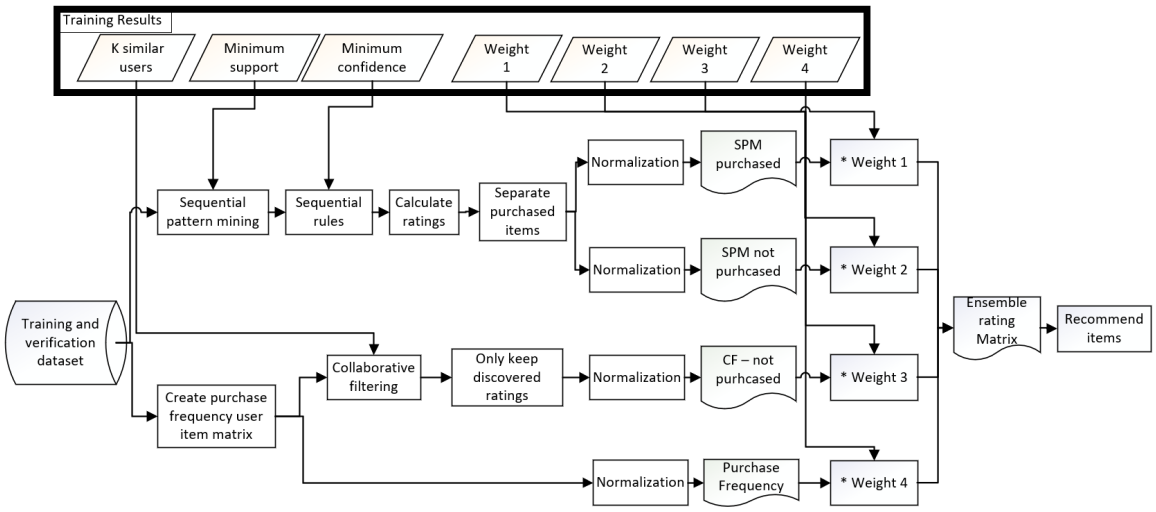

Fig. 6: Overview of recommendation in Stacking Ensemble E-commerce Recommendation System (SEERS)

Step 2.1) Run sequential rule mining with discovered minimum support and confidence (discovered in section 3.3.3) over train and verification dataset and create *SPMpurchased* user item matrix as explained in section 3.3.2.

Table 42: Normalized user item ratings in sequential pattern mining for purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	0.55	0.83
User$_2$	-	-	0.32	0.95
User$_3$	-	-	1	-

Step 2.2) Run sequential rule mining, as explained in section 3.3.2 with discovered minimum support and confidence (discovered in section 3.3.3) over train and verification dataset and create $SPMnotpurchased$ user item matrix.

Table 43: Normalized user item rating in sequential pattern mining fro not purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	-	-
User$_2$	-	-	-	-
User$_3$	-	-	-	1

Step 2.3) Run collaborative filtering (explained in section 3.3.1) with discovered best K similar users (discovered in section 3.3.3) over train and verification dataset and create $CFnotpurchased$ user item matrix.

Table 44: Normalized collaborative filtering user item matrix without purchased item of each user

	A	**B**	**C**	**D**
User$_1$	-	-	-	-
User$_2$	1	-	-	-
User$_3$	-	0.45	-	0.89

Step 2.4) Create purchase frequency user item matrix (explained in section 3.3.1) from historical purchases in train and verification dataset.

Table 45: Normalized user item purchase matrix

	A	**B**	**C**	**D**
User$_1$	0.43	0.85	0.21	0.21
User$_2$	-	0.41	0.41	0.82
User$_3$	0.24	-	0.97	-

Step 2.5) In step 2.1 to 2.4, four intermediate user item rating matrices are created. Also, the best weight for each one is discovered in step 1. Now we use equation 3.3.4 to find ensemble rating of each user item. Final values are displayed in Table 46.

$$Ensemble_{\text{ui}} = (w1{*}SequentialRulePurchased){+}(w2{*}SequentialRuleNotPurchased) + (w3 * CollaborativeFiltering) + (w4 * PurchaseHistory) \quad (9)$$

Table 46: Ensemble user item matrix

	A	**B**	**C**	**D**
User$_1$	40.85	80.75	52.95	69.75
User$_2$	5	38.95	58.15	134.9
User$_3$	22.8	2.25	152.15	14.45

Step 2.6) Recommend top N product to each user. We get the ensemble user item matrix from the previous step and recommend top N items. For example, if we want to recommend three items, then the final output of this system is shown in Table 47.

Table 47: Top 2 items recommended to each user with their normalized purchase frequency

	Rec 1	**Rate 1**	**Rec 2**	**Rate 2**	**Rec 2**	**Rate 2**
$\mathbf{User_1}$	B	80.75	D	69.75	C	52.95
$\mathbf{User_2}$	D	134.9	C	58.15	B	38.95
$\mathbf{User_3}$	C	152.15	A	22.8	D	14.45

3.4 Comparison of HOPE vs SEERS Through an Example

In the previous section, we take input purchase data in Table 22 and used SEERS method to recommend products. In this section to compare the method, we use the same data as input to HOPE method to recommend products and compare the final results with SEERS method.

3.4.1 Problem Definition

As it is mentioned, input data to both HOPE and SEERS is purchase transaction data (e.g. Table 22).

Problem: Given the customer purchase transaction data of Table 22 as input, from which user item matrices are created; The task of the recommendation system is to recommend best items that have the highest $F1score$, to each user. Two recommendation systems to be used to solve this problem are (1) HOPE (Choi, Yoo, Kim, & Suh, 2012) and (2) SEERS, which is proposed by this thesis.

3.4.2 Solution 1: HOPE Method

This section presents the framework for solving the problem, giving the input purchased data (Table 22), using HOPE recommendation system approach and its results. The general framework for solving this problem with HOPE approach is provided as

Figure 7. In hope method, the input purchase data is used in collaborative filtering and sequential pattern mining methods to predict the rating for each user item. Then these two values are integrated, and items with the highest rating are recommended.

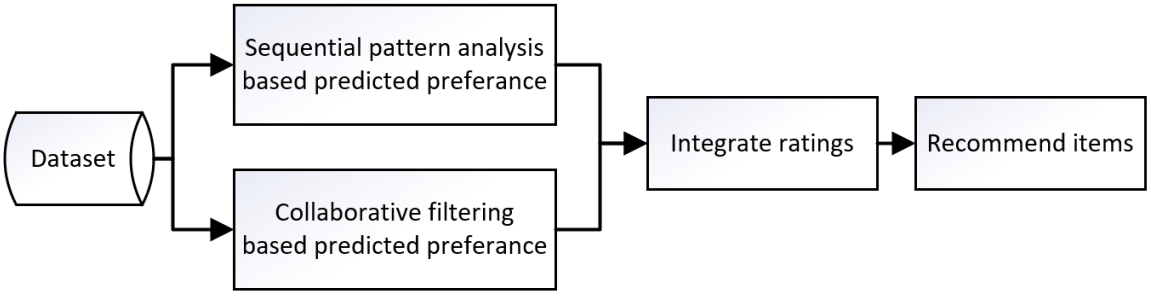

Fig. 7: HOPE Framework

The HOPE recommendation system first creates the collaborative filtering user item matrix from the input purchase transaction data (Table 48). Then uses the user item rating matrix of Table 48 to run collaborative filtering mining algorithm to find unknown values in this matrix (Table 53). It also uses sequential pattern mining to create the user item matrix (Table 55), from the input purchase data of Table 22. To compute the frequent purchase sequence from the purchase data, HOPE first creates a sequential database of Table 22 as shown in Table 31. HOPE will find frequent sequences from this sequential database of 31 with user-specified minimum support. After running SPM algorithm, it found the frequent sequences shown in Table 54. Next HOPE finds the probability of a user purchasing any item from getting the sum of the percentage of all frequent sequences that have the item as their last item in the sequence. For example, in frequent sequences in Table 54, item C is the last item in the two frequent sequences $(A)(C)$ and $(B)(C)$ both with support 0.66. This causes the probability of item C being purchased next to be $0.66 + +0.66 = 1.32$ as in Table 55

Next, the computed SPM rating matrix of Table 56 is normalized to obtain Table 57, while the collaborative filtering matrix is normalized too. Then these two values are integrated by giving the weight of 90 percent to sequential pattern mining results and 10 percent to collaborative filtering results. Finally, for each user, items with the highest rating will be recommended. Overview of this framework is displayed in

Figure 8.

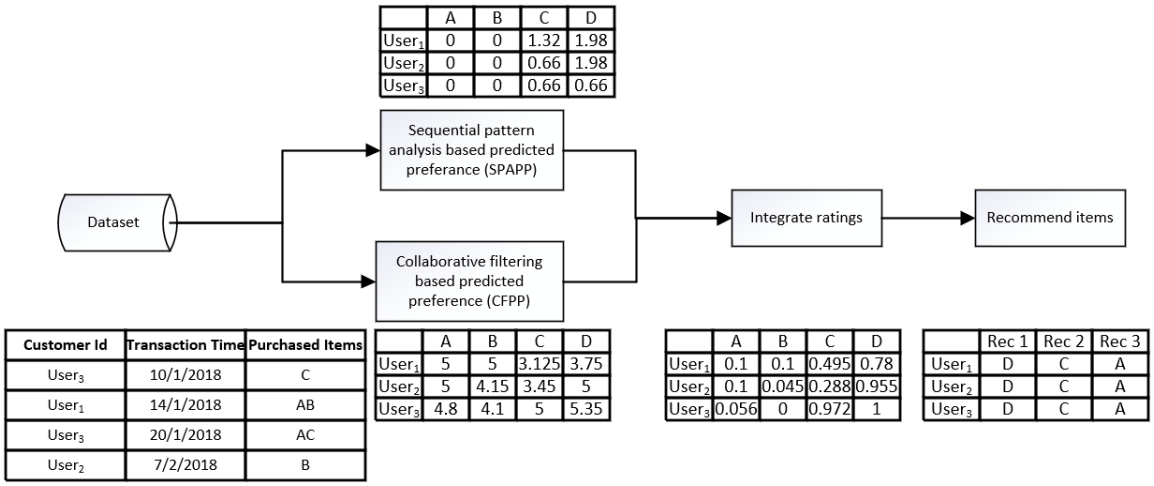

Fig. 8: HOPE Framework with Table 22 as input

In this part, each step of HOPE framework is explained in more detail with examples.

1) Collaborative filtering-based recommendation. First, create the user item rating matrix (Table 48) from input historical purchase data (Table 22). Each rating in this matrix, indicates purchase frequency of an item by a user. Then by using the next four steps, unknown ratings inside this matrix will be calculated.

Table 48: User item rating matrix from Table 22

	A	**B**	**C**	**D**
$User_1$	2	4	1	1
$User_2$	-	1	1	2
$User_3$	1	-	4	-

1.1) Calculate absolute preference (Table 49). Count number of transactions that a user has bought that item and divide it to the total number of user's transactions and add one to the final result. For $user_u$ and $product_i$ Equation 10 is used for calculating absolute preference. This method is used to calculate the absolute rating for all values inside the purchase frequency user item matrix (Table 48). Results are displayed in Table 49.

$$AP(u,i) = \frac{\text{Number of tranactions } user_u \text{ has bought } item_i}{\text{Total number of } user_u \text{ transactions}} + 1 \tag{10}$$

Table 49: Absolute preference of ratings in user item purchase frequency matrix

	A	**B**	**C**	**D**
User$_1$	1.25	1.5	1.125	1.125
User$_2$	-	1.25	1.25	1.5
User$_3$	1.20	-	1.80	-

1.2) Calculate relative preference (Table 50). For user_u and item_i, use Equation 3 to find relative preference.

$$RP(u,i) = \frac{AP(u,i)}{\max\limits_{c \epsilon U} AP(c,i)} \tag{11}$$

In this equation, U indicates users who purchase item_i. $AP(u,i)$ indicate absolute preference of $user_u$ for $item_i$. $AP(c,i)$ indicates absolute preference of $user_c$ for $item_i$. By using this equation all absolute preferences (Table 49) are converted to relative preference (Table 50).

Table 50: Absolute preference of ratings in user item purchase frequency matrix

	A	**B**	**C**	**D**
User$_1$	1	1	0.625	0.75
User$_2$	-	0.83	0.69	1
User$_3$	0.96	-	1	-

1.3) Multiply ratings by five (Table 51). Relative values from the previous step (Table 50) are multiplied by 5 and then it is round up to create user item matrix in table 51.

Table 51: Multiply ratings by five in the user item purchase frequency matrix

	A	**B**	**C**	**D**
User$_1$	5	5	3.125	3.75
User$_2$	-	4.15	3.45	5
User$_3$	4.8	-	5	-

1.4) Finding k nearest neighbors of each user (Table 52) by using Pearson similarity (Equation 12). In this Equation, u and v are two users that we want to find their similarity. $\bar{R}_u$ and $\bar{R}_v$ are average ratings of user$_u$ and user$_v$.

$$PearsonSimilarity(u,v) = \frac{\sum_{i=1}^{n}(R_{ui} - \bar{R}_u)(R_{vi} - \bar{R}_v)}{\sqrt{\sum_{i=1}^{n}(R_{ui} - \bar{R}_u)^2}\sqrt{\sum_{i=1}^{n}(R_{vi} - \bar{R}_v)^2}} \tag{12}$$

For example we have $\bar{R_{u1}} = 4.2$, $\bar{R_{u2}} = 4.2$, $\bar{R_{u3}} = 4.9$.

$$Sim(user_1, user_2) = \frac{(5-4.2)*(4.15-4.2)+(3.125-4.2)*(4.45-4.2)+(3.75-4.2)*(5-4.2)}{\sqrt{(5-4.2)^2+(3.125-4.2)^2+(3.75-4.2)^2}\sqrt{(4.15-4.2)^2+(3.45-4.2)^2+(5-4.2)^2}}$$

$$= -0.26$$

$$Sim(user_1, user_3) = \frac{(5-4.2)*(4.8-4.9)+(3.125-4.2)*(5-4.9)}{\sqrt{(5-4.2)^2+(3.125-4.2)^2}\sqrt{(4.8-4.9)^2+(5-4.9)^2}} = -0.99$$

$$Sim(user_2, user_3) = \frac{(3.45-4.2)*(5-4.9)}{\sqrt{(3.45-4.2)^2}\sqrt{(5-4.9)^2}} = -1$$

Table 52: Similarity between users

	User$_1$	**User$_2$**	**User$_3$**
User$_1$	-	0.26	-0.99
User$_2$	-	-	-1
User$_3$	-	-	-

1.5) Discover unknown ratings. (Table 53). Use Equation 13 to calculate unknown

values for each user item. $Sim(a,b)$ is similarity between $user_a$ and $user_b$. $\bar{R}_a$ and $\bar{R}_b$ are average ratings of $user_u$ and $user_v$.

$$CF(a,i) = \bar{R}_a + \frac{\sum_{b=1}^{n} Sim(a,b) * (R_{bi} - \bar{R}_b)}{\sum_{b=1}^{n} |Sim(a,b)|} \tag{13}$$

Table 53: Output matrix of collaborative filtering on user item matrix of Table 48

	A	**B**	**C**	**D**
User$_1$	5	5	3.125	3.75
User$_2$	5	4.15	3.45	5
User$_3$	4.8	4.1	5	5.35

2.1) Find sequential patterns in all users' transaction sequences except target user. (Table 54).

Table 54: Frequent sequences

Sequence	**Support**
(A)(C)	0.66
(B)(C)	0.66
(B)(D)	0.66
(C)(D)	0.66
(B)(C)(D)	0.66

2.2) Find each user's subsequences and compare it with frequent sequences (Table 54) to calculate sequential pattern analysis-based prediction. For example, when user purchase sequence is <item1, item2, item3> then its subsequences are <item1>, <item2>, <item3>, <item1, item2>, <item1, item3>, <item2, item3>, <item1, item2, item3>. If it is matched with starting part of a frequent sequence, then the next item in the frequent sequence is a candidate for recommendation. Finally,

calculate support value of candidate item which is total support of that item in all frequent sequences (Equation 6) and return those which have support value, higher than predefined minimum support.

$$SPAPrediction(a, i) = \sum_{s \epsilon SUB} Support_s^i \tag{14}$$

Table 55: Next purchase product rating calculated by sequential pattern mining

	A	**B**	**C**	**D**
User$_1$	0	0	1.32	1.98
User$_2$	0	0	0.66	1.98
User$_3$	0	0	0.66	0.66

3) Normalized collaborative filtering-based ratings (Table 56) and sequential pattern analysis-based ratings (Table 57) by using min max normalization formula. $Min(x)$ is minimum rating of $user_x$, $Max(x)$ is maximum rating of $user_x$ and x_i is rating of $user_x$ for $item_i$ which we want to normalize it.

$$Normalized(x_i) = \frac{x_i - Min(x)}{Max(x) - Min(x)} \tag{15}$$

Table 56: Normalized collaborative filtering-based ratings

	A	**B**	**C**	**D**
User$_1$	1	1	0	0.33
User$_2$	1	0.45	0	1
User$_3$	0.56	0	0.72	1

Table 57: Normalized Sequential pattern analysis-based prediction

	A	**B**	**C**	**D**
User$_1$	0	0	0.66	1
User$_2$	0	0	0.33	1
User$_3$	0	0	1	1

4) Give 0.1 weight to collaborative filtering and 0.9 weight to sequential pattern mining results to integrate them (Table 58).

Table 58: Sequential pattern analysis-based prediction

	A	**B**	**C**	**D**
User$_1$	0.1	0.1	0.495	0.78
User$_2$	0.1	0.045	0.288	0.955
User$_3$	0.056	0	0.972	1

5) Recommend items which have the highest rating (Table 59).

Table 59: Recommended items using hope method

	Rec 1	**Rec 2**	**Rec 3**
User$_1$	D	C	A
User$_2$	D	C	A
User$_3$	D	C	A

As we see in final results, this method recommends items regardless of if the user has purchased the item before or not.

3.4.3 SEERS Method

We have already explained the proposed method at the beginning of this chapter, but for comparison to HOPE system, in this section, we show how SEERS method

recommends products, briefly. Overview of this method is displayed in Figure 9 with example.

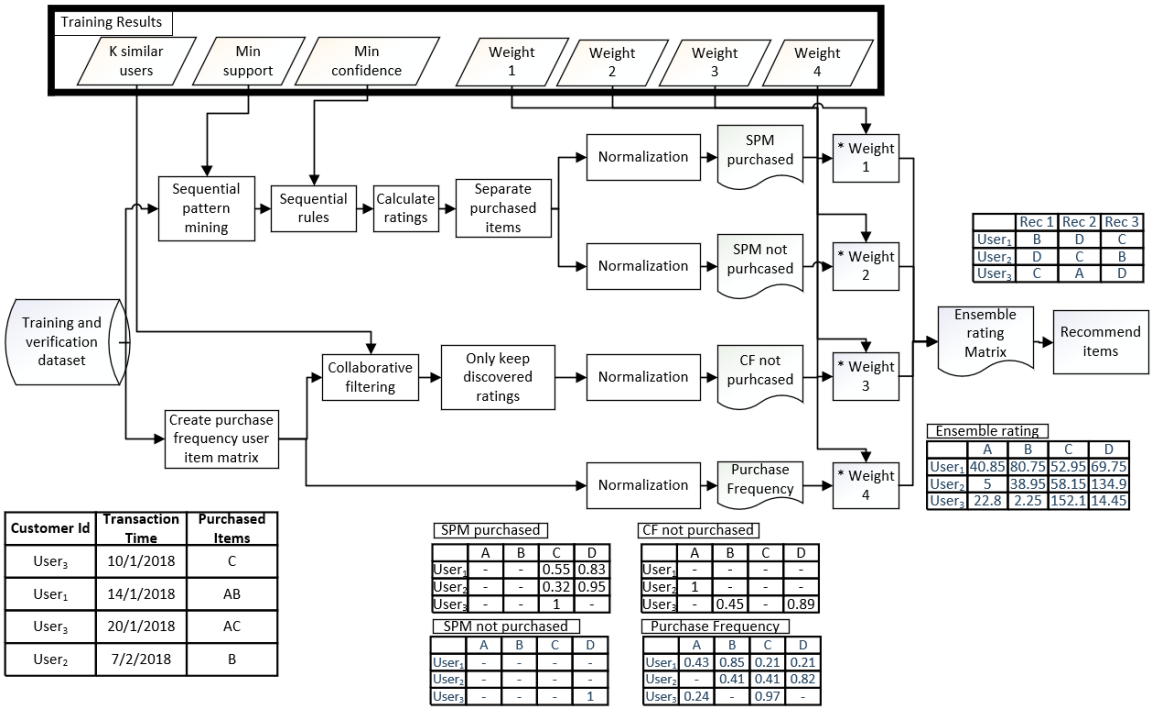

Fig. 9: SEERS Framework with Table 22 as input

1) Run sequential rule mining (section 3.3.2) with discovered minimum support (section 3.3.3) and create the user item matrix for items which user has already purchased the consequent of the sequential rule. Then normalize all ratings using Equation 1 (Table 60).

Table 60: Normalized user item rating in sequential pattern mining for purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	0.55	0.83
User$_2$	-	-	0.32	0.95
User$_3$	-	-	1	-

2) Run sequential rule mining (section 3.3.2) with discovered minimum support and confidence (section 3.3.3) and create the user item matrix for items which user

has not purchased yet and normalize the ratings (Table 61).

Table 61: Normalized user item rating of sequential pattern mining for not purchased items

	A	**B**	**C**	**D**
User$_1$	-	-	-	-
User$_2$	-	-	-	-
User$_3$	-	-	-	1

3) Run collaborative filtering (section 3.3.1) with discovered best K similar users (section 3.3.3) and create collaborative not purchased user item matrix. Then normalize the ratings (Equation 1).

Table 62: Normalized collaborative filtering user item matrix without purchased item of each user

	A	**B**	**C**	**D**
User$_1$	-	-	-	-
User$_2$	1	-	-	-
User$_3$	-	0.45	-	0.89

4) Create purchase frequency user item matrix from input purchase transactions and normalized ratings.

Table 63: Normalized purchase frequency user item matrix

	A	**B**	**C**	**D**
User$_1$	0.43	0.85	0.21	0.21
User$_2$	-	0.41	0.41	0.82
User$_3$	0.24	-	0.97	-

5) Use the best weights discovered in section 3.3.3 for each intermediate user item matrix, then use Equation 3.3.4 to find rating for each user item. Final matrix is

displayed in Table 46.

Table 64: Ensemble user item matrix

	A	**B**	**C**	**D**
User$_1$	40.85	80.75	52.95	69.75
User$_2$	5	38.95	58.15	134.9
User$_3$	22.8	2.25	152.15	14.45

6) Recommend top N product to each user. We receive integrated user item matrix from the previous step and recommend top N items. For example, if we want to recommend three items, then the final output of this system is shown in Table 65.

Table 65: Top 3 items recommended to each user

	Rec 1	**Rec 2**	**Rec 3**
User$_1$	B	D	C
User$_2$	D	C	B
User$_3$	C	A	D

CHAPTER 4

Comparative Analysis

In this chapter, the accuracy of different recommendation systems, including e-commerce recommendations based on collaborative filtering, sequential pattern and rule mining, purchase frequency, and combination of these methods are evaluated. To evaluate accuracy of these recommendations, precision (Equation 5), recall (Equation 6) and f1-score (Equation 7) are used.

4.1 Datasets

In this research, we use the online retail dataset provided by University of California Irvin(UCI) machine learning repository (`https://archive.ics.uci.edu/ml/datasets/online+retail`) (Dua & Graff, 2017). This dataset contains all the transactions of a UK-based and registered non-store online retailer. This dataset has 541909 instances. Each instance is a transaction which has eight attributes: CustomerID, Stockcode, InvoiceDate, InvoiceNo, Description, Quantity, UnitPrice, Country. In this research, we select this dataset, because it has more than half a million transactions with 4372 distinct users and 3958 distinct product and it has three attributes we need (customerId, ProductId, transactionDate) . We only need three attributes, which are CustomerID, Stockcode, InvoiceDate. In this dataset, transaction dates are from 01/12/2010 until 09/12/2011. In this research, transactions from 01/12/2010 until 30/9/2011 are used to train the recommendation systems, transactions from 01/10/2011 until 31/10/2011 are used as verification dataset and transactions from 01/11/2011 until 30/11/2011 are used as test dataset to calculate

precision, recall and f1 score of each recommendation methods and compare their accuracy.

4.2 Implementation and Tools

Testing environment

- Operation system: 64-bit Windows 10 enterprise, version 10.0.17763
- System type: x64 based processor
- CPU: Intel core i7-4790 with 3.6GHz frequency
- Ram: 16 GB

Development tools

- Language: C#
- Development tool: Microsoft Visual Studio enterprise 2019 , version 16.0.1
- Software framework: Microsoft .Net Core 2.1, Microsoft ASP.NET Core MVC
- Database: Microsoft SQL Server enterprise 2017, version 14.0.2002.14

4.3 Evaluation Results and Analysis

In this section precision, recall and f1-score of each mentioned recommendation system and the proposed SEERs system is analyzed and compared.

4.3.1 Sequential Rule Mining Recommendation Evaluation

In this method, we find sequential rules with various support and confidence. Then recommend products to users based on discovered sequential rules. In the training phase of the proposed method in this research, we test sequential rule mining with various minimum support and confidence to find the best values which are lead to the

highest F1 score. In the following three figures and tables, sequential pattern mining method is used with minimum support values from 1 to 10 percent with steps of 0.5. Then precision, recall and F1-score of each method is calculated. Precision (Table 66 and Figure 10) , recall (Table 67 and Figure 11) and f1 score (Table 68 and Figure 12) of this method with various support values are demonstrated here.

Table 66: Precision of SPM recommendation with various support

	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
5	0.2226	0.2070	0.2032	0.2129	0.2178	0.2256	0.2331	0.2423	0.2590	0.2581	0.2786	0.2941	0.3364	0.3742	0.4227	0.4227	0.4815	0.4737	0.4737
10	0.1652	0.1635	0.1608	0.1709	0.1727	0.1781	0.1954	0.2078	0.2215	0.2368	0.2643	0.2813	0.3363	0.3742	0.4227	0.4227	0.4815	0.4737	0.4737
20	0.1404	0.1390	0.1419	0.1433	0.1464	0.1509	0.1571	0.1822	0.2107	0.2347	0.2639	0.2813	0.3363	0.3742	0.4227	0.4227	0.4815	0.4737	0.4737
50	0.1092	0.1094	0.1104	0.1159	0.1210	0.1389	0.1491	0.1762	0.2136	0.2347	0.2639	0.2813	0.3363	0.3742	0.4227	0.4227	0.4815	0.4737	0.4737

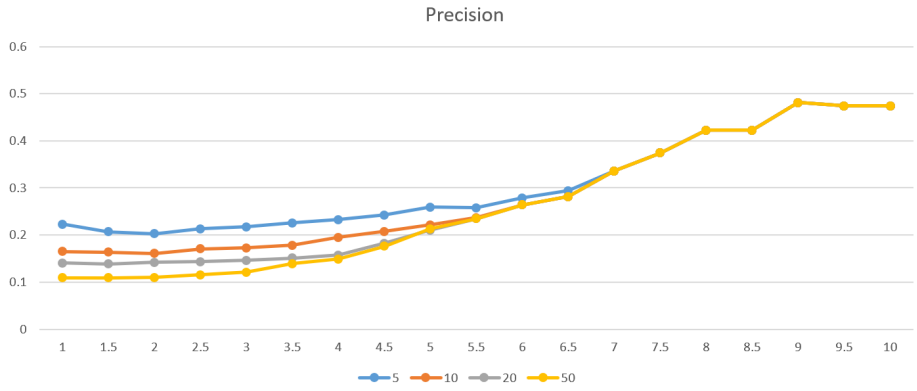

Fig. 10: Precision of SPM recommendation with various support

Table 67: Recall of SPM recommendation with various support

	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
5	0.0340	0.0316	0.0308	0.0316	0.0319	0.0319	0.0321	0.0313	0.0305	0.0276	0.0252	0.0239	0.0196	0.0154	0.0109	0.0109	0.0104	0.0048	0.0048
10	0.0504	0.0499	0.0483	0.0502	0.0494	0.0488	0.0523	0.0496	0.0470	0.0417	0.0342	0.0287	0.0202	0.0154	0.0109	0.0109	0.0104	0.0048	0.0048
20	0.0857	0.0847	0.0844	0.0823	0.0812	0.0796	0.0757	0.0733	0.0682	0.0510	0.0353	0.0287	0.0202	0.0154	0.0109	0.0109	0.0104	0.0048	0.0048
50	0.1667	0.1641	0.1587	0.1585	0.1508	0.1447	0.1128	0.0863	0.0727	0.0510	0.0353	0.0287	0.0202	0.0154	0.0109	0.0109	0.0104	0.0048	0.0048

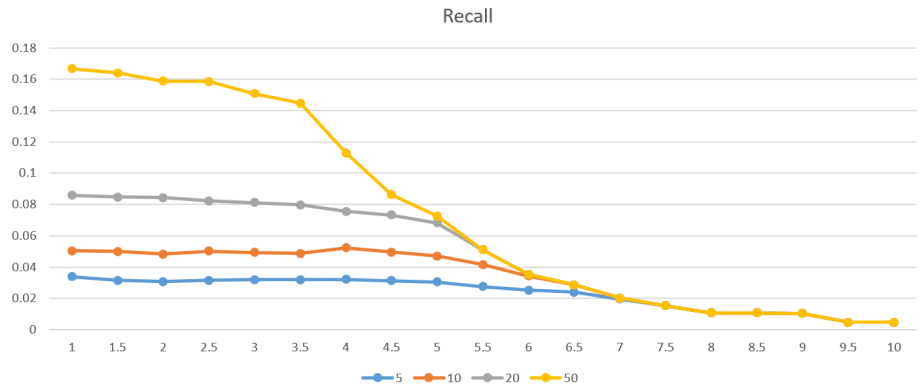

Fig. 11: Recall of SPM recommendation with various support

Table 68: F1 score of SPM with various support

	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7	7.5	8	8.5	9	9.5	10
5	0.0590	0.0548	0.0535	0.0550	0.0556	0.0558	0.0565	0.0555	0.0546	0.0499	0.0463	0.0442	0.0371	0.0296	0.0212	0.0212	0.0203	0.0095	0.0095
10	0.0773	0.0765	0.0743	0.0776	0.0768	0.0767	0.0825	0.0801	0.0775	0.0709	0.0606	0.0520	0.0381	0.0296	0.0212	0.0212	0.0203	0.0095	0.0095
20	0.1065	0.1052	0.1059	0.1045	0.1045	0.1043	0.1021	0.1045	0.1031	0.0838	0.0623	0.0520	0.0381	0.0296	0.0212	0.0212	0.0203	0.0095	0.0095
50	0.1320	0.1313	0.1302	0.1339	0.1343	0.1417	0.1284	0.1158	0.1085	0.0838	0.0623	0.0520	0.0381	0.0296	0.0212	0.0212	0.0203	0.0095	0.0095

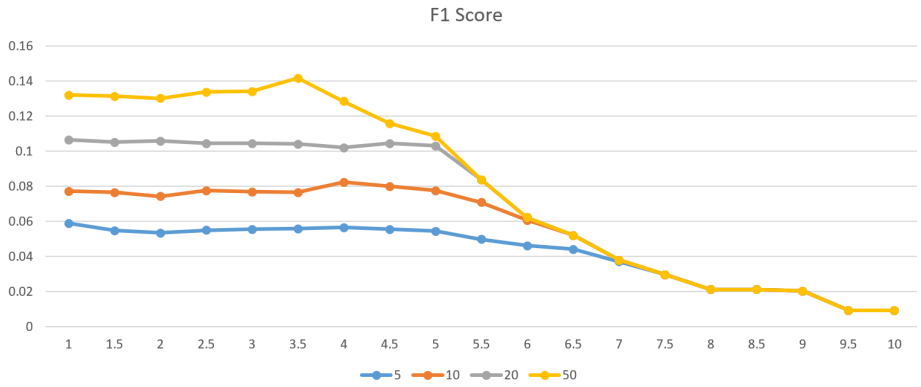

Fig. 12: F1 score of SPM purchased with various support

As we can see in these charts, by increasing minimum support, precision is increased, but the recall is decreased. As a result, F1 score increases slightly in the begging but drops sharply by increasing support from 5 to 10 percent.

On the other hand, selecting best confidence have a big effect on accuracy of recommendation system. in this section we have increased confidence from 0 to 100 percent with steps of 10 percent. Precision (Table 69 and Figure 13) , recall (Table 70

and Figure 14) and f1 score (Table 71 and Figure 15) with various confidence values are displayed here. All these test are done with minimum support of 4 percent.

Table 69: Precision of sequential rules recommendation with minimum support =4% and various confidences

	0	10	20	30	40	50	60	70	80	90	100
5	0.2331	0.2312	0.2377	0.2645	0.3163	0.2727	0.0000	0.0000	0.0000	0.0000	0.0000
10	0.1954	0.1954	0.2091	0.2522	0.2940	0.2727	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.1571	0.1571	0.1707	0.2334	0.2886	0.2727	0.0000	0.0000	0.0000	0.0000	0.0000
50	0.1491	0.1491	0.1658	0.2305	0.2886	0.2727	0.0000	0.0000	0.0000	0.0000	0.0000

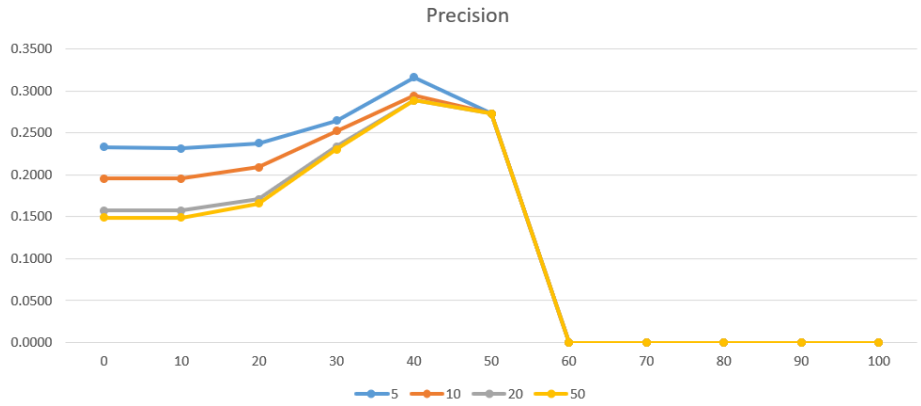

Fig. 13: Precision of sequential rules recommendation with various confidence

Table 70: Recall of sequential rules recommendation with minimum support =4% and various confidence

	0	10	20	30	40	50	60	70	80	90	100
5	0.0321	0.0319	0.0321	0.0327	0.0263	0.0064	0.0000	0.0000	0.0000	0.0000	0.0000
10	0.0523	0.0523	0.0526	0.0526	0.0350	0.0064	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.0757	0.0757	0.0759	0.0717	0.0377	0.0064	0.0000	0.0000	0.0000	0.0000	0.0000
50	0.1128	0.1128	0.1046	0.0746	0.0377	0.0064	0.0000	0.0000	0.0000	0.0000	0.0000

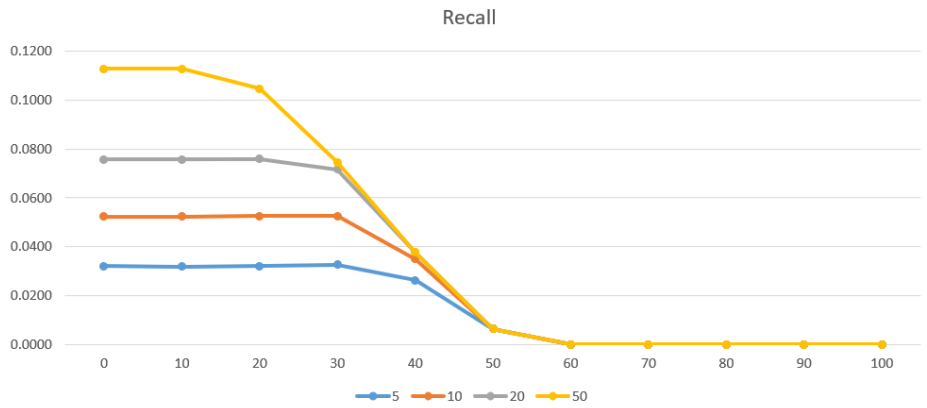

Fig. 14: Recall of sequential rules recommendation with various confidence

Table 71: F1 score of sequential rules recommendation with minimum support =4% and various confidence

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
5	0.0565	0.0560	0.0566	0.0581	0.0485	0.0125	0.0000	0.0000	0.0000	0.0000	0.0000
10	0.0825	0.0825	0.0840	0.0870	0.0626	0.0125	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.1021	0.1021	0.1051	0.1097	0.0667	0.0125	0.0000	0.0000	0.0000	0.0000	0.0000
50	0.1284	0.1284	0.1283	0.1127	0.0667	0.0125	0.0000	0.0000	0.0000	0.0000	0.0000

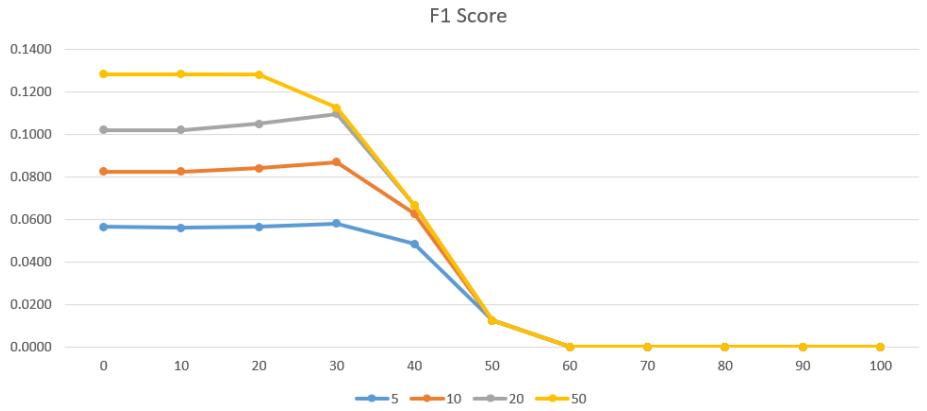

Fig. 15: F1 score of sequential rules recommendation with various confidence

By increasing confidence from 0 to 50 percent, precision increases but with higher minimum confidences, precision would be zero. From the f1 score result, we can conclude that for recommending 20 items, confidence =30 percent generate the highest accuracy.

4.3.2 Best Support and Confidence

In training phase (section 3.3.3), we find a combination of minimum support and minimum confidence which resulted to the highest accuracy. Table 72 shows the best minimum support and confidence for sequential rule mining of input dataset. As we can see the best values also depended on number of recommended items. For example, for recommending 20 items, support=0.5% and confidence=30% generate the highest accuracy, but for recommending 10 items, support=0.5% and confidence=30% have leads to the highest f1 score.

Table 72: Best minimum support and confidence values of sequential rule mining for recommending 20 items

Support	**Confidence**	**F1Score Rec 5**	**F1Score Rec 10**	**F1Score Rec 20**	**F1Score Rec 50**
0.5	30	0.0691	0.0949	0.1350	0.1543
0.5	40	0.0750	0.1031	0.1344	0.1537
1	30	0.0641	0.0892	0.1279	0.1559
2.5	30	0.0656	0.0912	0.1267	0.1563
2	30	0.0630	0.0877	0.1262	0.1606
3	30	0.0630	0.0890	0.1251	0.1490
1.5	30	0.0638	0.0894	0.1251	0.1587
0.5	50	0.0558	0.0855	0.1230	0.1506
3.5	30	0.0606	0.0922	0.1221	0.1372

4.3.3 Collaborative Filtering Recommendations Evaluation

In collaborative filtering, we recommend items based on ratings of similar users. The number of similar users is an important parameters in performance of collaborative filtering recommendations. Collaborative filtering recommended items are a combination of purchase frequency of items by users and items recommended based on

ratings of similar users. In this section we have remove the items that user has already bought. Then we analyses precision (Table 73 and Figure 16) , recall (Table 74 and Figure 17) and f1 score (Table 75 and Figure 18) of remaining items in collaborative filtering with various number of similar users.

Table 73: Precision of collaborative filtering recommendation without purchased items by various number of similar users

	0.25	**0.5**	**0.75**	**1**	**1.25**	**1.5**	**1.75**	**2**
5	0.0087	0.0052	0.0035	0.0017	0.0017	0.0004	0.0004	0.0004
10	0.0061	0.0026	0.0043	0.0017	0.0017	0.0012	0.0012	0.0012
20	0.0078	0.0070	0.0061	0.0043	0.0043	0.0029	0.0026	0.0031
50	0.0101	0.0090	0.0082	0.0073	0.0064	0.0069	0.0066	0.0060

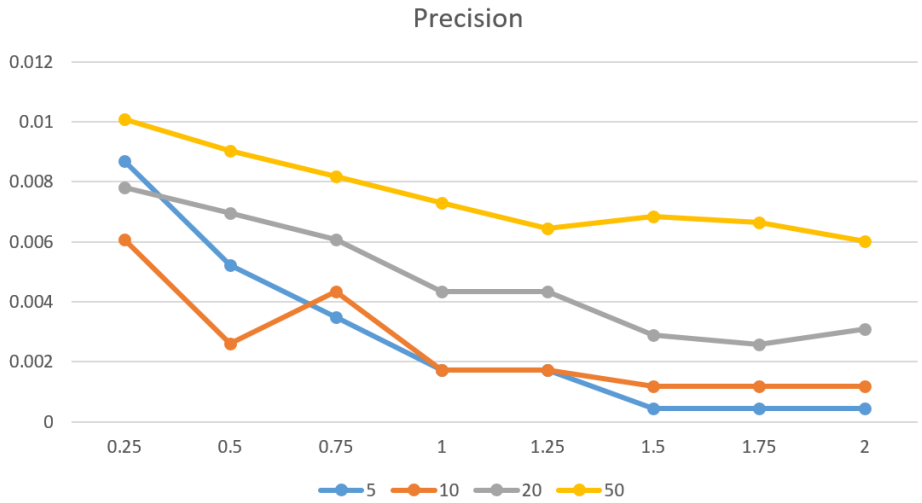

Fig. 16: Precision of collaborative filtering recommendation without purchased items by various number of similar users

Table 74: Recall of collaborative filtering recommendation without purchased items by various number of similar users

	0.25	**0.5**	**0.75**	**1**	**1.25**	**1.5**	**1.75**	**2**
5	0.0013	0.0008	0.0005	0.0003	0.0003	0.0004	0.0003	0.0003
10	0.0019	0.0008	0.0013	0.0005	0.0005	0.0012	0.0008	0.0008
20	0.0048	0.0042	0.0037	0.0027	0.0027	0.0029	0.0021	0.0024
50	0.0154	0.0138	0.0125	0.0111	0.0098	0.0069	0.0082	0.0074

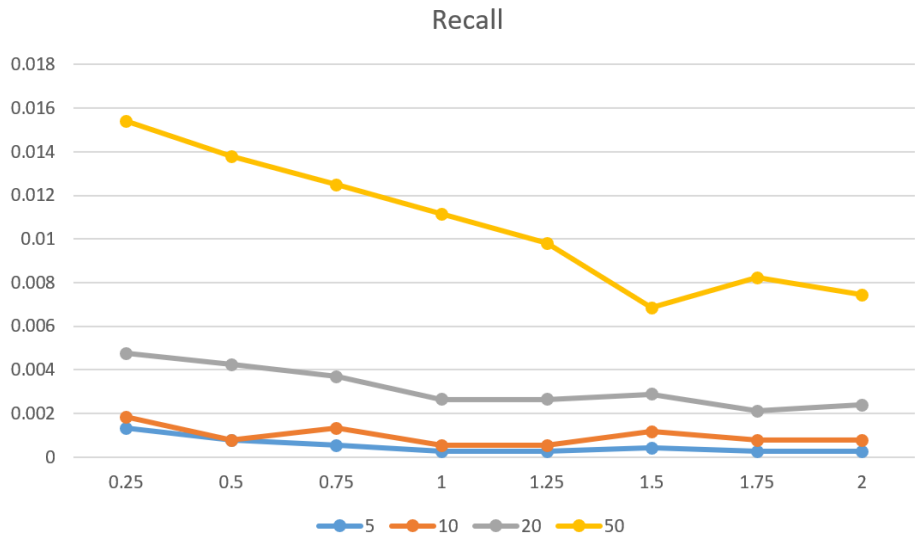

Fig. 17: Recall of collaborative filtering recommendation without purchased items by various number of similar users

Table 75: F1 score of collaborative filtering recommendation without purchased items by various number of similar users

	0.25	**0.5**	**0.75**	**1**	**1.25**	**1.5**	**1.75**	**2**
5	0.0023	0.0014	0.0009	0.0005	0.0005	0.0005	0.0005	0.0005
10	0.0028	0.0012	0.0020	0.0008	0.0008	0.0012	0.0012	0.0012
20	0.0059	0.0053	0.0046	0.0033	0.0033	0.0030	0.0026	0.0030
50	0.0122	0.0109	0.0099	0.0088	0.0078	0.0067	0.0065	0.0059

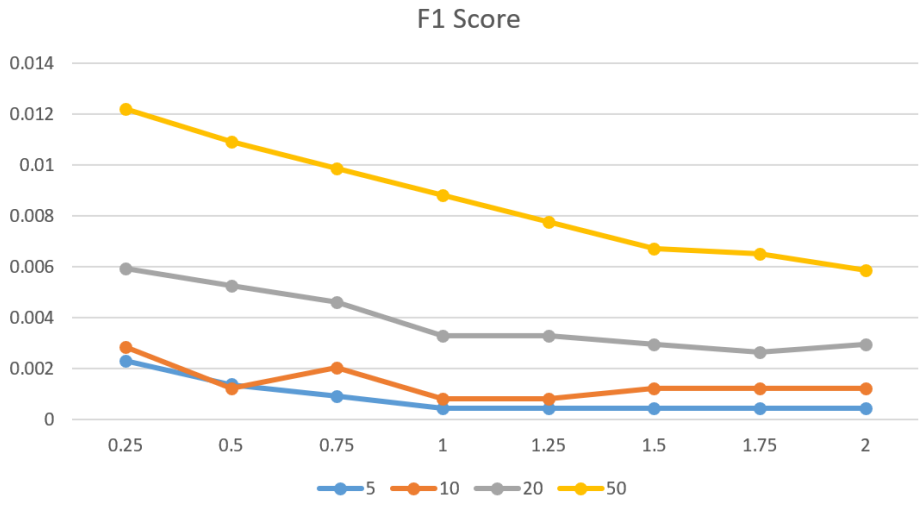

Fig. 18: F1score of collaborative filtering recommendation without purchased items by various number of similar users

4.3.4 Best Discovered Weights for Intermediate Matrices

As it is explained in chapter 3, in training phase, we trains recommendation system on training dataset and test it over verification dataset to find best weights for each user item matrix. Table 76 shows the weights which have the highest f1-score for recommending 10 items. These discovered weights will be used in recommendation phase for each intermediate user item matrix.

Table 76: Best weights for each user item matrix

Purchase frequency	CF not purchased	SPM purchased	SPM not purchased	F1 score Rec5	F1 score Rec 10	F1 score Rec 20	F1 score Rec 20
95	5	60	10	0.0697	0.1147	0.1525	0.1695
95	10	60	10	0.0697	0.1147	0.1525	0.1695
95	5	60	5	0.0697	0.1147	0.1525	0.1692
95	15	60	10	0.0697	0.1147	0.1522	0.1694
95	15	60	15	0.0697	0.1147	0.1522	0.1694
95	20	60	15	0.0697	0.1147	0.1522	0.1694
95	5	60	15	0.0697	0.1147	0.1522	0.1692
95	10	60	15	0.0697	0.1147	0.1522	0.1692
85	20	55	20	0.0708	0.1147	0.1520	0.1683

4.3.5 Compare Accuracy of Recommendation Methods

In this section we analyze precision, recall and F1-score of all discuses recommendation system and compare it with our proposed ensemble e-commerce recommendation system.

Table 77: Comparing precision of recommendation systems

	SPM np	SPM p	SPM	CF np	PF	CF	HOPE	SEERS
5	0.0893	0.2976	0.2013	0.0118	0.2805	0.0478	0.2013	0.3296
10	0.0723	0.2682	0.1857	0.0099	0.2635	0.0352	0.1849	0.2843
20	0.0580	0.2447	0.1649	0.0099	0.2265	0.0390	0.1648	0.2418
50	0.0422	0.2058	0.1396	0.0093	0.1830	0.0308	0.1106	0.1929

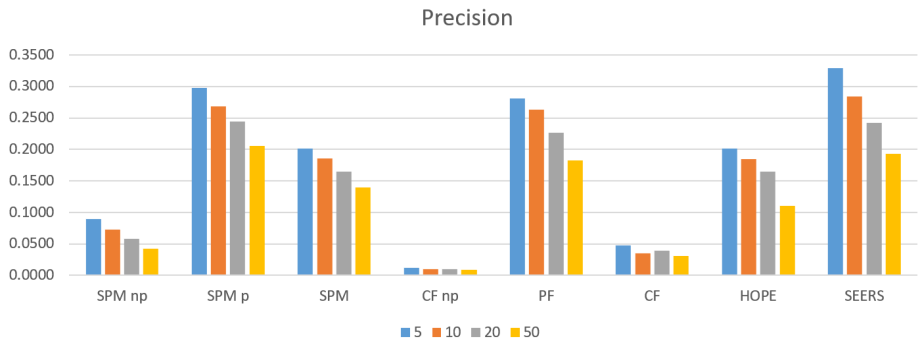

Fig. 19: Comparing precision of recommendation systems

Table 78: Comparing recall of recommendation systems

	SPM np	SPM p	SPM	CF np	PF	CF	HOPE	SEERS
5	0.0140	0.0458	0.0314	0.0018	0.0438	0.0075	0.0314	0.0515
10	0.0226	0.0796	0.0580	0.0029	0.0814	0.0110	0.0578	0.0888
20	0.0362	0.1340	0.1008	0.0059	0.1338	0.0244	0.1010	0.1511
50	0.0656	0.2181	0.1877	0.0140	0.2283	0.0481	0.1605	0.2407

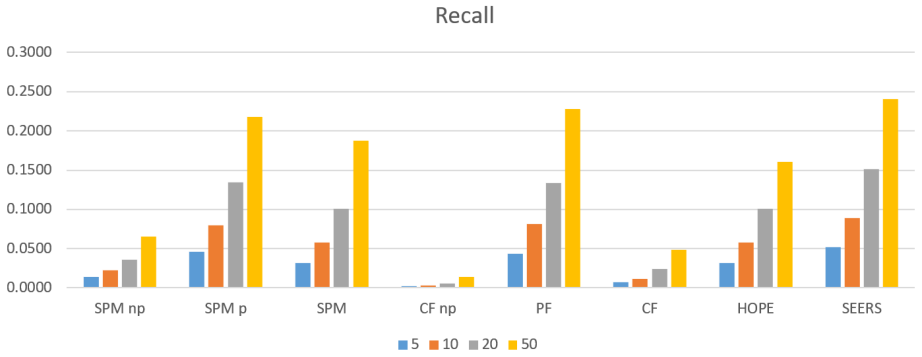

Fig. 20: Comparing recall of recommendation systems

Table 79: Comparing F1 score of recommendation systems

	SPM np	SPM p	SPM	CF np	PF	CF	HOPE	SEERS
5	0.0241	0.0794	0.0544	0.0031	0.0758	0.0129	0.0544	0.0891
10	0.0344	0.1227	0.0883	0.0045	0.1243	0.0168	0.0880	0.1353
20	0.0445	0.1732	0.1251	0.0074	0.1683	0.0300	0.1252	0.1860
50	0.0514	0.2118	0.1601	0.0112	0.2032	0.0376	0.1309	0.2142

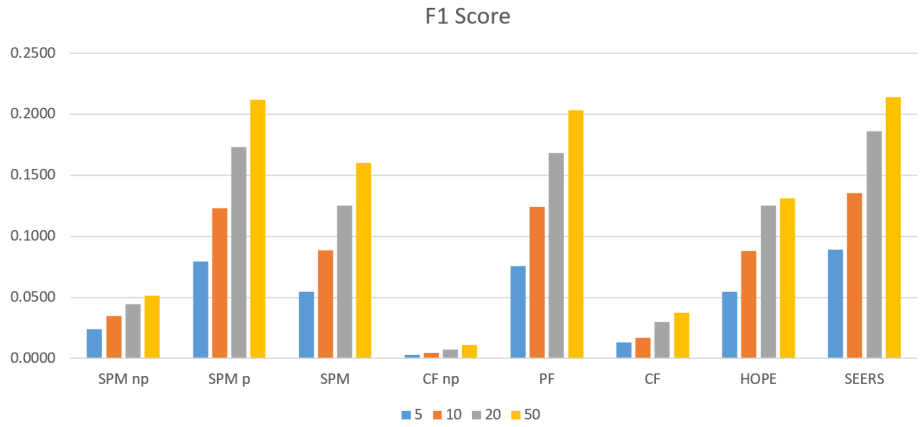

Fig. 21: Comparing F1 score of recommendation systems

In these diagrams to create SPM recommendations, we have used sequential rules with minimum support of 0.5 percent and minimum confidence of 40 percent. For generating CF recommendations, 0.25 percent of similar neighbors are used. As it is explained in chapter 3, CF is consist of purchased frequency and ratings calculated by collaborative filtering. These diagrams show that precision, recall, and F1 score of purchase frequency (PF) is much higher than CF-not-purchased recommendations. Also, we can see the same situation in SPM: SPM-purchased items are generating recommendations with higher accuracy compare to SPM of items which the user has never purchased. HOPE system gives 90 percent weight to SPM and 10 percent to CF. As we can see, this method's accuracy is better than collaborative filtering and very similar to SPM method. On the other hand, SEERS separate CF to two user item matrices which are purchase frequency and CF not purchased. Also, divides SPM to SPM purchased and SPM not purchased. Then gives each one the discovered best weights in the training phase. As a result, SEERS recommend items with higher accuracy compare to HOPE system.

CHAPTER 5

Conclution and Future Work

Buying or selling products or services online is called e-commerce. E-commerce recommendation system allows customers to find products more conveniently and also helps sellers to sell more products. Collaborative filtering is the most widely used method in recommendation systems. It uses a matrix of user item rating for recommending the next item. But usually, this matrix is very sparse because, for many user items, there is not any purchase or rating information. On the other hand, sequential pattern mining can be used to extract customers purchase pattern effectively. This method's limitation is that it can not recommend items that did not appear in frequent sequential patterns. Hybrid recommendation systems have been developed to overcome, or at least, mitigate the limitations of collaborative filtering and sequential pattern mining recommendation systems.

Existing e-commerce recommendation system, which we referred to in this paper are liu2009hybrid, HOPE, and HPCRec. liu2009hybrid first, segment customers based on frequency, recency, and monetary. Then in each segment, cluster transaction into groups based on similar products, then by using sequential pattern mining predict transaction cluster of customers in the next period and return items in that cluster as recommendation candidates. Also, by using collaborative filtering in each cluster, top items are recommended. Finally, combine two method candidates linearly. HOPE algorithm first runs collaborative filtering and sequential pattern mining separately to recommend items. Then integrate them by giving a weight to each one of the methods. HPCRec first uses the historical purchase to create the normalized user item rating matrix. Then uses similarity between click streams patterns to fill up

more rating in use item matrix and finally find unknown items by using collaborative filtering. Limitation of these methods is that in collaborative filtering, they give the same weight to the rating of the user (extracted from the purchase frequency of users) and other ratings calculated based on similar users. But our experimental results show that user rating is more important, and in our method, we give it more weight. Also, they give static weight to collaborative filtering and sequential pattern mining recommendations, but we use stacking ensemble learning for discovering the best weight for each dataset.

In this thesis, a novel Stacking Ensemble E-commerce Recommendation System (SEERS) has been proposed. This method consists of two phases: training and recommendation. In the training phase, it finds the best support and confidence values for sequential pattern mining and best number of similar users in collaborative filtering. Also, in the training phase, it finds the best weights for each intermediate user item matrices. In the recommendation phase, this method, first, calculate the rating of items based on similar users by using collaborative filtering method. Then uses sequential rule mining method to predict next purchase and creat two user item matrices for purchased and not purchased items. Also based on a historical purchase, it creates a purchased frequency user item matrix. Finally, integrate these for intermediate matrices by adding weights. Experimental results discovered weights in the training phase. Experimental results show that our proposed method predicts the next purchase of customers better than existing related methods. It has better precision, recall, and F1 score compares to HOPE system.

5.1 Future Work

In this research, a combination of collaborative filtering and sequential rule mining methods are used to recommend best products as e-commerce recommendation system output. But we can also use Recurrent Neural Network (RNN), and it's extensions such as Long Short Term Memory(LSTM) to create recommendation systems and integrate its results with collaborative filtering and sequential pattern mining.

The benefit of RNN and LSTM over traditional recommendation method is simplicity, adaptability, and scalability. In this method, we can create a model for recommendation system, and then we train it over historical purchase data. As a result, we do not need to analyze internal patterns of data in e-commerce database (Simplicity). There are different types of products and customers in each e-commerce dataset. For each dataset, we are training our model only on that dataset. As a result, the model completely adapts itself to each specific dataset (Adaptability). In traditional recommendation systems, after adding different categories of products or users to the e-commerce dataset, we need to analyze the system again. But by using RNN and LSTM, new nodes will be added, and the model will optimize itself automatically in the training phase (Scalability).

In a traditional neural network, we assume that inputs and outputs are independent of each other. But in recommendation systems or sequential pattern mining, the next output is depended on previous data. Recurrent Neural Network (RNN) uses sequential information to predict the next item. RNNs are called recurrent because they perform the same task for every element of the input sequence, with the output being depended on the previous computations. Long Short Term Memory (LSTM) is an extension of RNN. LSTM consists of four neural networks. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell, and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. As a result, it is very flexible for predicting next sequences such as products in e-commerce recommendations systems. The LSTM recommendation system's outputs are recommended items, and inputs are users, products, and historical purchase data. But we need to design and tailor an LSTM model for ecommerce recommendation to find the best number of layers, the best number of nodes in each layer, and fining the weights by training the model. Finally, we can use stacking ensemble learning method to integrate the output of LSTM with the four intermediate user item matrices, which we found in chapter 3 by using sequential pattern mining and collaborative filtering.

References

Aggarwal, C. C. (2016). *Recommender systems.* Springer.

Aggarwal, C. C., Bhuiyan, M. A., & Al Hasan, M. (2014). Frequent pattern mining algorithms: A survey. In *Frequent pattern mining* (pp. 19–64). Springer.

Aggarwal, C. C., Procopiuc, C., & Yu, P. S. (2002). Finding localized associations in market basket data. *IEEE Transactions on Knowledge and Data Engineering*, *14*(1), 51–62.

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Data engineering, 1995. proceedings of the eleventh international conference on* (pp. 3–14).

Ahmed, C. F., Tanbeer, S. K., Jeong, B.-S., & Lee, Y.-K. (2009). Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, *21*(12), 1708–1721.

Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002). Sequential pattern mining using a bitmap representation. In *Proceedings of the eighth acm sigkdd international conference on knowledge discovery and data mining* (pp. 429–435).

Bao, X. (2009). Applying machine learning for prediction, recommendation, and integration.

Bao, X., Bergman, L., & Thompson, R. (2009). Stacking recommendation engines with additional meta-features. In *Proceedings of the third acm conference on recommender systems* (pp. 109–116).

Bhatta, R., Ezeife, C., & Butt, M. N. (2019). Mining sequential patterns of historical purchases for e-commerce recommendation. In *International conference on big data analytics and knowledge discovery* (pp. 57–72).

Chen, L., & Su, Q. (2013). Discovering user's interest at e-commerce site using clickstream data. In *Service systems and service management (icsssm), 2013 10th international conference on* (pp. 124–129).

Cho, Y. B., Cho, Y. H., & Kim, S. H. (2005). Mining changes in customer buying behavior for collaborative recommendations. *Expert Systems with Applications*, *28*(2), 359–369.

Choi, K., Yoo, D., Kim, G., & Suh, Y. (2012). A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *Electronic Commerce Research and Applications*, *11*(4), 309–317.

Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from `http://archive.ics.uci.edu/ml`

Ezeife, C., Lu, Y., & Liu, Y. (2005). Plwap sequential mining: open source code. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations* (pp. 26–35).

Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., & Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, *1*(1), 54–77.

Fournier-Viger, P., Wu, C.-W., Zida, S., & Tseng, V. S. (2014). Fhm: Faster high-utility itemset mining using estimated utility co-occurrence pruning. In *International symposium on methodologies for intelligent systems* (pp. 83–92).

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., & Hsu, M.-C. (2000). Freespan: frequent pattern-projected sequential pattern mining. In *Proceedings of the sixth acm sigkdd international conference on knowledge discovery and data mining* (pp. 355–359).

Huang, C.-L., & Huang, W.-L. (2009). Handling sequential pattern decay: Developing a two-stage collaborative recommender system. *Electronic Commerce Research and Applications*, *8*(3), 117–129.

Kim, Y. S., & Yum, B.-J. (2011). Recommender system based on click stream data using association rule mining. *Expert Systems with Applications*, *38*(10), 13320–13327.

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

Liu, M., & Qu, J. (2012). Mining high utility itemsets without candidate generation. In *Proceedings of the 21st acm international conference on information and knowledge management* (pp. 55–64).

Liu, Y., Liao, W.-k., & Choudhary, A. (2005). A two-phase algorithm for fast discovery of high utility itemsets. In *Pacific-asia conference on knowledge discovery and data mining* (pp. 689–695).

Lu, H. (2014). Recommendations based on purchase patterns. *International Journal of Machine Learning and Computing*, *4*(6), 501.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, *23*, 187–192.

Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., ... Hsu, M.-C. (2004). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on knowledge and data engineering*, *16*(11), 1424–1440.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1–35). Springer.

Saini, S., Saumya, S., & Singh, J. P. (2017). Sequential purchase recommendation system for e-commerce sites. In *Ifip international conference on computer information systems and industrial management* (pp. 366–375).

Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *International conference on extending database technology* (pp. 1–17).

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), 1–13.

Tseng, V. S., Wu, C.-W., Fournier-Viger, P., & Philip, S. Y. (2015). Efficient algorithms for mining the concise and lossless representation of high utility itemsets. *IEEE transactions on knowledge and data engineering*, *27*(3), 726–739.

Wolpert, D. H. (1992). Stacked generalization. *Neural networks*, *5*(2), 241–259.

Xiao, Y., & Ezeife, C. (2018). E-commerce product recommendation using historical purchases and clickstream data. In *International conference on big data analytics and knowledge discovery* (pp. 70–82).

Yang, Z., & Kitsuregawa, M. (2005). Lapin-spam: An improved algorithm for mining sequential pattern. In *21st international conference on data engineering workshops (icdew'05)* (pp. 1222–1222).

Yap, G.-E., Li, X.-L., & Philip, S. Y. (2012). Effective next-items recommendation via personalized sequential pattern mining. In *International conference on database systems for advanced applications* (pp. 48–64).

Yin, J., Zheng, Z., & Cao, L. (2012). Uspan: an efficient algorithm for mining high utility sequential patterns. In *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining* (pp. 660–668).

Yin, J., Zheng, Z., Cao, L., Song, Y., & Wei, W. (2013). Efficiently mining top-k high utility sequential patterns. In *Data mining (icdm), 2013 ieee 13th international conference on* (pp. 1259–1264).

Yun, U. (2008). A new framework for detecting weighted sequential patterns in large sequence databases. *Knowledge-Based Systems*, *21*(2), 110–122.

Yun, U., & Leggett, J. J. (2006). Wspan: Weighted sequential pattern mining in large sequence databases. In *Intelligent systems, 2006 3rd international ieee conference on* (pp. 512–517).

Zaki, M. J. (2001). Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, *42*(1-2), 31–60.

Zhao, G., Lee, M. L., & Wynne, H. (2014). Utilizing purchase intervals in latent clusters for product recommendation. In *Proceedings of the 8th workshop on social network mining and analysis* (p. 4).

Zheng, L., Cui, S., Yue, D., & Zhao, X. (2010). User interest modeling based on browsing behavior. In *Advanced computer theory and engineering (icacte), 2010 3rd international conference on* (Vol. 5, pp. V5–455).

Zida, S., Fournier-Viger, P., Wu, C.-W., Lin, J. C.-W., & Tseng, V. S. (2015). Efficient mining of high-utility sequential rules. In *International workshop on machine learning and data mining in pattern recognition* (pp. 157–171).

Vita Auctoris

NAME:	Mehdi Naseri
PLACE OF BIRTH:	Shiraz, Iran
YEAR OF BIRTH:	1982
EDUCATION:	Islamic Azad University of Shiraz, B.Sc., Computer Software Engineering, Shiraz, Iran, 2005
	Islamic Azad University of Najafabad, M.Sc., Computer Software Engineering, Najafabad, Iran, 2010
	University of Windsor, M.Sc., Computer Science, Windsor, Ontario, 2019