Probability-based Approach for Predicting E-commerce Consumer Behaviour Using Sparse Session Data

Øyvind H. Myklatun
Department of Computer and Information Science
Norwegian University of Science and Technology

Thorstein K. Thorrud
Department of Computer and Information Science
Norwegian University of Science and Technology

Hai Nguyen*
Telenor Research,
7052 Trondheim, Norway
haithanh.nguyen@telenor.com

Helge Langseth
Department of Computer and Information Science
Norwegian University of Science and Technology

Anders Kofod-Petersen
Telenor Research
7052 Trondheim, Norway
Department of Computer and Information Science
Norwegian University of Science and Technology

ABSTRACT

This paper describes some of the key properties of the proposed solution for the RecSys 2015 Challenge from the team Tøyvind thørrud. Three contributions will be highlighted: *i*) Feature extraction, *ii*) Classifier design, and *iii*) Decision rules to optimize the prediction results towards the RecSys Challenge's score. We finished sixth out of more than 250 active teams in the competition.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Data mining

General Terms

Recommender systems; Consumer behaviour; Classification

Keywords

Feature extraction; Random forest; Decision rules

1. INTRODUCTION

The Recsys Challenge 2015[1] is concerned with sequences of click events, and the task is to decide whether a given session results in items being bought, and if so, which items are purchased. The data is collected over six months, and contains 9 249 729 sessions, out of which only 509 696 sessions are buying-sessions. Participants were evaluated on a separate test-set which contains 2 312 423 sessions. For each click-session s, the contestants were asked to determine a set A_s of items they assumed the user bought during s; $A_s = \emptyset$ would signify that no items were purchased. The prediction would be compared to the actual purchase from session s, B_s and scored as

$$\text{Score}(\mathbf{Sl}) = \sum_{\forall s \in \mathbf{Sl}} \begin{cases} \frac{|S_b|}{|S|} + \frac{|A_s \cap B_s|}{|A_s \cup B_s|} & \text{if s} \in S_b \\ -\frac{|S_b|}{|S|} & \text{otherwise} \end{cases}, \quad (1)$$

where **Sl** are sessions assumed to have a purchase, S are all sessions in the test-set and S_b are sessions in test-set which end with a buy.

We decided to divide the problem into two classification tasks: one that chooses if a session is a buying session (termed "Buy-or-not"), and one deciding which items are bought in a buying session (named "What-to-buy"). We note that our approach appears to be overly complicated, in the sense that one in theory could achieve the same result using only the "What-to-buy" classifier. If this classifier decides to not buy any of the items, it will in effect be the same as if the "Buy-or-not" classifier decides it is a non-buying session. However, somewhat surprisingly, our results using only the "What-to-buy" classifier were vastly inferior to those obtained by the two-tier setup, and we therefore separated the problem into two subtasks.

To solve these tasks, first we needed to analyse the statistical characteristics of the training data to extract valuable features which reflected consumer behaviour. These characteristics are click-based behaviours, such as the number of clicks, the duration of a session, the number of consecutive returns to items, and so on. In total, we extracted 7 features for the "What-to-buy" task and 11 features for the "Buy-or-not" task. Second, we evaluated different classification algorithms on the training data by using accuracy measurements like *precision*, *recall* and *ROC*, as well as the RecSys Challenge's score. It turned out that the *Random Forest* classifier [2] worked best for both classification problems. Third, we used the trained classifiers to predict the results on the provided test-set. Finally, we applied the proposed decision rules to optimize our decision wrt. the RecSys

*Corresponding author.

Challenge's score and submitted our solution.

Our contributions to the RecSys Challenge are as follows:

- We have extracted important features for the "Buy-or-not" and "What-to-buy" classification tasks.
- We have found that Random Forest is the best classifier for both classification tasks.
- We have proposed rules to optimize our prediction results with respect to the RecSys Challenge's score.

2. FEATURE EXTRACTION

As discussed in the previous section, the problem was divided into two classification tasks, which will be solved in sequence: first using the "Buy-or-not" classifier, then "What-to-buy". Therefore, two different sets of features are needed, one set is used to determine if a session ended with a purchase, the other for classifying items as bought or not in the buying sessions. In the following, we describe how to extract meaningful features from the training data for the two classification tasks.

Before getting into details about the selected features, we want to highlight some characteristics of the training data that might affect the quality of the extracted features: First, we emphasize the sparsity of the session data, where there is not much information within a session. Only the clicked items (with time-stamps) are known. Therefore, finding a good set of features to differentiate between sessions is a challenge. Second, the information about an item's type (it being clothes, tools and so on), is not available and there is a large number of missing values when it comes to information about prices and categories. It is therefore challenging to build informative profiles or models for the different items. Third, the data is imbalanced, with only approximately 5% of the sessions ending with a purchase.

Extracting relevant features obviously has a significant impact on the performance of the classifiers, with better classification accuracy following from clever feature design. The importance of different features was analysed offline through statistical properties of the training data, and was also tested online using the challenge score.

Below we list the most relevant features for both classification tasks. As some of the features for the "What-to-buy" classifier were aggregated to construct features for the "Buy-or-not" classifier, we will first describe the features for "What-to-buy". The features describe one particular item being clicked during a given session:

F1: Indicator of the item being the first item clicked.
F2: Indicator of the item being the last item clicked.
F3: Number of times the item was clicked in the session.
F4: Time spent on the item during the session.
F5: Baseline probability of the item being bought.
F6: Number of consecutive clicks on the item.
F7: Maximal duration between consecutive clicks on item.

Features F1 and F2 are included because the items clicked first or last have higher probability of being bought (approx. 0.7) compared to the items clicked in between (approx. 0.4).

The number of clicks on an item during a session, which is reflected in the feature F3, also indicates buying intention. Items that were clicked once are purchased in 35% of the relevant sessions, while the percentage increases to above 80% if an item has been clicked four times or more. A similar observation was made for feature F4, which is the time spent on the item during a session.

We included the probability-based feature F5, which is the

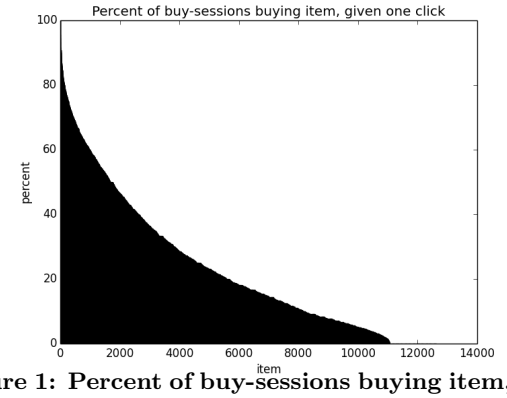

Figure 1: Percent of buy-sessions buying item, given one click on the item.

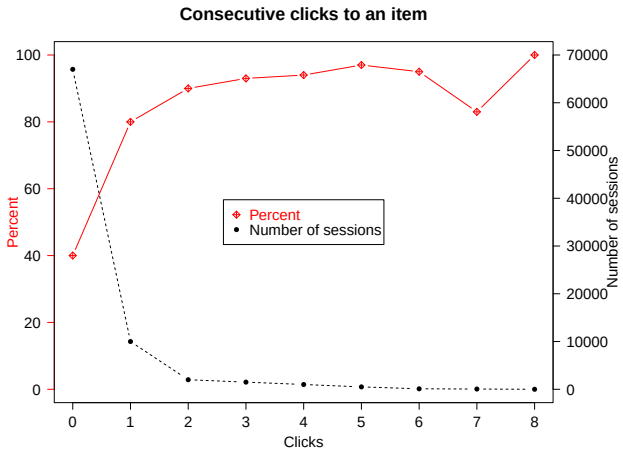

Figure 2: Percent of items bought, given consecutive clicks to the same item (red and solid line, left y-axis) and the number of sequences where this happens (black and dashed line, right y-axis).

portion of times when a specific item has been bought over sessions in a period of time. This feature is relevant because, as shown in Figure 1, some items were bought more often than others. More details on how to set up the period of time for calculating the probability will be discussed in Section 4.

Features F6 and F7 are related to the consecutive return to an item, i.e., repeated clicks on the same item without clicking another item in-between (and we use "1 consecutive click" to denote that the item has only been clicked once). In Figure 2 we observed that the probability of an item being bought increases when the number of consecutive clicks increases. In fact, if the item is not clicked repeatedly the percentage of items bought is about 40%, while repeated clicking increases the percentage to over 80%.

For the "Buy-or-not" classification task, we used the following features describing a given session:

P1: Total number of clicks during the session.
P2: Average number of clicks per item.
P3: Total duration of the session.
P4: Maximal duration between two clicks.
P5: Average duration between clicks in the session.
P6: Maximal number of times the session has clicked on the same item.
P7: Average of items' buying probabilities in the session.
P8: Maximum of items' buying probabilities in the session.

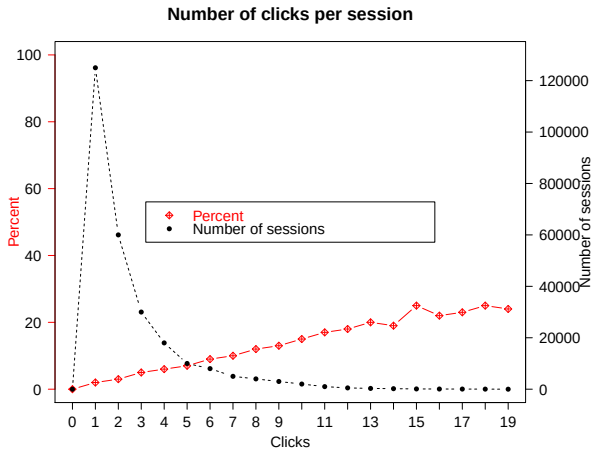

Figure 3: Relation between the number of clicks and percent of buy-sessions (red and solid line, left y-axis), and the number of sequences where this happens (black and dashed line, right y-axis).

P9: Probability of buying at least one item in the session.
P10: Maximal number of consecutive clicks on any item in the session.
P11: Maximal duration between two consecutive clicks on any item in the session.

As the data is made up by click events, it is natural to analyse how the different click patterns affect the probability of a session ending in a purchase. This is done by looking at different statistical characteristics of the clicks in the session, which are described in the features from P1 to P6. For example, Figure 3 shows that the probability of buying increases as the total number of clicks increases. Similar observations were made regarding features from P2 to P6.

For the features P7 and P8 we aggregated the probabilities of the session buying its items (calculated as feature F5 for the "What-to-buy" classification task). Its importance is obvious, as the probability of an item being bought is related to the probability of a session being a buy-session.

As mentioned, some items have a higher probability of being bought than others. Furthermore, a session that contains at least one item with a very high probability of being bought is likely to be a buy-session. We wanted to use this information in our prediction of buy-sessions, and thus extracted the feature P9, which is the probability of the session buying at least one item. An example on how we can calculate the value for P9 is as follows: For a session that has visited three items with corresponded probabilities 0.4, 0.6 and 0.7, the probability of the session buying at least one item would be $(1-(1-0.4)\cdot(1-0.6)\cdot(1-0.7)) = 0.928$.

Lastly, we aggregated the features F6 and F7 from the "What-to-buy" feature list to construct the features P10 and P11, which reflect the consecutive returns of different items in a session.

3. CLASSIFIER DESIGN AND DECISION RULES TOWARDS THE SCORE

In the classifier design of our solution, we chose to evaluate *probabilistic* classifiers for the "Buy-or-not" and the "What-to-buy" classification tasks. This is because the probability values calculated by these algorithms can be utilized to optimize our decisions wrt. the score given in (1). In the following, we will motivate the approach.

Let p_s be the calculated probability that the session s ended with a purchase. According to the score given in (1), the expected value of the score for the session s when it is classified as a buying session is

$$\mathbb{E}[\text{Score}(s)] = p_s \left(\frac{|S_b|}{|S|} + \frac{|A_s \cap B_s|}{|A_s \cup B_s|} \right) + (1-p_s)\left(-\frac{|S_b|}{|S|} \right). \tag{2}$$

The idea now is that the session s should only be classified as a buy-session if the expected value is positive, that is, when $\mathbb{E}[\text{Score}(s)] > 0$. This insight provides a hint towards how to use the results from the classifiers in our decision rules. We can estimate the value $|S_b|/|S|$ from the training data; 509 696 of the 9 249 729 sessions are buy-sessions, giving $\nu = |S_b|/|S| \approx .0551$. In the following, we will consider how to estimate the score of different classification choices and discuss how that leads to the *decision rules* we used to try to maximize the expected score of our solution.

3.1 The "Buy-or-not" classifier

Given features describing the session s, the output of the classifier was the probability p_s, namely the probability that the session ended with a purchase. Naïvely, one may decide a fixed decision rule, and call a session s with $p_s \geq .5$ a buying session. However, a better approach is rather to classify the session in light of Equation (2). For simplicity, assume that the session contains only one item i, so we should either predict $A_s = \emptyset$ with associated score 0 or $A_s = \{i\}$, with score either $1+\nu$ (if the session really ended with a buy) or $-\nu$ (if it did not). It is a buying session with probability p_s, and calculating the expected score for each of the decisions, we find that it is beneficial to select $A_s = \{i\}$ if and only if $p_s \cdot (1+\nu) + (1-p_s)\cdot(-\nu) > 0$, i.e., when $p_s > \nu/(1+2\nu) \approx .0496$. This is therefore the threshold for the decision rule, which we applied to classify all sessions containing only a single item. For sessions containing $\kappa > 1$ items, the expected score (and therefore the decision rule for the "Buy-or-not" classifier) will depend on the fraction $|A_s \cap B_s|/|A_s \cup B_s|$, which is again determined by the quality of the "What-to-buy" classifier. In these cases we defined the decision rule by approximating the fraction by the average obtained from all sessions in the training data containing exactly κ items.

3.2 The "What-to-buy" classifier

It is not possible to select items in session s to directly maximize the value $|A_s \cap B_s|/|A_s \cup B_s|$, because the set B_s of the actual buy-session is not available in the test-set. However, we can still estimate the maximal expected value for the fraction, thus giving a way to come up with the final set of items assumed to be purchased during a buying-session.

The classifier addressing "What-to-buy" generates probabilities for each item in the session to be purchased separately. Thus, if a session looked at items $\{i_1, i_2, \ldots, i_n\}$ the classifier was trained to use the supplied features to calculate the probabilities $\{p_1, p_2, \ldots, p_n\}$. Assuming that the items $\{i_1, i_2, \ldots, i_n\}$ are sorted in decreasing order of their p-values, we can then calculate the expected value for the fraction when making each of the decision $A_s^{(1)} = \{i_1\}$, $A_s^{(2)} = \{i_1, i_2\}$, up to $A_s^{(n)} = \{i_1, \ldots, i_n\}$, choosing the one with the highest value. As the "Buy-or-not" classifier has already determined that the session ended in a purchase, we

ensured that at least one item is predicted as bought. That is, we did not include $A_s^{(0)} = \emptyset$ as an alternative outcome from this classifier.

Note that, with these proposed decision rules for the "Buy-or-not" and the "What-to-buy" classifiers, the performance of our method was significantly improved. The details will be given in the next section.

4. EXPERIMENT AND RESULTS

There is a limited number of times per day a participant can upload a solution to the challenge website. Therefore, we created an offline testing environment where we evaluated different classification algorithms, parameters and the extracted features. Since the training data is highly imbalanced, we applied undersampling to randomly select a subset of the original data where the number of buying and and non-buying sessions were approximately equal. Accuracy measurements (precision, recall and ROC) and the RecSys Challenge's score were used to guide model section processes. It is not guaranteed that the best algorithm chosen from the offline testing will provide best online results, and we therefore used online testing whenever this seemed necessary.

We used the entire training data to extract the sets of features for both classification tasks. Regarding the probability of an item being bought (reflected in the feature F5, see Section 2), our analysis showed that the value depends on the time when the session occurred. As an example, Figure 4 shows how the percent of buy-sessions varies from day to day. Therefore, to compute the probability we looked at the day when the click occurred, as well as in the surrounding days. The problem was then to select a number of sessions that must have clicked on the item before accepting the probability. For this, we tested 11 different values: 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55. It turned out that the offline and online testing got the best performances when the threshold was set to 25 to calculate the probability of the item being bought.

As mentioned in Section 3, we evaluated different probabilistic classifiers because their outputs can be used to optimize our decision on the final solution wrt. the RecSys Challenge's score. The algorithms employed in our experiment were Logistic Regression, Naive Bayes, Bayesian Networks, Decision Trees and Random Forest. It turned out that the Random Forest [2] worked best for offline and online testing for both classification tasks. Several parameters of the Random Forest were evaluated and we observed that the following parameters are the best settings in term of the obtained accuracies and the score:

Parameters	What-to-buy	Buy-or-not
max_depth	∞	∞
min_samples_split	350	200

Here *min_samples_split* is the minimum number of samples that have to be present within an internal node in a tree in order to split it into new nodes and *max_depth* indicates how deep a tree is allowed to be build. "∞" signifies that the tree will be built until all leaf nodes contain samples of the same class label or until all leaf nodes contain less samples than the value set in the *min_samples_split*.

As shown in Figure 4, the user behaviour varies over the days, for instance, the buy behaviour on Wednesday was not the same as on Saturday. Therefore, we decided to build several models for each time period. This time period should be as small as possible to catch user behaviour in that period precisely, but we still have to have enough data to build reliable classifiers. We experimented with several time periods: one day, one week, two weeks and one month. Data undersampling was carried out during each time period, both for training and validation data. The best performance was obtained when the time period was set to two weeks.

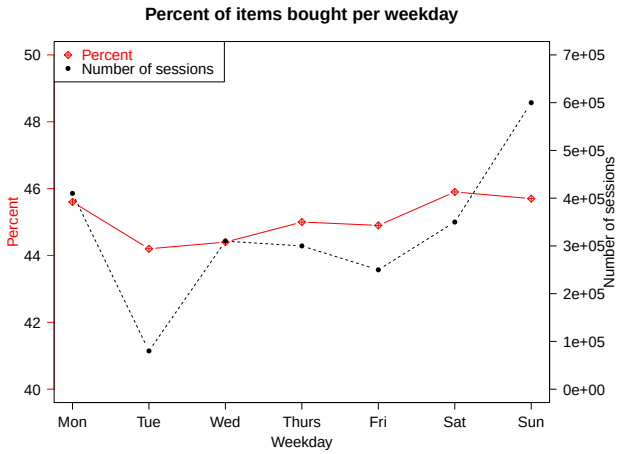

Figure 4: Percent of sessions buying, per weekday (red and solid line, left y-axis) and the number of sessions where this happens (black and dashed line, right y-axis).

The effect of optimizing the decision rules towards the score compared to classifying each session to the most probable class (see Section 3.1 and Section 3.2) was quantified by submitting two solutions: one naïvely following the original outputs from the Random Forest classifiers and the other applying the decision rules. We observed that the score was increased 15% with the latter approach.

5. CONCLUSIONS AND FUTURE WORK

We have described the contributions from the Tøyvind thørrud team, which utilizes probability-based features and two probabilistic classifiers learned in sequence. During learning, the classifiers were optimized with respect to classification accuracy, which necessitates rather intricate decision rules being employed (see Section 3). Alternatively, classifiers could also be optimized wrt. the *score* directly. Additionally, other ensemble methods (both combining different algorithms and using, e.g., random sub-space methods) could have been employed, and the list of features could have been extended with statistical features like (time-dependent) mean or median values. Our proposed solution is not a direct application for recommendation systems. However, we can use the relevant features to infer implicit rankings of user preferences for a learning-to-rank-based recommendation system.

6. REFERENCES

[1] D. Ben-Shimon, A. Tsikinovsky, M. Friedmann, B. Shapira, L. Rokach, and J. Hoerle. Recsys challenge 2015 and the yoochoose dataset. *In Proceedings of the 9th ACM conference on Recommender systems. ACM.*

[2] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, Oct. 2001.