E-Commerce Product Recommendation Using Historical Purchases and Clickstream Data

Ying Xiao and C. I. Ezeife(✉)

School of Computer Science, University of Windsor,
401 Sunset Ave, Windsor, ON N9B 3P4, Canada
{xiao11q,cezeife}@uwindsor.ca

Abstract. In E-commerce, user-item rating matrices for collaborative filtering recommendation systems are usually binary and sparse, showing only whether or not a user has purchased an item previously. Clickstream data containing more customer behavior have been used to improve recommendations by some existing systems referred in this paper as Kim05Rec, Kim11Rec, and Chen13Rec, using decision tree, association rule mining and category-based interest measurements respectively. However, they do not integrate valuable information from historical purchases and the consequential bond information between session-based clicks and purchases. This paper proposes Historical Purchase with Clickstream recommendation system (HPCRec), which normalizes the historical purchase frequency matrix to improve rating quality, and mines the session-based consequential bond between clicks and purchases to generate potential ratings to improve the rating quantity. Experimental results show HPCRec outperforms these existing methods, and is also capable of handling infrequent user cases, whereas other methods can not.

Keywords: E-commerce recommendation system
Collaborative filtering · CF · Clickstream history
Weighted frequent item · Data mining

1 Introduction

Recommendation systems provide suggestions of products to users, such as what items to buy, what music to listen to, or what online news to read [12]. Collaborative filtering (CF) is the most popular method used for recommendations, where a user's preferences can be associated with the preferences of a similar user community. In E-commerce, CF usually takes a binary user-item rating matrix (e.g., Table 1) as input, where "1" indicates that a user has purchased a

C. I. Ezeife—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.

product before, and "?" means a user has not. Given an incomplete user-rating matrix R of m users for n items with missing ratings (r_{uj}) of item j for user u, the user-based neighborhood model CF [1] would predict the unknown ratings (r_{uj}) through the following steps: (1) computing the mean rating for each user u; (2) calculating the similarity between target user u and all other users v; (3) computing user $u's$ peer group P; and then (4) predicting rating for target user u for item j.

The user-item rating matrix (eg., Table 1) in E-commerce usually contains part information of the historical transaction records (e.g., Table 2). In Table 2, each row records a purchase (a collection of item ids) that happened in a session and there may be multiple products for each purchase. There are some known fundamental issues with this approach as follows: (1) cold start: when new items or new users with unknown ratings/preferences appear in the database; (2) sparsity issue: When the known rating data takes only a very small proportion in the user-item rating matrix, (only a few hundreds of billions of products purchased by users) leading to confusing and compromised recommendations; (3) scalability issue: As the numbers of users and products grow rapidly, the time complexity and space complexity issues become more prominent.

Methods for enhancing recommendation input data have been studied in various papers, such as [4,8,9], since the user-item rating matrix in e-commerce only shows what items a user has purchased previously, which does not provide a lot of information about customer purchase history or item purchase history for the purposes of improving recommendation accuracy. In addition to the rating matrix, some other data sources such as clickstream data, meta data and transactions have been discovered and utilized to improve recommendations. Clickstream data have been used to predict a user's next request [5], discover patterns to build profile for customers [11], find the possibilities of purchasing items [14]. **Transaction data** (Table 2) is the detailed purchase history where each row records item purchases that occurred in a session by a specific user. **User-item rating matrix** (Table 1) is the basic input data format for CF recommendation systems. **Clickstream data** is the electronic record of a user's activity on the Internet [3]. In e-commerce, clickstream data reflects a user's Internet foot-

Table 1. A user-item rating matrix

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

Table 2. A historical transaction table

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

prints for behaviors such as clicks, basket placement, purchases, reading reviews and so on. Clickstream events may include attributes like SessionId, UserId, ItemId, Category, Time, visit duration, visit types and IP address. **Meta data** is the description of user preferences (such as product categories purchased (eg. automotive, beauty, electronics, software); and product details (eg., OS is iOS, item weight is 136 g, color is Black, etc.)). These information can be used in the recommendation engines for comparing products and recommending similar products.

1.1 Observations and Assumptions

Two assumptions are given in this section based on some observations.

Data Sparsity. CF method in E-commerce suffers from data sparsity of the rating matrix given the large number of products. It only uses the binary user-item rating purchase matrix (Table 1) which does not reflect much regarding: (1) how much a user likes an item; (2) how frequently or how long ago a user purchased an item; (3) what quantity of a product was purchased. This information is not integrated in the CF user-rating matrix but can potentially improve recommendations accuracy.

Information Distribution. Traditional CF systems only take purchase data into calculation, despite the fact that there are other data available for analysis. Moreover, from the data provided by ACM RecSys 2015 [2] in the flies yoochoose-clicks and yoochoose-buys. We can observe that the click data (yoochoose-clicks) is almost 27 times more than the purchase data (yoochoose-buys).

Consequential Relationship between Clicks and Purchases. Clickstream data and the purchase data (Table 2) can be re-organized to include these user click sequences as well as products purchased during each session as in Table 3. Sometimes there are no purchases but only clicks in a session. The relationship between clicks and purchases is called consequential relationship because clicks lead to certain purchases.

Assumption 1. Need to improve the rating quality. The binary user-item rating matrix in Table 1 is less informative compared to the original transaction records in Table 2 as it does not indicate how frequently an item is purchased. If there is a way to extract more information from the transaction records into the rating matrix to capture missing data, the recommendation system would perform better with the more informative input data. We first form a user-item purchase frequency matrix (Table 4) from Table 2, where each value represents the amount of a product purchased by a user. We then normalize the purchase frequency to a scaled value (0 to 1 in Table 5) representing how interested a user is in one item as compared to various others, the formula is introduced in Sect. 3.1.

Assumption 2. Need to improve the rating quantity. In the sessions with purchases where purchases were made by a user in Table 3, the interest of the user

Table 3. Consequential table

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

Table 4. User-item purchase frequency matrix

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

for the purchased products is affirmative, whereas for the sessions without a purchase, we cannot determine whether a user is interested in a product. However, the clicks in a session without a purchase may imply potential interest. We mainly discover the session-based consequential bond to generate more potential rating scores. For example, for session 6 where user 3 clicked products 3, 5 and 2 in Table 3, if we can find sessions sharing a similar click pattern but has purchased some product, then we can use the possibility to enrich the rating matrix such as Table 6 which is less sparse than the original table (Table 1). We assume by integrating the consequential bond information to the user-item purchase matrix, the accuracy of recommendations will be improved.

Table 5. Normalized user-item purchase frequency matrix

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

Table 6. Enriched and normalized user-item purchase matrix

Customer\Item	1	2	3	4
1	?	0.89	0.45	?
2	0.27	0.53	?	0.8
3	1	1	0.189	0.167

1.2 Paper Contributions

Studies in [3,10,13] have implied that there is a consequential relationship between behaviors collected as clickstream data and purchase data. This paper proposes the HPCRec system, which enriches the rating matrix from both quantity (converting some initial ratings of 0 to a consequential bond values) and quality (converting some initial ratings of 1 to a value between 0 and 1 that includes frequency of purchase of an item) aspects. The enriched matrix is then processed using the CF method. It takes the consequential table (eg., Table 3) and user-item purchase frequency matrix (eg., Table 4) as input, follows four main steps below to output a rating matrix with predicted ratings.

1. Normalizing user-item purchase frequency matrix to a new user-item rating matrix using unit vector formula with details in Sect. 3.1.
2. In the consequential table, for each session without a purchase belonging to a user, find the top-N similar sessions with purchases by comparing the click sequences using function CSSM (Clickstream Sequence Similarity Measurement) in Sect. 3.2. Then use the similarity as weight and assign to the purchases in the selected top-N session. This step generates a weighted transaction table where weights are similarities assigned to purchases.
3. Using the weighted transaction table from step 2, call function TWFI (Transaction- based Weighted Frequent Item) in Sect. 3.3 to get a list of items with purchasing possibilities.
4. For each item from the previous step, if the user from step 2 has not previously purchased the product, enrich the resulting matrix from step 1 with the possibility. Then, return to step 2 for the next session without a purchase if possible, and otherwise continue to step 5.
5. With the enriched rating matrix, run the CF algorithm and predict ratings. Return the rating matrix with predicted ratings.

The new feature contributions of paper include:

- **(i) Improving the quality of ratings** by capturing the level of interest in a product already purchased by a user before through record of normalized frequency of purchase using the method in [15].
- **(ii) Improving the quantity of ratings** with consequential bond between clicks and purchases, for the sessions without purchases.
- **(iii) Improving the recommendation accuracy** by processing the enriched rating matrix generated in the CF algorithm.
- **(iv) Making recommendations for infrequent users.** Our HPCRec system performs a session-based interest mining algorithm which finds the interests of the most similar sessions compared to the existing sessions without a purchase for the user.

1.3 Paper Outline

Section 2 shows some important research related to integrating clickstream data to make better recommendations; Sect. 3 proposes HPCRec system. Section 4 discusses experiments, and Sect. 5 presents conclusions and future work.

2 Related Work

Some important work have been done to integrate clickstream data into recommendation systems. A few popular related strategies is discussed in this section. noindent **A stage-based decision tree approach** [9]. Kim05Rec [9] first forms a decision tree for basket placement based on behaviors such as searching, browsing and click times. For each path in the tree, it gives the proportion of users taking that path and uses it as the probability to enhance the user-item matrix with these basket placement probability data before running CF algorithm to predict ratings. Kim05Rec [9] improves accuracy, but it shrinks the candidate range by only taking the products which have been previously paid attention to, only focuses on major cases by always choosing the popular path in the decision tree. **An association rule approach** [8]. Kim11Rec [8] integrated with association rule mining calculates the confidence between products in different stages including click, basket placement, and purchase. For instance, the clicks $(\langle abc \rangle, \langle bcd \rangle, \langle efa \rangle)$ that happened in the click stage for three different sessions, and there are some items such as a and b that a user has shown interest during current session, then the system finds the most relevant clicked item in (c, d, e, f) for a and b by comparing the lift score. Same for the basket placement and purchase stages. The next step is assigning weights to the scores of three different stages to calculate a final score. It was proven to outperform the decision tree approach, but by applying association rule mining, it loses the connection between users sharing special interests. **A category-based common interest approach** [4] Chen13Rec finds the similarity between two users on their clickstream sequences to address a better neighborhood with similar interest. It first uses the longest common subsequence to compare two click sequence groups of two users; the second indicator is the similarity between user-product click frequency vectors showing the click times of a user for all products; the third indicator is the similarity between user-product visiting duration vectors. By selecting top-N similar users using three indicators, the CF method can use it for neighbor selection and improve the poor relationship between users in the rating matrix.

3 Proposed HPCRec System

This paper proposes a novel recommendation system HPCRec (Algorithm 1) which integrates purchase frequencies and the consequential bond relationship between clicks and purchases. By processing this information, it enhances the user-item rating matrix in both quantity and quality aspects and then improves recommendations. We use pre-processed consequential table (Table 3) showing user click sequences with their purchases and frequency matrix (Table 4) showing number of times an item is pruchased as input, and HPCRec returns a matrix with predicted ratings (Table 7). There are three functions FN (Frequency Normalization), CSSM and TWFI used in HPCRec. HPCRec was also proven to give better recommendations to infrequent users. The HPCRec (Algorithm 1) is explained with example Tables 3 and 4 as input:

1. Normalize the purchase frequency in Table 4 for each user on each item, and get a normalized rating matrix by applying unit vector formula (Eq. 1) of Sect. 3.1 as Table 5;
2. For each session without a purchase, such as session 6 for user 3 in Table 3. Calculate the similarity between session 6 and other sessions with purchases (1,2,3,4,5) by comparing the clicks calling CSSM in

Algorithm 1. HPCRec System to Predict Ratings

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

Sect. 3.2, get CSSM($\langle 3,5,2\rangle, \langle 1,2\rangle$) = 0.37, CSSM($\langle 3,5,2\rangle, \langle 3,5,2,3\rangle$) = 0.845, CSSM($\langle 3,5,2\rangle, \langle 2,1,4\rangle$) = 0.33, CSSM($\langle 3,5,2\rangle, \langle 4,4,1,2\rangle$) = 0.245, CSSM($\langle 3,5,2\rangle, \langle 1,2,1\rangle$) = 0.295; form a weighted transaction table using the similarity as purchases and weight as transaction records such as $[\langle (2) : 0.37\rangle, \langle (2,3) : 0.845\rangle, \langle (1,2,4) : 0.33\rangle, \langle (2,4,4) : 0.245\rangle, \langle (1) : 0.295\rangle]$;

3. Call TWFI in Sect. 3.3 with the weighted transaction table from step 2, and get weighted frequent items (2:1, 3:0.189, 4:0.167); for all weighted frequent items, if the user has not purchased it, add the possibility into the normalized frequency matrix such as in Table 6.
4. Return to step 2 if there is more session without a purchase, otherwise, run the CF algorithm using the updated rating matrix (Table 6) to get predicted ratings for all of the original unknowns as demonstrated in Table 7, return the rating table with predicted ratings. Accuracy also can be calculated.

Table 7. User-item rating matrix with predicted ratings

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

3.1 FN: Frequency Normalization

In this module, we take the user-item purchase frequency (Table 4) as input, normalize the frequencies into numbers between 0 and 1 using the unit vector formula [15] (Eq. 1). For each user, $\langle x_1, x_2, x_3, \ldots, x_n \rangle$ is the purchase vector showing the purchase frequency of product $1, 2, 3, \ldots, n$ respectively. For user 2, the purchase vector is $\langle 1, 2, 0, 3 \rangle$, so the normalized purchase frequency for user 2 on item 2 is $\frac{2}{\sqrt{1^2+2^2+0^2+3^2}} = 0.53$. The normalized frequency matrix is in Table 5, from which we can see that for each user, the differences between ratings reflects the different levels of interest. We also tried the feature scaling normalization method (Eq. 3) to normalize the frequencies, but the unit vector formula was more effective.

$$x' = \frac{x}{\sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}}. \tag{1}$$

3.2 CSSM: Clickstream Sequence Similarity Measurement

Inspired by the idea of Chen in [4], we introduce CSSM (Clickstream sequence similarity measurement) which takes the frequency and position of items in sequences into consideration to calculate the similarity. Instead of calculating the category visiting sequences and frequencies, CSSM calculates product click sequences and frequencies. We explain this function in steps using two click sequences $\langle 3, 5, 2 \rangle$ and $\langle 3, 5, 2, 3 \rangle$ in Table 3 as an example.

1. Calculate the longest common subsequence rate $LCSR(x, y) = \frac{LCS(x,y)}{max(|x|,|y|)}$, where the longest common subsequence (LCS) [7] is defined in Eq. 2. e.g., $LCS(\langle 3,5,2 \rangle, \langle 3,5,2,3 \rangle) = 3$, the maximum sequence size is 4, so LCSR($\langle 3,5,2 \rangle, \langle 3,5,2,3 \rangle$) = 3/4;

$$LCS(X_i, Y_j) = \begin{cases} \emptyset & \text{if } i = 0 \text{ or } j = 0 \\ LCS(X_{i-1}, Y_{j-1}) \frown x_i & \text{if } x_i = y_j \\ longest(LCS(X_i, Y_{j-1}), LCS(X_{i-1}, Y_j)) & \text{if } x_i \neq y_j \end{cases} \tag{2}$$

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

3.3 TWFI: Transaction-Based Weighted Frequent Item

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

1. Calculate support. Form a distinct item set from all the transactions, and find the support for each item, e.g., $\langle 1:2, 2:4, 3:1, 4:3\rangle$;
2. Compute the average weighted support (AWS = AW × support) for each item using the same strategy in [16], where average weight (AW $= \frac{sum(weight)}{support}$), which makes AWS = sum(weight), e.g., AWS(4) $= 0.33 + 0.245 + 0.245 = 0.82$, $\langle 1:0.625, 2:1.79, 3:0.845, 4:0.82\rangle$; We also tried using maximum weighted support (MWS = max(weight) × support), AWS(4) = max(0.33, 0.245,0.245) = 0.33 × 3 = 0.99, the maximum approach was proven good, but the average approach is better.
3. Normalize weighted support using feature scaling (Eq. 3), so for the average weighted support, max = 1.79, min = 0.625, then the new average weighted support for item 3 is $^{(0.845-0.625)}/_{(1.79-0.625)} = 0.189$, all the weighted supports are $\langle 1:0, 2:1, 3:0.189, 4:0.167\rangle$;

$$x' = \frac{x - min}{max - min}. \tag{3}$$

4. Return all the items with normalized weighted support greater or equal to MinWeightedSupport, e.g., (2:1, 3:0.189, 4:0.167) for using average weighted support;

4 Evaluation and Comparative Analysis

We have implemented HPCRec and compared with some existing approaches, and it has been proven that HPCRec is capable of making recommendations for infrequent users which the existing systems can not. Experimental results show that HPCRec is more accurate which proves that the consequential bond with the normalized frequencies are more effective at predicting user interest.

4.1 An Example for Handling Infrequent User Cases

Here we use a small example (Tables 9 and 8) to explain that HPCRec can handle the infrequent case. There are 100 sessions in this example, user 100 intends to purchase "g" after reviewing "g" repeatedly for three times, but only HPCRec can predict that. The weak bond between infrequent users is ignored by other approaches. We find the most similar clicks sequence for "ggg" which is "fgh", the similarity between them maybe weak, but it is the strongest for "ggg", then we use the purchases in session 99 as recommendations, which successfully finds item "g" for user 100.

Table 8. Product table

ItemId	1	2	3	4	5	6	7	8
ItemName	a	b	c	d	e	f	g	h
Category	1	2	2	1	3	4	1	4

Table 9. Consequential table

SessionId	UserId	Clicks	Purchases
1	1	abcdade	ace
2	2	cdea	acdd
3	3	ddcabe	aed
4–98	4–98	abce	ace
99	99	fgh	fg
100	100	ggg	

4.2 Experimental Design

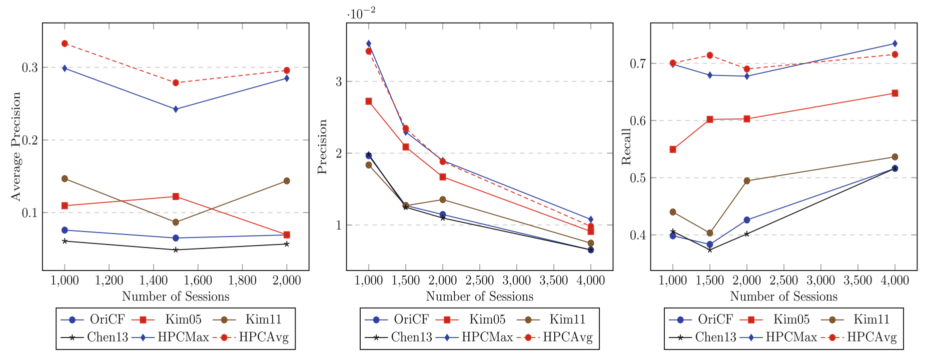
To make sure the evaluation is fair, we only select the top-N (N is productNumber/10) scores from different approaches and normalize the scores using Eq. 3 as the rating to keep the measuring standard consistent. Then we feed the new rating matrix to an evaluation method of an existing recommendation library Librec [6] to test all of the approaches. For Chen13Rec [4], we use the measured relationship to enhance the similarity table during the evaluation. For HPCRec, we ran through using both average weighted and maximum weighted support strategies in module TWFI (Sect. 3.3).

Dataset and Sample Selection: We use the dataset provided by YOOCHOOSE GmbH for ACM RecSys 2015 [2], which is from an online retailer in Europe. There are two files recording 33,040,175 clicks and 1,177,769 purchase events respectively, all of the events happened in 9,512,786 unique sessions, the total amount of product is 52,739 belonging to 339 categories. For sample selection, we randomly select a certain amount of session 10 times and use the average value. For each time, given the lack of user information in the dataset, we generate a reasonable number of user and assign to the sessions randomly then use the average value from 10 times attempts. It has been proven that regardless of how users are distributed in sessions, our methods are better.

Method: We evaluate with both user-based and item-based CF recommendations. Itemknn selections and pcc similarity method are used for item-based evaluation, userknn and cosine similarity method are used for user-based evaluation. We use evaluation measurements AP (Average Precision), Precision and Recall from Librec [6]. For the calculation details, please visit Librec.

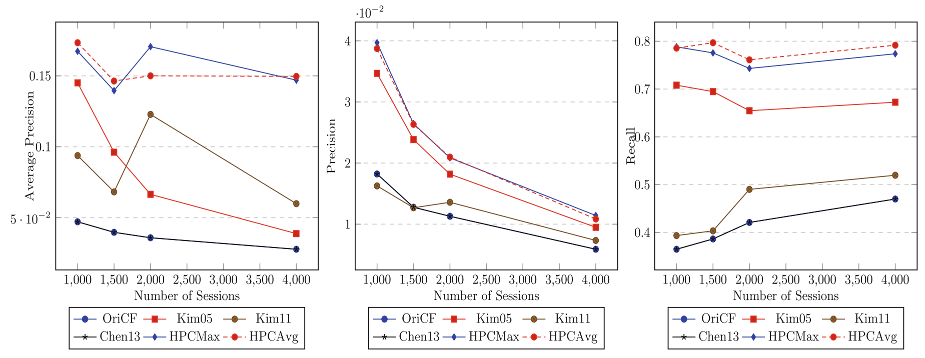
4.3 Experimental Results

From both user-based (Fig. 1) and item-based (Fig. 2) CF evaluation results on varying numbers of sessions, we can see the accuracy keeps dropping as the amount of sessions increases, our approaches are still better in this respect. For average accuracy and recall, our methods significantly beats others. We select a different number of top-N scores from all of the methods for calculation and evaluation. Both user-based CF (Fig. 3) and item-based CF (Fig. 4) are still the best which proves the high quality of our scores. Kim05Rec [9] also demonstrates good performance.



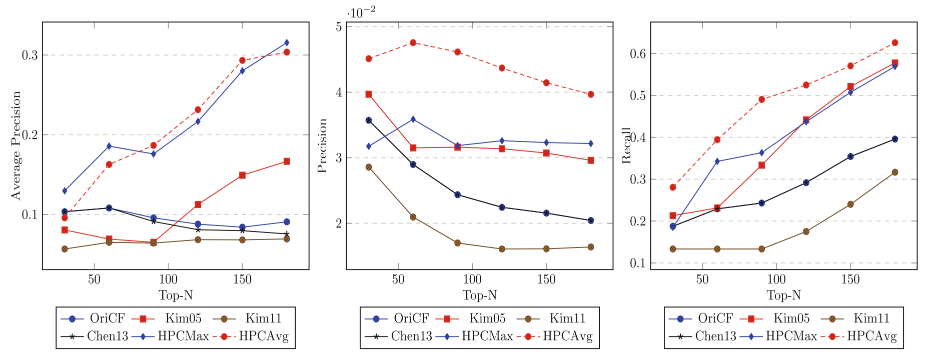
(a) Average Precision (b) Precision (c) Recall

Fig. 1. User-based CF Evaluation on different number of sessions



(a) Average Precision (b) Precision (c) Recall

Fig. 2. Item-based CF Evaluation on different number of sessions



(a) Average Precision (b) Precision (c) Recall

Fig. 3. User-based CF Evaluation on different number of top-N scores

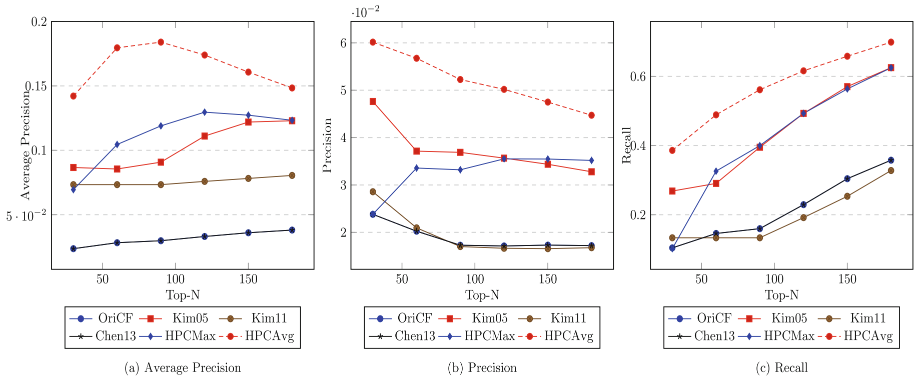


Fig. 4. Item-based CF Evaluation on different number of top-N scores

5 Conclusions and Future Work

To conclude, proposed system HPCRec enhances the quantity and quality of ratings of the user-item matrix by integrating historical purchases and click-stream data, therefore improves the recommendation qualities. Our experimental results show that our approaches outperform some existing systems referred in this paper. For future work, we plan to mine more information out of the historical data to improve recommendations such as how long ago a user purchased an item, and the frequent sequential purchase patterns, toward incorporating multiple data sources.

References

1. Aggarwal, C.C.: Recommender Systems. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29659-3
2. Ben-Shimon, D., Tsikinovsky, A., Friedmann, M., Shapira, B., Rokach, L., Hoerle, J.: Recsys challenge 2015 and the yoochoose dataset. In: Proceedings of the 9th ACM Conference on Recommender Systems, pp. 357–358. ACM (2015)
3. Bucklin, R.E., Sismeiro, C.: Click here for internet insight: advances in clickstream data analysis in marketing. J. Interact. Mark. **23**(1), 35–48 (2009)
4. Chen, L., Su, Q.: Discovering user's interest at e-commerce site using clickstream data. In: 2013 10th International Conference on Service Systems and Service Management (ICSSSM), pp. 124–129. IEEE (2013)
5. Gündüz, Ş., Özsu, M.T.: A web page prediction model based on click-stream tree representation of user behavior. In: Proceedings of the ninth ACM SIGKDD, pp. 535–540. ACM (2003)
6. Guo, G., Zhang, J., Sun, Z., Yorke-Smith, N.: LibRec: a java library for recommender systems. In: UMAP Workshops, vol. 4 (2015)
7. Hunt, J.W., MacIlroy, M.D.: An Algorithm for Differential File Comparison. Bell Laboratories, Murray Hill (1976)
8. Kim, Y.S., Yum, B.J.: Recommender system based on click stream data using association rule mining. Expert Syst. Appl. **38**(10), 13320–13327 (2011)

9. Kim, Y.S., Yum, B.J., Song, J., Kim, S.M.: Development of a recommender system based on navigational and behavioral patterns of customers in e-commerce sites. Expert Syst. Appl. **28**(2), 381–393 (2005)
10. Moe, W.W., Fader, P.S.: Dynamic conversion behavior at e-commerce sites. Manag. Sci. **50**(3), 326–335 (2004)
11. Park, Y.J., Chang, K.N.: Individual and group behavior-based customer profile model for personalized product recommendation. Expert Syst. Appl. **36**(2), 1932–1939 (2009)
12. Ricci, F., Rokach, L., Shapira, B.: Introduction to recommender systems handbook. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 1–35. Springer, Boston (2011). https://doi.org/10.1007/978-0-387-85820-3_1
13. Sismeiro, C., Bucklin, R.E.: Modeling purchase behavior at an e-commerce web site: a task-completion approach. J. Mark. Res. **41**(3), 306–323 (2004)
14. Van den Poel, D., Buckinx, W.: Predicting online-purchasing behaviour. Eur. J. Oper. Res. **166**(2), 557–575 (2005)
15. Weisstein, E.W.: Normal Vector: From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/NormalVector.html (2002)
16. Yun, U., Leggett, J.J.: WFIM: weighted frequent itemset mining with a weight range and a minimum weight. In: Proceedings of the 2005 SIAM International Conference on Data Mining, pp. 636–640. SIAM (2005)