In-House Solution for the RecSys Challenge 2015

Conference Paper · September 2015

DOI: 10.1145/2813448.2813519

6 authors, including:

David Ben-Shimon
Sigmabit
15 PUBLICATIONS **78** CITATIONS
SEE PROFILE

Bracha Shapira
Ben-Gurion University of the Negev
158 PUBLICATIONS **4,743** CITATIONS
SEE PROFILE

Lior Rokach
Ben-Gurion University of the Negev
298 PUBLICATIONS **10,150** CITATIONS
SEE PROFILE

In-House Solution for the RecSys Challenge 2015

Nadav Cohen
Ben-Gurion University of the Negev, Beer-Sheva, Israel
conad@post.bgu.ac.il

Adi Gerzi
Ben-Gurion University of the Negev, Beer-Sheva, Israel
gerzi@post.bgu.ac.il

David Ben-Shimon
YOOCHOOSE Labs, Omer, Israel
david.ben-shimon@yoochoose.com

Bracha Shapira
Ben-Gurion University of the Negev, Beer-Sheva, Israel
bshapira@bgu.ac.il

Lior Rokach
Ben-Gurion University of the Negev, Beer-Sheva, Israel
liorrk@bgu.ac.il

Michael Friedmann
YOOCHOOSE GmbH, Cologne, Germany
michael.friedmann@yoochoose.com

ABSTRACT

RecSys Challenge 2015 is about predicting the items a user will buy in a given click session. We describe the in-house solution to the challenge as guided by the YOOCHOOSE team. The presented solution achieved 14th place in the challenge's final leaderboard with a score of 51,932 points, while the winner obtained 63,102 points.

We suggest two simple and easy to reconstruct approaches for obtaining a prediction in each session. In the first approach we suggest one classifier to determine whether each item in the session will be bought. In the second approach we suggest a two level classification model in which the first level determines whether the session is going to end with a purchase or not, and if it ends with a purchase, the second level classification determines the items that are going to be purchased.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information filtering; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Recommender systems, RecSys Challenge 2015, In-House Solution

1. INTRODUCTION

Many e-commerce businesses use recommender systems to suggest products to their users. This allows them to boost sales and, at the same time, increase the level of satisfaction among users. Many visitors to these websites are casual users who come to the site randomly, read content that appears on the site, and eventually either make a purchase or leave the site without having made a purchase. In RecSys Challenge 2015[1] YOOCHOOSE released data that matched the profile described above. The data comprises a large collection of user visits to a large European retailer's website. Each user visit is also referred to as a session, and each session is comprised of a sequence of item clicks the user performed during the visit, as well as any items the user purchased. The aim of the challenge is to determine which visits ended with a purchase and if indeed there was a purchase then the second goal it to reveal the items in that purchase. In order to do so, the training data contains buy and click events, and the test data comprises only click events.

DOI: http://dx.doi.org/10.1145/2813448.2813519

Only five percent of the visits in the training data contain at least one buying event, thus the data is significantly imbalanced. Because the algorithm we use requires balanced data, this issue must be treated at the outset. Intuitive solutions is to under sample the majority class or to over sample the minority in order to create a balanced testbed. Moreover, rather than maximizing a regular evaluation metric such as precision, RMSE, or the like, in this challenge there is a need to maximize a specific scoring function as presented in equation 1.

$$Score(Sl) = \sum_{\forall s \in Sl} \begin{cases} if\ s \in Sb \rightarrow \frac{Sb}{S} + \frac{|As \cap Bs|}{|As \cup Bs|} \\ else \rightarrow -\frac{|Sb|}{|S|} \end{cases} \quad (1)$$

For each session *s* that relates to the *Sl* sessions in the solution file, if *s* relates to the *Sb* positive sessions than add the score $|Sb|/|S|$ to the overall score where *S* is the total number of sessions in the test file, and also compute the size of the Jaccard between the predicted items for the session (*As*) and the actual purchases in the session (*Bs*) that were removed from the test file. Otherwise penalize the size $|Sb|/|S|$ of the overall score. Therefore for each click session *s* in the test set *Sl* in the solution file, the goal was to determine a set *As* of items that the user bought during *s*. If *As* is empty then it means no items were purchased. From now on we denote session with a purchase as a positive session and session without a purchase as a negative session.

There are no user IDs in the data that will enable a connection between the sessions, and therefore building a type of collaborative filtering model is not possible. Also, the metadata of the items is not of a high quality – for example only the category of the items, which is in place for only ~50% of the cases, and price in the training data are available. Therefore, there is not enough information to evaluate content based models. As a result of the limited information available in the data, we decided to focus on finding patterns in the sessions' click sequences, particularly patterns that exist when the likelihood of a purchase in a given click sequence is high.

In this paper we present two approaches for solving the problem. In the first approach we suggest a binary classifier to determine whether each item clicked in a session will be purchased or not. In

the second approach we suggest two binary classifiers where the first one determines whether a session is positive or not, and the second classifier determines whether the items are going to be purchased or not in the predicted positive sessions. The first approach is easier, because it is based on one classifier, however, the second approach is more flexible and enables more room for mistakes. In this paper we present the two approaches, each of which requires slightly different set of features.

The remainder of this paper is organized as follows: Section 2 surveys the analysis to the data and the feature engineering. Section 3 thoroughly describes the proposed approaches. Section 4 describes the experimental study and the comparative results of the proposed methods, and the paper concludes with a discussion in Section 5

2. DATA ANALYSIS AND FEATURE ENGINEERING

The data for the challenge includes two training files. The first is **yoochoose-clicks.dat** where each record/line describes a click and has the following fields: session ID, time stamp, item ID, and item category. The second file is **yoochoose-buys.dat** where each record/line describes a purchase and has the following fields: session ID, time stamp, item ID, price, and quantity. Merging all of the clicks from the clicks file for a given session ID together with all the buys from the buys file for the same session ID, forms the user's entire activity in that specific session. The test data comprises only the click events from the tested sessions, and the goal is to predict the buy events that will take place in these sessions. Table 1 presents some of the data's basic statistics.

Table 1. Data facts

	#sessions	#clicks	#buys	#items
Training Set	9,249,729	33,003,944	1,150,753	52,739
Testing Set	2,312,432	8,251,791	?	42,155

2.1 Statistics

For generating features we explored the training set for revealing patterns in the data, and figures 1 through 4 present some of our most important findings. Figure 1 suggests that buying events occur mainly between 6:00 am and 8:00 pm. Figure 2 shows that buying events are most likely to occur on weekends rather than on weekdays, with Tuesdays being the weakest day of the week for purchases. Figure 3 suggests that if the item has been clicked on a few times in a session, the probability that the item will be bought is high. Figure 4 demonstrates that when users spent more time on an item in a session, the probability of buying this item increases. Figure 5 suggests that the duration of the session provide an indication as to whether the session will end in a purchase or not – longer sessions leads to a purchase.

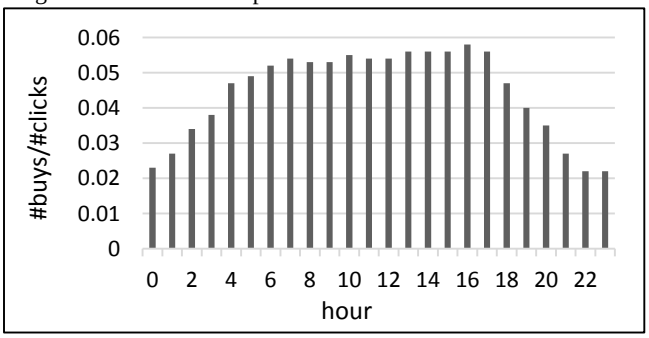

Figure 1. Buy ratio as a function of the time of day

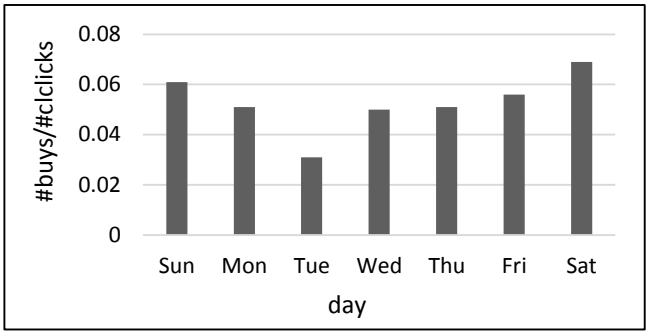

Figure 2. Buy ratio as a function of the day of the week

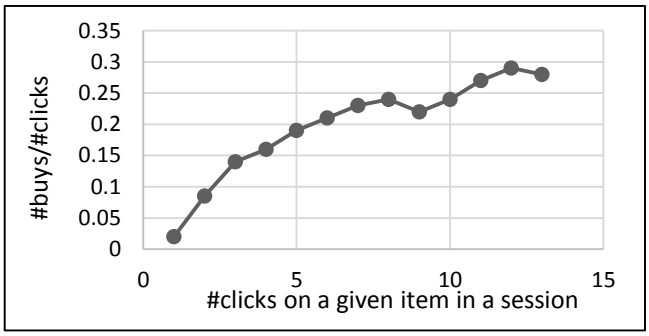

Figure 3. Buy ratio as a function of the number of clicks on a given item in a session

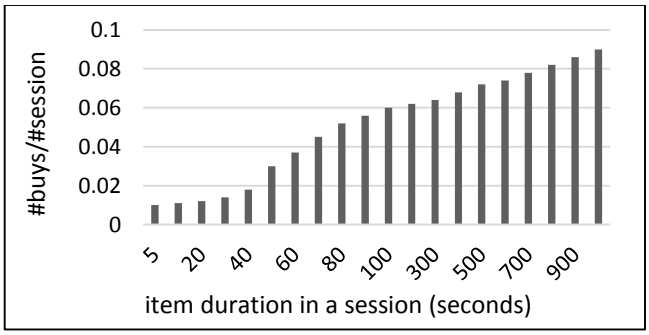

Figure 4. Buy ratio as a function of the time spent on an item in a session

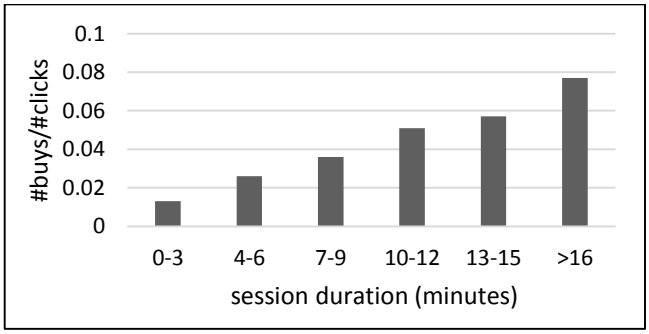

Figure 5. Buy ratio as a function of the session's duration

Besides the important findings presented above, and based on our data analysis we also extracted the following from the training data.

- Positive sessions have 6.78 clicks on average, while negative sessions have only 3.39
- There are only 509,696 positive sessions in the training set

- Positive sessions last for 866 seconds on average, while negative sessions last only 352 seconds
- The distribution of the buys over the categories is varied with some categories attracting more buys than others
- The distribution of the buys across the items is varied with some items being purchased more frequently than others
- Items which were clicked on first or last in the session have a higher chance of being bought than items clicked on in the middle of the session

Items which have been clicked on more than once in a session are three times more likely to be bought than items which have been clicked on only once

Together, these facts help us enrich the data and engineer new features as presented in the next section.

2.2 Feature Extraction and Selection

After analyzing the data, we define multiple features and divide the features into a group of features by sessions (**FS**) and a group of features by item (**FI**).

2.2.1 Features by Session

The **FS** provides various aspects of a session in order to determine if the session is positive. We started with 16 features, but after running several feature selection algorithms (CfsSubset, OneR, InfoGain, and GainRatio) and applying a simple voting mechanism among their decisions, we ended up with a list of the following 13 features.

FS-1: Session time in seconds
FS-2: Average time between two clicks
FS-3: Maximal time between two clicks
FS-4: Day of the week
FS-5: Month of the year
FS-6: Time during the day – morning | afternoon | evening | night
FS-7: Number of clicks
FS-8: Maximal number of clicks on one item
FS-9: Percentage of items in the session that are popular – we compute an overall popularity score for each item as #buys/#clicks.
FS-10: Number of distinct items
FS-11: Average price of clicked items
FS-12: Percentage of "S" category from all categories of the items
FS-13: Percentage of items that have category

A unique identifier of a record containing the **FS** set of features will be the session ID.

2.2.2 Features by Item in Session

The **FI** set of features provides various aspects on an item in a specific session in order to determine if the item is going to be bought. This set is comprised of features which refer to global information about the item based on all of the training data, as well as features that provide local information about an item's lifespan in a specific session. Here we started with 56 features, but after we applied the feature selection process described above we ended up with the following list of 22 features.

FI-1: Whether the item appears more than once in the session
FI-2: Whether the item was clicked first in the session
FI-3: Whether the item was clicked last in the session
FI-4: Number of appearances in the session
FI-5: Number of clicks on that item divided by the number of clicks on the most clicked on item in the session
FI-6: Number of clicks on that item divided by the average number of clicks on items in that session
FI-7-10: Number of clicks in the session before and after the first and last click on that item (four features in total)
FI-11: Number of clicks before first and last click on the item divided by the total number of clicks in the session
FI-12: Price of the item
FI-13: Popularity score of the item
FI-14: Rank of the item according to the popularity score
FI-15: Number of previous buys recorded for the item
FI-16: Percentage of previous buys for that item
FI-17: Time gap from first to last click on the item in that session
FI-18: Average time gap between clicks on the item in the session
FI-19: Maximum time gap between clicks on the item in the session
FI-20: Time gap from the first click on the item to the last click on it divided by the length of the session
FI-21: Minimum time gap between the clicks on that item in the session
FI-22: Is the item at the peak of its sales

A unique identifier of a record containing the **FI** set of features will be a concatenation between the item ID and the session ID.

3. METHOD

3.1 Generating Balanced Data

The fact that the data is imbalanced and the need to optimize the solution to a customized measure were two important obstacles in the challenge. To tackle the issue of imbalanced training data we tried sever solutions, however we found that under sampling the majority class to create a balanced testbed worked best for us. Given that the training data had 509,696 positive sessions, we randomly sampled 509,696 negative sessions from the training data and used all of these sessions as our training data. The negative samples were extracted from the entire period, and we maintained the sessions' distribution size.

3.2 Classification of Item per Session Approach - CIPS

The first and easier approach is to classify each item in a session as a buy or not. We denote this approach as CIPS. If, in a given session, none of the items was classified as a buy, then the session is said to be negative, eliminating the need to apply a classifier for the session. The strengths of this approach are that it requires only one classifier and that it takes advantage of the fact that there is more to gain (in terms of scoring) by predicting the items correctly than by predicting the sessions correctly. According to equation 1, predicting the session correctly increase the score by 0.05 points, while an incorrect prediction penalizes it by -0.05. Predicting the exact items in a session may add one point to the score, thus it is slightly more valuable to focus on the items. We used feature sets **FI** and **FS** to generate a new input dataset to this classifier. This new dataset was engineered from the generated balanced data described in section 3. The reason behind using the **FS** set of features for an item in a session classification is that if it is effective for the classification of sessions, it will likely be correlated to items that are in buying sessions. Each record in the new dataset was made up of 35 features and an additional binary class attribute indicating whether the item was bought in the session or not. We also sampled 500,000 negative records and an equal number of positive records and used it as balanced input to the algorithm.

We examined several ensemble methods using Weka and applying the above described settings, and we revealed that random subspace provide us with the best results. Random subspace consists of multiple trees constructed systematically by pseudo randomly selecting subsets of components of the feature vector, that is, trees constructed in randomly chosen subspaces [2]. The base classifier in the random subspace was the REPTree. The REPTree is a fast

decision tree learner that builds the tree using information gain and pruning it using reduced-error. We ran the random subspace for 50 iterations, each with a subspace consisting of half of the 35 features.

3.3 Two Level Classification Approach - TLC

In this approach we divided the challenge's task into two levels: predicting buying sessions in the first level and predicting the items that are going to be bought in these sessions in the next level. We denote this approach as TLC. To do so, we built a classifier for the sessions using **FS** and the ~1 million records of balanced data, and we also built a classifier for the items using **FI** and only the ~500,000 positive sessions. Note that for building the classifier in the second level we use only the positive sessions. For each session in the test data we used the session classifier to determine whether it is positive or negative, and for each item in each session that was predicted as positive we use the item classifier to determine whether it is going to be bought or not. We used the same algorithm and settings as in the CIPS approach described previously, i.e., we trained the two classifiers for both levels using the random subspace with REPTree as the base classifier.

3.4 Filtering Rules

By analyzing the predictions which generated by our two approaches we noticed that the rate of the false positive classification is very high, since we predicted significantly more than 5% of the test sessions as positive. We also discovered that while the percentage of positive sessions with one item purchased was 51% out of the total number of positive sessions, in our solutions it was 60%. We assume that this is due to the false positive classification. Hence, we conclude that by filtering out the predicted items in these sessions we will also discarding sessions which are not buying sessions, removing the negative consequence. We therefore increase the threshold to 0.6 in the sessions consisting of one buying event which improves the score by ~800 points for both approaches. Increasing the threshold to 0.6 for all the sessions reduced the false positive error, however it also reduced the true positive classification, and therefore wasn't effective at improving the overall score. In addition, in each predicted positive session with more than four items in the list, we also increase the threshold to 0.6 to reduce the list in a manner similar to the previously described filter. That also improved our solutions by an additional several hundred points. Figure 6 presents the overall structure and the components of the two approaches.

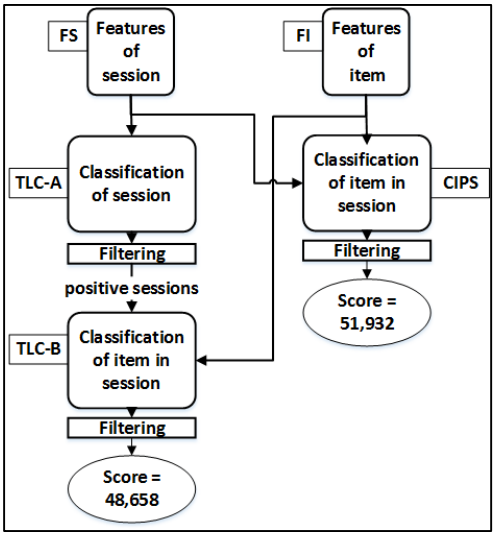

Figure 6. Overview of the methods, features, and scores

4. RESULTS

The results of the described approaches along with the algorithms' parameters are shown in Table 2. The CIPS approach performed better than the TLC approach. In terms of the number of sessions (after filtering rules), TLC produced 597,453, while CIPS produced 649,946. In general, solutions that were smaller than 500,000 sessions scored lower than larger solutions.

Table 2. Results

Approach	Parameters	Score
Two Level Classification (TLC)	First-Level - 13 Features Second-Level - 22 Features Both: Random subspace - 50 iterations REPTree - half of the features Filtering Rules - 0.6 threshold on sessions with 1 item	48,658
Classification of Item per Session (CIPS)	35 Features Random subspace - 50 iterations REPTree - 24Features Filtering Rules: 0.6 threshold on sessions with 1item 0.6 threshold on sessions > 4 items	51,932

5. CONCLUSION AND DISCUSSION

In this paper we describe two simple approaches for solving the RecSys Challenge 2015. The TLC approach builds two classifiers, and the CIPS approach builds only one classifier. The CIPS approach outperformed the TLC by a few thousand points. We believe that this is due to the fact that there is a larger number of features to extract for CIPS since it can benefit from information on the item from the overall data as well as from a single session. Nevertheless, the TLC can be generalized more easily to other problems, because of its ability to separate the task into a two-step prediction process.

In the TLC approach, according to the confusion matrix and area under the curve measures, we noticed that the first classifier (session) performed less well than the second classifier (item). We hence believe that the session classifier could be improved. We compared the solution files of the two approaches and found that they differ in terms of sessions and items predicted. This implies that ensemble the two into one approach may improve the score.

The solution presented in this paper is based on the work conducted by the paper's first two authors who registered for the challenge and competed like the other challenge participants. This work served as the basis of the final project of their Bachelor of Science degree in software engineering. The remaining authors included the supervisors of this project and one of the challenge's organizers. This paper doesn't present the best solution to the challenge but rather presents a simple and easy to reconstruct solution from the organizers' perspective.

6. REFERENCES

[1] Ben-Shimon, D., Tsikinovsky, A., Friedman M., Shapira, B., Rokach, L., Hoerle J. RecSys challenge 2015 and the YOOCHOOSE dataset. In ACM RecSys, 2015.

[2] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. ACM SIGKDD explorations newsletter, 11(1), 10-18.