The HSPRec E-Commerce System Open Source Code Implementation

Christie. I. Ezeife
School of Computer Science,
University of Windsor
cezeife@uwindsor.ca

Mahreen Nasir
School of Computer Science,
University of Windsor
nasir11d@uwindsor.ca

Ritu Chaturvedi
School of Computer Science,
University of Guelph
chaturvr@uoguelph.ca

Angel Veliz Castro
School of Computer Science,
University of Windsor
velizcaa@uwindsor.ca

2021 IEEE/ACIS 20th International Fall Conference on Computer and Information Science (ICIS Fall) | 978-1-7281-7679-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICISFALL51598.2021.9627424

[1] ***Abstract*—To promote big data application access, usage and deployment, this paper presents a downloadable open source code implementation for an E-Commerce Recommendation system, HSPRec (Historical Sequential Pattern Recommendation System), in JAVA. The HSPRec system is composed of six different modules for generating purchase/click sequential databases, mining sequential patterns, computing click purchase similarities, generating purchase sequential rules, computing weights for frequent purchase patterns through Weighted Frequent Purchase Pattern Miner, and normalization of the user-item ratings to predict level of interest. The source code of each module and the main runner are discussed under four possible headings of running environment, input data files and format, minimum support format, output data files and format. The overall goal of the HSPRec system is to improve E-commerce Recommendation accuracy by incorporating more complex sequential patterns of user purchase and click stream behavior learned through frequent sequential purchase patterns. HSPRec provides more accurate recommendations than the tested comparative systems.**

***Index Terms*—E-commerce recommendation, Sequential Pattern Mining, Source code, application deployment, Collaborative Filtering, user-item matrix quality**

I. Introduction

An important application of recommendation system is in the E-commerce domain [11], where the implicit ratings derived from historical purchase and/or clickstream data are used as rating of items in the user-item rating matrix [8], [9]. Many users may not be ready to provide explicit ratings for many items and there is a large set of continuously growing items (products), a very small percentage of which, each user may have purchased. In addition, users' purchase behavior change with time so that sequences of clicks and purchases play important role in capturing more realistic users' purchase behavior. Thus, the HSPRec (Historical Sequential Pattern Recognition) [2] is a recent E-commerce recommendation system that integrates sequential patterns of purchases generated from historical and/or clickstream E-commerce data into the user-item rating matrix to capture more complex purchase behavior.

Collaborative filtering is one of the most widely used recommendation techniques. It accepts a user-item rating matrix (R), having ratings of each item *i* by user (u_i) denoted as r_{u_i}. The goal of collaborative filtering is to predict r_{u_i}, a rating of user u on item i which may be unknown, by going through the following four major steps [1]:(1) Compute the mean rating for each user u_j using all of their rated items, (2) calcula the similarity between a target user v and all other users u Similarity can be computed with Cosine Similarity (v, u_j) Pearson Correlation coefficient [1] function, (3) find simil users of target user v as their Top-N users and (4) Predi rating for target user v for item i using only ratings of v Top-N peer group.

HSPRec algorithm [2] learns more complex sequential patter of user historical purchase behavior which it includes in tl user-item matrix to make it quantitatively and qualitative rich before applying collaborative filtering. Other existi recommendation systems that the HSPRec in its previo experimental analysis were compared to, are the Choi12R system [5] and the HPCRec18 system [13]. For purposes important application deployment, access and reusability, th paper presents new discussions of the implementations of tl HSPRec E-Commerce recommendation system [2] with tl E-Commerce datasets used, which are cleaned and process by modules of the HSPRec with a url access to downlo the source code of the system. A detailed explanation a implementation of the six modules are presented with example in section II.

A. Related Work

Work on E-Commerce recommendation system includ the Choi12Rec system [5], the HPCRec18 system [13], tl HSPRec system [2] which uses sequential pattern mini algorithms such as the GSP [12]. Other related systems wi open source implementation and deployment of proposed a gorithms are available in paper [7] in 2005 SigKDD worksh on data mining open source code system implementations. summary of some of these related systems are provided ne

The GSP (Generalized Sequential Patterns) [12] algorith is an Apriori based sequential pattern mining algorithm th mines frequent sequential patterns from a sequential databas SDB (such as sequence of sets of products purchased by ea user every day), given a minimum support percentage s (e. 0.2), and a set of candidate 1-items (such as store produc like milk, sugar). The result of the GSP algorithm is to find frequently purchased sequence sets that are purchased in qua tities greater or equal to the minimum support number of tim such as <(milk, egg)(bread, butter)> to indicate that whenev users purchased milk and egg in one day, it is learned that th purchase bread and butter next in the following day. Table shows an example of daily purchase sequential database. mine frequent sequential patterns using a sequential patte mining algorithm such as the GSP (Generalized Sequenti Pattern Mining) [12] are available.

[1]*This research was supported by the Natural Science and Engineering Research Council (NSERC) of Canada under an Operating grant (OGP-0194134) and a University of Windsor grant

SID	Click Sequence
1	<(1,2,3), (7,5,3), (1,6), (6), (1,5)>
2	<(1,4), (6,3), (1,2), (1,2,5,6)>
3	<(1,5), (6,5,2), (6), (5)>
4	<(2,7), (6,6,7)>
5	<(1,5)>

TABLE I
DAILY CLICK SEQUENTIAL DATABASE

SID	Purchase Sequence
1	<(1,2), (3), (6), (7), (5)>
2	<(1,4),(3),(2), (1,2,5,6)>
3	<(1),(2),(6), (5)>
4	<(2),(6,7)>

TABLE II
DAILY PURCHASE SEQUENTIAL DATABASE

A fragment of E-commerce historical database data is presented in Table III with schema {Uid, Click, Clickstart, Clickend, Purchase, Purchasetime} for attributes representing User identity, Clicked items, Clickstart and Clickend. Furthermore, purchase contains list of items purchased by the user and Purchasetime represents timestamp when the purchase happened.

Uid	Click	clickstart	clickend	Purchase	Purchasetime
1	1,2,3	2014-04-04 11:25:14	2014-04-04 11:45:19	1, 2	2014-04-04 11:30:11
1	7,5,3	2014-04-05 15:30:07	2014-04-05 15:59:36	3	2014-04-05 15:56:32
1	1,6	2014-04-06 4:10:01	2014-04-06 04:30:29	6	2014-04-06 4:18:26
1	6	2014-04-07 8:50:29	2014-04-07 9:50:07	7	2014-04-07 8:59:21
1	1,5	2014-04-08 14:10:24	2014-04-08 14:25:18	5	2014-04-08 14:19:55
2	1, 4	2014-04-13 4:01:11	2014-04-13 4:30:15	1, 4	2014-04-13 04:04:34
2	6,3	2014-04-15 9:30:34	2014-04-15 9:40:11	3	2014-04-15 09:34:37
2	1,2	2014-04-17 13:40:11	2014-04-17 13:59:11	2	2014-04-17 13:54:48
2	1,2,5, 6	2014-04-17 11:30:18	2014-04-17 11:50:19	1, 2,5,6	2014-04-17 11:44:55
3	1, 5	2014-04-20 09:40:45	2014-04-20 10:10:15	1	2014-04-20 10:02:53
3	6, 5,2	2014-04-21 11:59:59	2014-04-21 12:10:39	2	2014-04-21 12:07:15
3	6	2014-04-22 17:05:19	2014-04-22 17:30:06	6	2014-04-22 17:10:28
3	5	2014-04-23 11:00:05	2014-04-23 11:20:15	5	2014-04-23 11:06:37
4	2,7	2014-04-23 12:00:11	2014-04-23 12:30:10	2	2014-04-23 12:06:37
4	6,6,7	2014-04-26 9:45:11	2014-04-26 10:20:13	6, 7	2014-04-26 10:06:37
5	1,5	2014-04-27 16:30:25	2014-04-27 16:45:45	?	

TABLE III
HISTORICAL E-COMMERCE DATA SHOWING CLICKS AND PURCHASES

The term consequential bond [13] is used to measure the relationship between clicks and purchases made by a user in the historical databases. They are derived as sequential pattern rules mined from the click sequence database. For example the consequential bond for user 5 in Table 4 is computed by finding frequent click sequential patterns using the SPR (click sequence database) call. From the generated sequential patterns, only strong rules containing the sequences in user 5 clicks were filtered out in order to derive user 5's possible future purchases.

Summary of the HSPRec Algorithm. The HSPRec takes a minimum support count, historical user-item purchase frequency matrix and consequential bond as input to generate rich user-item matrix as output. Although a summary of the main algorithm is provided in this paper, the full algorithm can be obtained through [2]. It consists of six main modules or functions, each of which could be run separately, plugged in or out, updated to create revised versions of systems for mining E-Commerce database systems.

B. *Problem Definition*

Given the HSPRec sequential pattern based E-commerce recommendation system algorithm [2], which takes as its input data historical click and purchase data over a certain period of time, and produces as its outputs, (1) the frequent periodic (daily, weekly, monthly) sequential purchase and click patterns in the first stage, (2) then, in the second stage, generates qualitatively (specifying level of interest or value for already rated items) and quantitatively (finding possible rating for previously unknown ratings) rich user item matrix based on the sequential purchase and click patterns obtained in the first stage. Collaborative Filtering (CF) is then applied to the enriched matrix to improve the overall accuracy of recommendation. The main contribution of this paper is to present, deploy and make available the source code implementation of the HSPRec E-Commerce recommendation system in Java for future use by other researchers and further improvement on the code with re-implementation in other programming languages.

C. *Contributions and Paper Outline*

A limitation of existing related systems such as (HPCRec18 [13], HSPRec [2]) is that there are no easily available and usable source codes for applications to build commercial systems, for experimental analysis and further incremental improvement or re-development in different programming languages. Thus, in this paper, we propose an Open source Java code for an E-Commerce recommendation system called Historical Sequential Pattern Recommendation (HSPRec), where a link for downloading working source JAVA codes and sample datasets for the HSPRec is included as part of this paper. The rest of the paper is organized as follows. Section 2 discusses example recommendation with the HSPRec E-Commerce system. Section 3 discusses the JAVA implementations of the HSPRec algorithm for E-Commerce recommendation, Section 4 discusses experimental analysis, while section 5 presents conclusions and future work.

II. EXAMPLE STEPS IN RUNNING THE HSPREC SYSTEM

Although the details of the main algorithm of HSPRec system are adapted from a previous work done by the same research group [..], the main steps are briefly explained with an example for convenience:

1. Convert historical purchase information (Table III) to user-item purchase frequency matrix (Table IV) by counting the number of each item purchased by each user. For example, User 2 purchased items 1 and 2 twice but other items once.
2. Create daily purchase sequential database (Table II) of customer purchase (Table III) by running sequential historical periodic database generation module (SHOD) with the customer purchase database as input. For example, User 2 daily purchase sequence is <(1, 4), (3), (2), (1, 2, 5, 6)>, which shows User 2 purchased item 1 and item 4 together on the same day and purchased item 3 on the next day then purchased item 2 on another day and finally purchased items 1, 2, 5 and 6 together on the next day.
3. Input daily purchase sequential database (Table II) to the Sequential Pattern Rule (SPR) module to generate sequential rule from frequent purchase sequences. For example, 1-frequent purchase sequences={ <(1)>, <(2)>, <(3)>, <(5)>, <(6)>, <(7)>}. Similarly, some of the 2-frequent purchase sequences= <(6), (5)>, <(3), (6)>, <(3), (5)>, <(2), (7)>, <(2),(6)>, <(2),(5)> and some of the 3-frequent purchase sequences=< (2), (6), (5)>, <(1), (6), (5)>, <(1), (3), (6)>, <(1), (3), (5)>, <(1), (2), (6)>. Thus, some of the possible sequential purchase pattern rules based on frequent purchase sequences are: (a) 1, 5 $\rightarrow$ 3 (b) 2, 6 $\rightarrow$ 1 (c) 2, 6 $\rightarrow$ 5, where rule (a) states that if user purchases items 1 and 5 together then user will purchase item 3 in next purchase, which will be applied in case of for user 3 in user-item purchase frequency matrix (Table IV).

User/item	1	2	3	4	5	6	7
User1	1	1	1	**?**	1	1	1
User2	2	2	1	1	1	1	**?**
User3	1	1	**?**	**?**	1	1	**?**
User4	**?**	1	**?**	**?**	**?**	1	1
User5	?	?	?	?	?	?	?

TABLE IV
USER-ITEM PURCHASE FREQUENCY MATRIX (M)

4. Apply purchase sequential rule in user-item purchase frequency matrix to improve quantity of ratings.
5. For each user, where click happened without a purchase such as user 5 in Table II, we use consequential bond between clicks and purchases derived as sequential pattern rules mined from the click sequence database. The consequential bond for user 5 in Table 4 is computed by finding frequent click sequential patterns using the SPR(click sequence database) call. From the generated sequential patterns, we filter out only strong rules containing the sequences in user 5 clicks so that they can be used to derive user 5 is possible future purchases.

User/item	1	2	3	4	5	6	7
User1	1	1	1	?	1	1	1
User2	2	2	1	1	1	1	?
User3	1	1	1	?	1	1	?
User4	1	1	?	?	1	1	1
User5	?	?	?	?	?	?	?

TABLE V
ENHANCED USER-ITEM FREQUENCY MATRIX WITH SEQUENTIAL PATTERN

UID	Click sequence	Purchase sequences
1	<(1,2,3), (7,5,3), (1,6), (6), (1,5)>	<(1,2), (3), (6), (7), (5)>
2	<(1,4), (6,3), (1,2), (1,2,5,6)>	<(1,4),(3),(2), (1,2,5,6)>
3	<(1,5), (6,5,2), (6), (5)>	<(1),(2),(6), (5)>
4	<(2,7), (6,6,7)>	<(2),(6,7)>
5	<(1,5)>	?

TABLE VI
CONSEQUENTIAL BOND TABLE SHOWING CLICKS AND PURCHASES

6. Next, compute Click and Purchase Similarity (click sequence, purchase sequence) using longest common subsequence rate (LCSR) and frequency similarity (FS) equation 3 presented in section 2 description of module.
7. Assign Click Purchase Similarity (click sequence, purchase sequence) value to purchase patterns present in consequential bond (Table II) to create weighted purchase patterns (Table VII).
8. Input weighted purchase patterns to Weighted Frequent Purchase Pattern Miner (WFPPM) presented in section 2 to calculate the weight for each frequent individual item based on its occurrence in weighted purchase patterns.
9. Repeat the steps 5, 6, 7, and 8 if there are more users with clicks but without purchases, otherwise assign computed items weights to modify enhanced user-item frequency matrix (Table V) and apply collaborative filtering. The summary of the six modules of the HSPRec algorithm is provided next.

1) **Historical Periodic Sequential Database Generation (SHOD) Modul**e: In our case, the SHOD algorithm created the daily purchase sequential database presented in Table II from purchase database and the same steps can be repeated to generate daily click sequential database by using click item database as input.
2) **Frequent sequences generation**: Generates frequent sequences using Generalized Sequential Pattern Mining (GSP) algorithm [12].
3) **Click Purchase Similarity (CPS):** To compute the CPS similarity between click sequence and purchase sequence of each user, we have used sequence similarity and frequency similarity of the two sequences. **Sequence similarity**: It is based on longest common subsequence rate (LCSR) [4] and presented in Equation (1).

$$LCSR(X,Y) = \frac{LCS(X,Y)}{max(X,Y)} \qquad (1)$$

In our case, X represents click sequence and Y represents purchase sequence and LCS is defined as:

$$LCS(Xi,Yj) = \begin{cases} \phi \quad if \quad i=0 \quad or \quad j=0 \\ LCS(X_{i-1},Y_{j-1}) \cap X_i \quad if \quad x_i = y_i \\ long(LCS(X_i,Y_{j-1}), \\ \qquad LCS(X_{i-1},Y_j)), \quad if \quad x_i \neq y_i \end{cases}$$

Frequency similarity: First, form distinct sets of items from both click and purchase sequential patterns and count the number of items occurring in each sequence

Purchase	CPS
<(1,2),(3),(6),(7),(5)>	0.624
<(1,4),(3),(2),(1,2,5,6)>	0.834
<(1),(2),(6), (5)>	0.636
<(2),(6,7)>	0.67
<(1),(3)>	0.5

TABLE VII
WEIGHTED PURCHASED PATTERNS

to form the vectors specifying the number of times a user clicked or purchased a particular item. Then, apply Cosine frequency similarity (Equation (2)) to the click and purchase vectors.

$$Cosine(X,Y) = \frac{X_1 * Y_1 + X_2 * Y_2 + ...X_n * Y_n}{\sqrt{{X_1}^2 + ... + {X_n}^2}\sqrt{{Y_1}^2 + ... + {Y_n}^2}} \quad (2)$$

Thus,

$$CPS = \alpha * LCSR(X,Y) + \beta * FS(X,Y) \quad (3)$$

where $\alpha + \beta = 1$, and $0 < \alpha < \beta < 1$, and α and β are weights assigned to reflect the importance of the two sequences of similarity and frequency. This CPS(X,Y) can be used as weight or probability that user u will purchase the entire sequence as shown in Table VII.

4) **Sequential Pattern Rule (SPR) module:** Sequential Pattern Rule (SPR) is based on the use of frequent sequential patterns created from periodic sequential database. Thus, input of SPR is periodic historical sequential database and output is recommended possible purchase items derived from generated sequential patterns rules. The major steps in SPR are: **Rule generation**: Represent frequent sequences in the form of $U_{\text{click}} \rightarrow U_{\text{purchase}}$, where the left-hand side of the rule refers to clicked item set while the right-hand side is the recommended item to be purchased. Furthermore, to verify the validity of SPR, confidence of SPR is defined as (note that conf is for confidence):

$$Conf(U_{\text{click}} \rightarrow U_{\text{purchase}}) = \frac{Support(U_{\text{click}} \cup U_{\text{purchase}})}{Support(U_{\text{click}})} \quad (4)$$

Here, some of the rules from frequent clicks sequences are: (a) (1,5) $\rightarrow$ (1),(3), (b)(1,5) $\rightarrow$ (3), (1), (c)(1,2) $\rightarrow$ (1,6), (d)(1)(5) $\rightarrow$ (6),(5) Assume we are only interested in rules that satisfy the following criteria: 1) At least two antecedents. 2) Confidence $>$ 50 percent. 3) Select one rule with highest confidence. Let us consider rule (a) recommends User 5 to purchase item 1 and item 3 when item 1 and item 5 are clicked together.

5) **Weighted Frequent Purchase Pattern Miner (WF-PPM) Module**: Weighted Frequent Purchase Pattern Miner (WFPPM) takes weighted purchase pattern as input (present in Table VII) and weighted purchase patterns is created by assigning CPS (click sequence, purchase sequence) value to purchase patterns in consequential bond Table (II) to generate frequent purchase patterns with weight under user specified minimum threshold as output. Major steps of WFPPM are: Count support of items: Count occurrence of items presented in weighted purchase pattern (Table VII). For example, {support(1):5, support(2):5, support(3):3, support(4):1, support(5):3, support(6):4, support(7):2} and **Calculate weight for individual item**: Compute weight of individual item from weighted purchase pattern (Table VII) using CPS module (Equation (3)).

$$R, item_i = \frac{\sum_{i=1}^{n} CPS \in item_i}{Support(item_i)} \quad (5)$$

For example, $R_1 = {}^{0.624+0.834+0.834+0.636+0.5}/_{5} =$ 0.68 Similarly, R_2=0.71, R_3=0.65, R_4=0.834, R_5=0.698, R_6=0.691, R_7=0.647, Next, **Test item weight with minimum threshold**: Define minimum threshold rating for user, here in our case, minimum threshold=0.2. So, all rating of item are frequent.

6) **User-item Matrix Normalization** Normalization in recommendation system helps to predict the level of interest of user on a particular purchased item. Thus, normalization function for a user u's rating of an item i (r_{ui}) takes user-item frequency matrix (Table VIII) as input and provides the level of user interest between 0 and 1 using unit vector formula (Equation (6)).

$$Normalization(r_{ui}) = \frac{r_{ui}}{\sqrt{{r_{ui1}}^2 + {r_{ui2}}^2 + ... + {r_{uin}}^2}} \quad (6)$$

The normalization of enhanced user-item matrix (Table VIII) using equation 5 is presented in Table IX.

User/item	1	2	3	4	5	6	7
User1	1	1	1	?	1	1	1
User2	2	2	1	1	1	1	?
User3	1	1	1	?	1	1	?
User4	1	1	?	?	1	1	1
User5	0.68	0.71	0.65	0.834	0.698	0.691	0.647

TABLE VIII
ENHANCED USER-ITEM PURCHASE FREQUENCY MATRIX WITH RATING FOR USER 5 (M_1)

User/item	1	2	3	4	5	6	7
User1	0.40	0.40	0.40	?	0.40	0.40	0.40
User2	0.57	0.57	0.28	0.28	0.28	0.28	?
User3	0.44	0.44	0.44	?	0.44	0.44	?
User4	0.44	0.44	?	?	0.44	0.44	0.44
User5	0.68	0.71	0.65	0.834	0.698	0.691	0.647

TABLE IX
NORMALIZED USER-ITEM PURCHASE FREQUENCY MATRIX (M_2)

III. PROPOSED JAVA IMPLEMENTATIONS OF THE HSPREC

The proposed deployment of the HSPRec system delivers the JAVA source codes for all components of the system from the url https://cezeife.myweb.cs.uwindsor.ca/codes.html. The codes are accompanied by some sample test data including the (1) data used in the paper as example historical purchase and click data and shown through Tables in sections 1 and 2 of this paper; and (2) a more general dataset of Yoochoose published in ACM RecSys 2015 [3], which is similar to the Amazon dataset. Yoochoose dataset has both click stream and purchase, therefore making it more adaptable. Next, we will present details regarding the execution and running environment of the HSPRec system. Figure. 1 presents the HSPRec system workflow.

A. Program Execution Environment

(a) Using an Integrated Development Environment (IDE): To compile and run any module of HSPRec (e.g., HSPRec) in an IDE such as Eclipse or IntelliJ, import the HSPRec module java project from within any IDE.

(b) Using Command Line: For easier portability and access, to run the program in Unix Command-Line, change directory to the folder where the SHOD module is located. Compile the program by using 'javac filename.java' and run it by using 'java filename.

B. HSPRec Program Execution

(a) Execution of HSPRec as a Standalone System

The HSPRec Mainrunner is the main program that will launch each module in sequence instead of manually running each module individually.

(1) Running Environment: Same as described in section 3.1, (2) Input Files and Format: The input data required to run the HSPRec system are ClickData.txt and PurchaseData.txt as shown in Figure 1 representing historical click and purchase data. Each line of input data from the ClickData.txt file includes UserID, ProductId, Rating, and Timestamp (when the click happened). It produces sequential databases as output, (3) Minimum Support Format: The program needs to accept a numeric value between 0 and 1 (e.g., 0.5 for 50%) as a minimum support or the minimum number of records in the input sequential database, This program also accept a numeric value for the number of lines to read (that is, number of records in the sequential database). The user also needs to adjust the wfppmInput7 value, which adjusts the number of people who only clicked and did not purchase that added to the matrix. Change this value depending on your need or value found from the intermediate user-item matrix. The lines as to where you can make these changes are listed in the internal documentation. (4) Output Data Format and Files: Once the HSPRec system terminates, it would have produced fourteen new files. These are the files produced by each module. The files produced are the following: clicks_hsc_intermediate.txt, EditedClickData.txt, hsc_daily_sequential_db_.txt, hsp_daily_sequential_db.txt, purchase_hsp_intermediate.txt, user_itemFrequency_matrix.txt, frequent_sequence.txt, cps_results.txt, cps_uniqueitems.txt, sequenceRules.txt, user_item_martix_updated.txt, user_item_matrix_WFPPM_results.txt, WFPPM_results.txt and user_item_matrix_normalized.txt.

(b) Execution of HSPRec as Individual Components

The HSPRec system is composed of six different modules as presented in Figure. 1 consisting of the (1) SHOD sequential historical database generator algorithm, (2) GSP sequential pattern miner module, (3) CPS click purchase similarity finder module for computing the correlation between purchased and clicked data using consequential bond, (4) SPR, the sequential rule generator module, (5) WFPPM Weighted Frequent Purchase Pattern Miner Module, and the (6) The user-item rating matrix normalization module for normalizing the ratings in the enhanced matrix; and the (7) HSPRec Mainrunner, which is the main function that will launch each of the six modules in sequence. The source code of each module and the main runner are discussed under four possible headings of (1) Running environment, (2) Input files and format, (3) Minimum support format, (4) Output files and format. Each module is documented with information on this section for code readability. The modules are saved as .java files. On Unix/Linux system, each of these modules can be compiled separately using the command: javac filename.java. After compilation, the executable code of the module is stored in its filename.class so that to run/execute the module, use the Unix command: java filename.

1.The SHOD (sequential historical database) Module

(1)Running Environment : Same as described in section 3.1, (2) Input Files and Format: The input files for the SHOD Module are ClickData.txt and PurchaseData.txt. The two files can be composed of hundreds of thousands of lines where each line represents a click or purchase of an item by a user. Each line of input data from the ClickData.txt file includes UserID, ProductId, Rating, and Timestamp (when the click happened). The date attribute is in the format (mm dd, yyyy), (3) Minimum Support Format: It does not use a minimum support value, (4) Output Data Format and Files: Once the SHOD module terminates, it will produce six different files: clicks_hsc_intermidate.txt, EditedClickData.txt, hsc_daily_sequential_db_.txt, hsp_daily_sequential_db.txt, purchase_hsp_intermediate.txt, user_itemFrequency_matrix.txt. The data produced are periodic sequential databases as output.

2. The GSP (Generalized Sequential Pattern Miner) Module

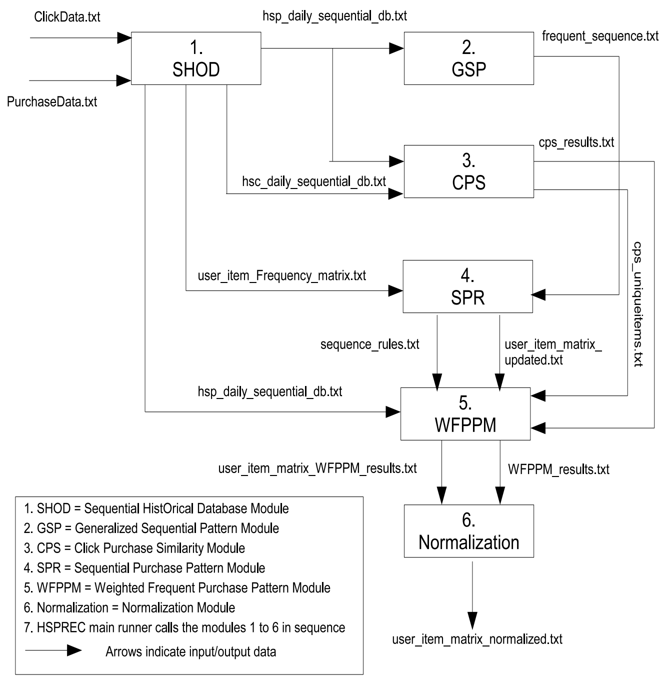
(1) Running Environment: Same as described in the SHOD Module, (2) Input Files and Format: The SHOD module produces the sequential database input data used for this module which is the hsp_daily_sequential_db.txt. It shows what each user purchased over time; for example, when the numbers are grouped, they are bought on the same day. An example of an input line is: '1,2','3','6','7','5' showing items 1 and 2 are bought on the same day and item 3 on a different day, (3) Minimum Support Format: The program accepts a numeric value between 0 and 1 as minimum support representing the percentage of the number of records (lines) in the input file that an itemset sequence such as '1', or '1, 2' needs to occur in the input data file to be considered frequent. This program also accepts a value for the number of lines (records) to read in the input sequential database file, (4) Output Data Format and Files: Once the GSP module terminates, it will produce one file: frequent_sequence.txt which contains repeating sequential patterns.

3. The CPS (Click Purchase Sequence Similarity) Module
(1) Running Environment: Same as described in the SHOD Module, (2) Input Files and Format: The SHOD module produces the CPS module's input data. The input files are hsc_daily_sequential_db_.txt and hsp_daily_sequential_db.txt. The hsp file shows what each user purchased over time; when the numbers are grouped, they were bought on the same day. An example of an input line is: '1,2','3','6','7','5' which shows that items 1 and 2 are bought on the same day and item 3 on a different day. The hsc file shows what each user clicks on overtime; when the numbers are grouped, it shows that those items were clicked on the same day. An example of an input line is: '1,2,3','7,5,3','1,6','6','1,5' which shows that the user clicked on items 1,2 and 3 in one day and 7,5 and 3 on another day, (3) Minimum Support Format: It does not have a minimum support value, (4) Output Data Format and Files: Once the CPS module terminates, it will produce two new files: cps_resuts.txt and cps_uniqueitems.txt.

4. The SPR Module
(1) Running Environment: Same as described in the SHOD Module, (2)Input Files and Format: The input data files used for the SPR module are user_itemFrequency_matrix.txt and frequent_sequence.txt produced by both the SHOD module and the GSP module respectively, (3) Minimum Support Format: It does not have a minimum support value, (4) Output Data Format and Files: Once the SPR module has terminated, it will produce two files: sequenceRules.txt, user_item_martix_updated.txt. The output of this module is recommended possible purchase items.

5. The WFPPM Module
(1) Running Environment: Same as described in the SHOD Module, (2) Input Files and Format: The input data files used for the WFPPM module are generated from the SHOD, CPS and SPR modules and are named as hsp_daily_sequential_db.txt, CPS module cps_resuts.txt, cps_uniqueitems.txt and user_item_martix_updated.txt respectively. This module will also need the ClickData.txt and PurhcaseData.txt, previously used in the SHOD module, (3) Minimum Support Format: The WFFPM does not have minimum support. However, the it has a different value the user must input into the source code that is the wfppmInput7 value, which adjusts the number of people who only clicked and did not purchase, added to the matrix. Change this value depending on your need, which can be read from the user-item rating matrix, (4) Output Data Format and Files: Once the WFPPM module terminates, it will produce two files: user_item_matrix_WFPPM_results.txt and WFPPM_results.txt.

6. The Normalization Module
(1) Running Environment: Same as described in the SHOD Module, (2) Input Files and Format: The WFPPM module produces the Normalization module's input data as file named user_item_matrix_WFPPM_results.txt, (3) Minimum Support Format: It does not have a minimum support value, (4) Output Data Format and Files: Once the Normalization module terminates, it will produce one file: user_item_matrix_normalized.txt.



ClickData.txt
PurchaseData.txt
hsp_daily_sequential_db.txt
1. SHOD
2. GSP
frequent_sequence.txt
3. CPS
cps_results.txt
hsc_daily_sequential_db.txt
cps_uniqueitems.txt
user_item_Frequency_matrix.txt
4. SPR
sequence_rules.txt
user_item_matrix_updated.txt
hsp_daily_sequential_db.txt
5. WFPPM
user_item_matrix_WFPPM_results.txt
WFPPM_results.txt
6. Normalization
user_item_matrix_normalized.txt
1. SHOD = Sequential HistOrical Database Module
2. GSP = Generalized Sequential Pattern Module
3. CPS = Click Purchase Similarity Module
4. SPR = Sequential Purchase Pattern Module
5. WFPPM = Weighted Frequent Purchase Pattern Module
6. Normalization = Normalization Module
7. HSPREC main runner calls the modules 1 to 6 in sequence
Arrows indicate input/output data

Fig. 1. Workflow of HSPRec Algorithm Modules

IV. Experimental Design and DataSet Selection

The performance of previously existing historical sequential pattern mining based recommendation system, HSPRec [2] was compared with two other existing recommendation systems of ChoiRec12 [5] and HPCRec18 [13] in user-based collaborative filtering. The results show that HSPRec [2] performs better using datasets similar to Amazon datasets. To perform experiments, we used data available from Amazon product data [10] and Yoochoose dataset [3]. The Yoochoose dataset represents six months activities of a big e-commerce businesses in Europe selling stuff such as garden tools, toys, clothes and electronics. The dataset provides a collection of sequences of click events; click sessions. For some of the sessions, there are also buying events. Data is pre-processed by the SHOD module to create sequential database that is mined for sequential patterns. Previous experimental results showed that [2] performed well against the tested compared systems using three different evaluation metrics (a) mean absolute error (MAE) (b) precision and (c) recall with LibRec [6] library of Java (https://www.librec.net/).

Figures 2 and 3 presents results of HSPRec experiments that used sequential rules to enhance user-item matrix quantitatively and found better results than previously done related work [5], [13]. To evaluate the performance of the recommendation system, we use different number of users and nearest neighbors using three different evaluation parameters (a) mean absolute error (MAE) (b) precision and (c) recall with https://www.librec.net/ LibRec[6] library available in Java.

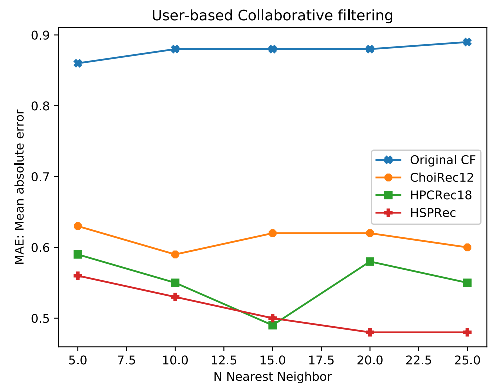


Fig. 2. User-based Collaborative Filtering with top-N

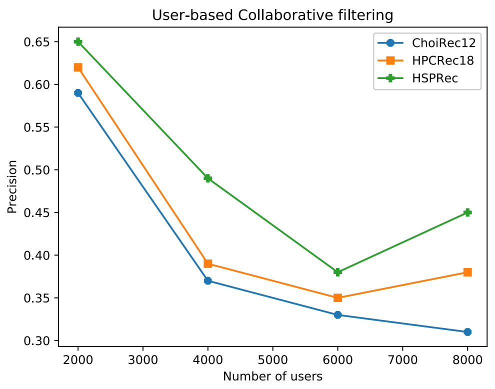


Fig. 3. Precision in User-based Collaborative Filtering

MAE measures the average of the errors in a set of predictions. It is the average over the test sample of the absolute differences between prediction and actual rating. Precision determines the fraction of relevant items (an item in which user is interested) retrieved out of all items in the recommendation system whereas recall determines the fraction of relevant items retrieved out of all relevant items in the recommendation system (note that results of recall are not shown here due to lack of space). The overall complexity of HSPRec is found to be $O(n^2)$.

V. Conclusions and Future Work

Many recommendation systems neglect sequential patterns during recommendation. Thus, to verify the necessity of sequential patterns in recommendation, we generated sequential patterns from historical E-commerce data and fed them into collaborative filtering to make user-item matrix rich from quantitative and qualitative perspectives. The main contribution of this paper is the Java source code implementation of a full recent efficient recent E-Commerce recommendation system, the HSPRec [2] that incorporates sequential patterns of user purchases and clicks into product recommendation. The source code of the system are downloadable for easy application deployment, use and research extension. Furthermore, implementation in other languages such as C++ and Python, separate modules of the HSPRec can be extended for further improvement to the accuracy of recommendation. An example extension is for handling high utility products for recommendation that also consider revenue to the retailer.

References

[1] Aggarwal, Charu C.: Recommender systems. Cham: Springer International Publishing, 2016.
[2] Bhatta Raj., Ezeife, C. I., Butt, Mahreen Nasir. (2019). Mining Sequential Pattern of Historical Purchases for E-Commerce Recommendation, in the proceedings of the 21st International Conference on Big Data Analytics and Knowledge Discovery - DaWaK 2019, Linz, Austria, Aug. 26 - 29, pages 57-72.
[3] Ben-Shimon, David., Tsikinovsky, Alexander., Friedmann, Michael., Shapira, Bracha., Rokach, Lior., and Hoerle, Johannes. (2015). Recsys challenge 2015 and the Yoochoose dataset. In Proceedings of the 9th ACM Conference on Recommender Systems, pages 357-358. ACM, 2015.
[4] Bergroth, L., Hakonen, H. and Raita, T. (2000). A survey of longest common subsequence algorithms. Seventh International Symposium on String Processing and Information Retrieval, pp. 39-48.
[5] 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.
[6] G. Guo, J. Zhang, Z. Sun, and N. Yorke-Smith. (2015). Librec: A java library for recommender systems. In Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd International Conference on User Modeling, Adaptation and Personalization.
[7] Ezeife, C. I., Lu, Yi. and Liu, Yi. (2005). PLWAP Sequential Mining: Open Source Code paper. Proceedings of the Open Source Data Mining Workshop on Frequent Pattern Mining Implementations, in conjunction with ACM SIGKDD, Chicago, IL, U.S.A., 2005.
[8] Kim, Y.S., Yum, B.J., Song, J. and Kim, S.M. (2005). Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. Expert Systems with Applications **28**(2), 381-393.
[9] Liu DR, Lai CH, Lee WJ. (2009). A hybrid of sequential rules and collaborative filtering for product recommendation. Information Sciences **217**(20), 3505-3519.
[10] McAuley,J. (2019). Amazon product data,http://jmcauley.ucsd.edu/data/amazon/.
[11] Schafer, J.B., Konstan, J.A. and Riedl, J. (2001). E-commerce recommendation applications. Data mining and knowledge discovery, 5(1-2), pp 115-153 (2001).
[12] Srikant, R. and Agrawal, R.: Mining sequential patterns. (1996). Generalizations and performance improvements. In International Conference on Extending Database Technology, pp. 1-17. Springer, Berlin, Heidelberg.
[13] Xiao Y, Ezeife C.I. (2018). E-Commerce Product Recommendation Using Historical Purchases and Clickstream Data. International Conference on Big Data Analytics and Knowledge Discovery, DAWAK, pp. 70-82. Springer LNCS.