Purchase Prediction and Item Suggestion based on HTTP sessions in absence of User Information

Sharif.TopMiners [pouya.esmailian@gmail.com]

Pouya Esmailian
Department of Computer Engineering
Sharif University of Technology, Tehran, Iran
pouya.esmailian@gmail.com

Mahdi Jalili*
School of Electrical and Computer Engineering
RMIT Universiy, Melbourne, Australia
mahdi.jalili@rmit.edu.au

ABSTRACT

In this paper, the task is to determine whether an HTTP session buys an item, or not, and if so, which items will be purchased. An HTTP session is a series of item clicks. A session has type *buy*, if it buys at least one item, or *non-buy* otherwise. Accordingly, data is in (session, item, time) format, which tells us when an item is clicked or purchased during an HTTP session. The main challenge comes from the fact that (1) user information is not available for clicked or purchased items, which are merely tagged with anonymous sessions, and (2) suggestions are highly temporal as they are suggested to sessions instead of users. In other words, users which are stable and identified are replaced with sessions which are temporal and anonymous. In this work, we propose a feature-based system that predicts the type of a session, and determines which items are going to be purchased. As the main contribution, we have modeled sessions separated by the number of unique items, prioritized item-features based on the number of clicks, and utilized cumulative statistics of similar items to attenuate the sparsity problem.

Keywords

Recommender Systems, RecSys Challenge, E-commerce Recommendation, HTTP sessions, Feature-based Methods

1. INTRODUCTION

Recommender systems (RS) play a central role in user navigation on the web. As the main goal, these systems try to put forward few directives to make the user experience as efficient and as fluent as possible. Currently, the application of RS is mostly involved with suggesting news articles [3], movies [2], friends [9], and E-commerce products [7] to users. Our task can be categorized as E-commerce recommendation, which tries to efficiently suggest products to users in order to increase the amount of purchases [8]. However, as the main difference, user informations such as ratings and the history of purchases are not available here. Instead, HTTP session of users is provided which consists of a short list of clicks and buys (3 items on average) at specific timestamps. This makes a huge difference since most of developed techniques rely heavily on activity history of users [6, 1]. As a connection, each HTTP session can be considered as a cold-start user [5].

In this paper, we propose a feature-based method that first determines whether a session is going to buy an item or not?, and if so, which items will be purchased?. Sessions were modeled separately based on the number of items. For the suggestion task, we utilized a simplifying fact that every purchased item was previously clicked during the session. Furthermore, features were built based on a key observation that the most clicked items were highly correlated to the type of their session (buy or non-buy). This observation helped us sorting items and building a suggestion model for each position separately. The rest of the paper is organized as follows. Sec. 2 describes the problem. Sec. 3 introduces the data set, our key observations and the proposed features. Section 4 illustrates our method. Section 5 shows the performance of method, and we conclude in Section 6.

2. PROBLEM DEFINITION

In this problem, the basic entities are HTTP sessions and items. Each session consists of series of items that are clicked or purchased at a specific time. The problem is to identify buy sessions and to specify purchased items. However, the problem will be much simpler if one utilizes the fact that all *buy* sessions end up purchasing only those items that are already clicked. For this reason, we restricted the problem of *including* purchased items to *predicting* the type of already seen items. This restriction drastically simplifies the problem since it only picks among already seen items. As a consequence a session s is represented as:

$$s : (i_1, T_1, y_1), (i_2, T_2, y_2)...(i_n, T_n, y_n), \quad (1)$$

where $i_x \neq i_y$ for $x \neq y$, T is a time vector of clicks, $|T|$ is the number of clicks, ΔT is the duration of clicks, y_j is one when item j is purchased at least once, and zero otherwise. Each item has various features introduced in Sec. 3.2. Each feature X of item i in session s is denoted by $X_s(i)$.

*Corresponding author

RecSys '15 Challenge, September 16-20, 2015, Vienna, Austria

http://dx.doi.org/10.1145/2813448.2813515 ...$15.00.

Accordingly, a session s has type y_s defined as

$$y_s = \begin{cases} 1 & \exists i : y_s(i) = 1 \\ 0 & \forall i : y_s(i) = 0 \end{cases}, \quad (2)$$

where $y_s = 1$ corresponds to a *buy* session. Finally, the problem is to (1) predict y_s for every test session s and (2) predict $y_s(i)$ for every $i \in [1, n]$. Item indexes are sorted such that:

$$|T_i| > |T_j| \text{ or } (|T_i| = |T_j| \text{ and } \Delta T_i > \Delta T_j) \text{ for } i < j. \quad (3)$$

A list of other notations is introduced in Table 1.

Table 1: Basic notations. Item and session are denoted as i and s, respectively. T is a period of time, e.g., last hour or Monday.

Symbol	Meaning
c_s	No. clicks in s
u_s	No. unique items in s
n_i	Set of items similar to i
$a_{s,i}$	i appeared (clicked or purchased) in s
$b_{s,i}$	i is purchased in s
$c_{s,i}$	i is clicked in s
$B_i(T)$	$\lvert\{s \lvert y_s = 1, a_{s,i}, t_s \in T\}\rvert$
$a_i(T)$	$\lvert\{s \lvert a_{s,i}, t_s \in T\}\rvert$
$b_i(T)$	$\lvert\{s \lvert b_{s,i}, t_s \in T\}\rvert$
$c_i(T)$	$\lvert\{s \lvert c_{s,i}, t_s \in T\}\rvert$
$b_{i,j}$	$\lvert\{s \lvert y_s = 1, a_{s,i}, a_{s,j}\}\rvert$
$b_{i \to j}$	$\lvert\{s \lvert a_{s,i}, b_{s,j}\}\rvert$

3. DATA SET

The YOOCHOOSE dataset consists of six months of user activity (clicks and buys) on an anonymous e-commerce website. However, click or buy events are not tagged with user IDs. In particular, they are merely known by their corresponding HTTP sessions. Also, for each item in a specific session there are extra informations such as time and category for all sessions and price and quantity (number of buys) only for buy sessions.

Table 2: Basic data set statistics.

Statistic	Value
Sessions	11,562,161
Labeled sessions	9,249,729 (80%)
Click sessions	8,740,033
Buy sessions	509,696 (5.5%)
Items	54287
Purchased items	19949 (37%)
$\overline{c_s}$	3.67
$\overline{u_s}$	2.88

3.1 Key observations

Here we introduce some key observations from data set, which helped us effectively select the features and design the final system.

3.1.1 *Number of unique items*

As shown in Fig. 1-a, the buy rate, $P(y_s = 1)$, almost linearly increases with the number of unique items u_s. This observation suggests that the buy distribution is shifted along u_s. Therefore, we decided to build separate models based on u_s to appreciate this significance. Although each model is deprived of other models' data resources, the abundance of buy sessions (500k) as compared to feature dimension (200) compensates for this problem. Also, as the number of buy sessions exponentially diminishes with u_s (Fig. 1), we placed sessions with $u_s \geq 7$ in one group.

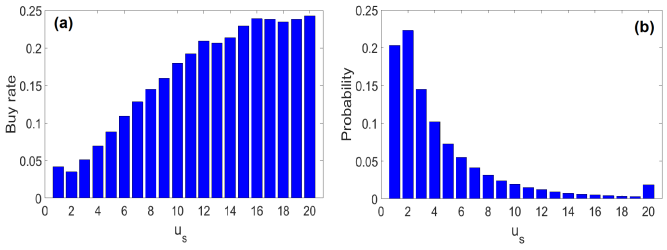

Figure 1: Distribution of (a) buyrate for sessions with length u_s, and (b) u_s for buy sessions. 20% of buy sessions have $u_s \geq 7$.

3.1.2 *Importance of click counts*

As experiments showed, the number of times item i is clicked in session s determines the impact of i on the type of s. As in Fig. 2-a, for a buy session with $u_s = 4$, probability of 1st item being purchased is 0.8 as compared to 0.5 for a randomly ordered set. Therefore, buy prediction and specially item suggestion tasks would benefit from placing features of these important items in special positions. This is similar to PCA and SVD methods which gather the most important features in specific positions of a vector.

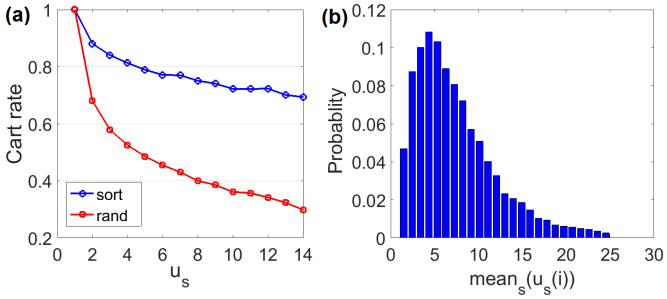

Figure 2: (a) Cart rate of 1st item sorted by rule (3), or chosen randomly. (b) Distribution of $\text{mean}_s(u_s(i))$ for purchased items. A purchased item appears in a session with $u_s \simeq 7$ on average.

3.1.3 *Length of buy sessions for each item*

Fig. 2-b demonstrates distribution of average length of buys sessions for each item. For many items, it shows a significant deviation from mean of distribution (7.36), signifying that they tend to be purchased in longer or shorter sessions. Indeed, this suggests not only that sessions have different buying patterns based on u_s, but also some items tend to appear in specific u_s regions more often. This further supports our decision on separating the model of sessions based on u_s to deal with more specific feature regions.

3.1.4 *Similar items*

We observed that there exist pairs of items which frequently (over 100 times) co-appear in buy sessions, and many of them exhibit a similar visiting pattern through time. Therefore, we built an item-item network to utilize this correlation[1]. A weight from item i to j is defined as $w(i,j) = b_{i\to j}/b_{i,j}$. Accordingly, set of similar items to i is defined as $n_i = \{j|w(j,i) > 0\}$. That is, when one visits $j \in n_i$ and i, there is $w(j,i)$ chance of purchasing i.

3.2 Features

Here are a list of features that were used for a session s:

- **Local features.** These features are only a function of s such as: number of clicks c_s, time duration, item prices, and item categories.

- **Temporal features.** These features depend on item i, time $t_s(i)$ and training data, such as buy rate defined as $br_i(T) = b_i(T)/c_i(T)$, and cart rate defined as $cr_i(T) = b_i(T)/B_i(T)$ for every $i \in s$, and T = last hour, last day, last week, hour of day (24 states), and day of week (7 states). The intuition is that having higher $br_i(T)$ ($cr_i(T)$) for any $i \in s$, roughly means higher probability of $y_s = 1$ ($y_s(i) = 1$) corresponding to the buy prediction (item suggestion) task. Also, to utilize Markovian pattern of buy events, we introduced $m_i(t|t')$ for each item, which is defined as:

$$m_i(t|t') = \frac{\sum_\tau b_i(\tau|\tau - \Delta t)}{\sum_\tau c_i(\tau|\tau - \Delta t)}, \quad (4)$$

  where $\Delta t = t - t'$, and $\sum_\tau b_i(\tau|\tau - \Delta t)$ is the number of sessions that purchased i, Δt after the previous buy (similar for c_i).

- **Non-temporal features.** These features depend on item i and training data, but they are independent of $t_s(i)$. Regarding Sec. 3.1.3, buy rate as a function of u_s is introduced as $br_{i,u} = b_{i,u}/c_{i,u}$, where only sessions with $u_s = u$ are considered and T = the whole time span.

- **Cumulative features.** These statistics treat similar items the same by cumulating their appearances, e.g., $br_{n_i}(T)$ is defined as:

$$br_{n_i}(T) = \frac{\sum_{j\in n_i} B_j(T)}{\sum_{j\in n_i} c_j(T)}. \quad (5)$$

  As an intuition, assume item i is purchased during the last day and it is similar to item j. Although there is no evidence for item j being purchased, cumulative features will increase the chance for item j. These features are most powerful when item statistics are highly sparse, e.g., $br_i(T)$ when T = last hour or day.

- **Global features.** These statistics are only a function of t_s such as: global buy rate which is defined as:

$$br(T) = \frac{\sum_i B_i(T)}{\sum_i c_i(T)}. \quad (6)$$

[1]Only item pairs that co-occurred more than 15 times were considered significant.

4. PROPOSED METHOD

Proposed method is schematically represented in Fig. 3. In the first part, an unlabeled session is fed into an appropriate buy model based on u_s. Next, when the model classifies the session as *buy*, it is fed into the item suggestion models that are built for sessions with $u_s = n$, e.g., if the session has $u_s = 3$ items it will be fed into three models $(3,1)$, $(3,2)$, and $(3,3)$, where model $(3,p)$ determines whether p-th item (using sort rule (3)) will be purchased or not. Thus, the session's final cart will be the list of accepted items. We used Random Forest classifiers for both the buy and suggestion models since, on this particular task, it outperformed kernel SVM with respect to the accuracy and runtime[2]. Buy Model n was built as follows:

1. Equal number of buy and non-buy sessions with $u_s = n$ were extracted (uniformly random) from labeled data (10,000 for each class). For $n = 7$, all sessions with $u_s \geq 7$ were considered together.

2. Session features were calculated (ignoring extracted sessions) according to Fig. 4-b and descriptions of Sec. 3.2.

3. A Random Forest model was trained on 75% of extracted sessions.

And Suggest Model (n,p) similarly was built as follows:

1. The extracted 20,000 sessions were split into two classes that either buy item p, or not.

2. Session features were calculated (ignoring the extracted sessions) according to Fig. 4-c and descriptions of Sec. 3.2.

3. A Random Forest model was trained on 75% of extracted sessions.

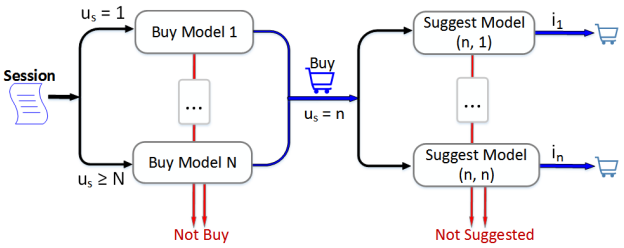

Figure 3: Proposed method. Buy Model n is responsible for sessions with $u_s = n$. Suggest Model (n,p) is responsible for p-th item of those sessions.

5. EXPERIMENTS

We tested our system using the hold out 25% extracted sessions (discussed in Sec. 4). The results are reported in Table 3 for buy prediction task and Table 4 for item suggestion task. We chose our metrics in accordance with the score formulation of RecSys challenge that rewards correct buy sessions (True Positive) and punishes incorrect buy sessions which are truly non-buy (False Positive). For the item

[2]A wide range of classifiers were tested using Weka software [4]

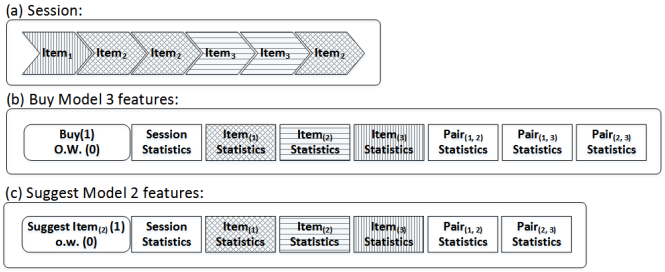

Figure 4: Creating a feature vector for a session. Item_i is the i^{th} clicked item in a session and $\text{Item}_{(i)}$ is the i^{th} important item after applying the sort rule (3). $\text{Pair}_{(i,j)}$ statistics are based on i^{th} and j^{th} items.

suggestion task, final metric is the Jaccard index between the suggested set and true set. It is worth mentioning that the challenge score would be rewritten in above metrics as:

$$score = |S_b| \left(\left(J + \frac{|S_b|}{|S|} \right) TP - FP \right) \tag{7}$$

Where $|S|$ ($|S_b|$) is number of total (buy) sessions in the test set, and J is the average Jaccard index per buy session, which is $J \simeq 0.79$ for our system based on the hold out set.

Table 3: Performance of buy prediction models. TP is the ratio of correct buy sessions to all buy ones, and FP is the ratio of incorrect buy sessions to all non-buy ones.

Buy Model	1	2	3	4	5	6	7
TP	0.79	0.64	0.71	0.77	0.76	0.87	0.77
FP	0.26	0.17	0.21	0.27	0.28	0.41	0.32

6. CONCLUSIONS

In this paper, we resolved the problem of buy prediction and item suggestion for e-commerce HTTP sessions. The main challenge, as compared to ongoing researches on recommender systems, was the absence of user information. Thus, the task was to suggest items to a temporary session that consists of a short (around 3) list of items with no previous history. Although at the first glance this problem seems very difficult as compared to user-item paradigms, we found that a satisfying model can be built that recalled 75% of true buy sessions at the cost of incorrectly including 25% of non-buy ones, as well as, a promising item suggestion method with a Jaccard index over 79%. As the key idea, we considered item clicks as the most important factor for HTTP sessions, and also modeled them separately based on the number of items. Considering the absence of user information as the worst case scenario, their inclusion, i.e., assignment of HTTP sessions to users, will evolve the task into a more interesting one which further leads to great improvements on the final result.

Table 4: Performance of item suggestion models. The average Jaccard index for a buy session is over 0.79.

Suggest Model	1	2	3	4	5	6	7
Jaccard	1.00	0.90	0.78	0.72	0.72	0.67	0.60

7. REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

[2] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez. Social knowledge-based recommender system. application to the movies domain. *Expert Systems with Applications*, 39(12):10990–11000, 2012.

[3] F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized news recommendation with context trees. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 105–112, New York, NY, USA, 2013. ACM.

[4] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[5] J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 199–206. ACM, 2010.

[6] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[7] J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. In *Applications of Data Mining to Electronic Commerce*, pages 115–153. Springer, 2001.

[8] Ben-Shimon, D., Tsikinovsky, A., Friedman, M., Shapira, B., Rokach, L., Hoerle, J. (2015, September). RecSys Challenge 2015 and the YOOCHOOSE Dataset. In Proceedings of the 9th ACM conference on Recommender systems. ACM

[9] N. B. Silva, I.-R. Tsang, G. D. Cavalcanti, and I.-J. Tsang. A graph-based friend recommendation system using genetic algorithm. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1–7. IEEE, 2010.