Detecting Search Engine Switching Based on User Preferences, Search Tasks, and Behavior Patterns

Denis Savenkov, Dmitry Lagun, Qiaoling Liu
Mathematics & Computer Science
Emory University
Atlanta, US
{denis.savenkov, dlagun, qiaoling.liu}@emory.edu

ABSTRACT

In this paper, we present our approach to the 2012 Yandex Switching Detection Challenge, which achieved the first place among all participants. Our approach built on multiple gradient boosting tree models predicting search engine switching based on more than 400 features derived from user preferences, search tasks, and user behavior patterns. There are four key aspects in our method: 1) Rich features are designed not only to cover different switching scenarios such as user preference (user-based features) and specific search task (query- and url-based features), but also to learn search failure from user behavior (session- and action-sequence-based features); 2) To better capture switching signals, we use datasets that are disjoint from the training datasets to compute rich aggregate features over (user specific) switch sessions and non-switch sessions; 3) To better deal with the imbalanced data, we tried both the cost-sensitive technique and the subsampling technique; 4) To alleviate the overfitting problem, we trained multiple models and used the average score for the final prediction.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval

General Terms

Design, Experimentation

Keywords

search engine switching

1. INTRODUCTION

Search engines such as Google, Bing, Yandex, etc. facilitate access to the vast amount of information available on the Word Wide Web. Among many available search engines, users typically prefer to use one or two, performing majority of their searches on a primary search engine. Decision on which search engine to use as a primary search provider could be based on many factors including search quality, locale, satisfaction, usability of search interface and popularity of a search engine [5]. Sometimes users switch from one search engine to another during a search task. As vast majority of search engine switching cases (56%) arise from user's dissatisfaction with the search engine [5], information on when users switch provides a valuable signal for search providers to improve user search experience in such cases. Other switching rationales include need for verification or finding additional information, user preference and unintentional switches due to default settings of the internet browser. For some search sessions, the fact of switching can be easily monitored, for instance via a web browser (developed by a search engine), a browser toolbar or a navigational query for another search engine. However, in reality only a fraction of switches can be monitored. The same users might switch in a way which cannot be monitored by a search engine, e.g. they might use their browser bookmarks to switch the engines. Detecting the fact of switching, when switching events cannot be monitored directly is difficult, but it is important for understanding users' satisfaction with the search engine and the complexity of search. Yandex organized a Switching Detection Challenge[1] this year (Oct 23, 2012 - Dec 22, 2012). The task of the challenge was to predict for each given search session whether it contains a switching action anywhere in the session.

In this paper, we present our approach to the Yandex Switching Detection Challenge, which achieves the first place among the participants. Our approach built on multiple gradient boosting regression tree models for predicting search engine switching based on more than 400 features which are derived based on user preferences, search tasks, user behavior patterns, etc. There are four key aspects in our method:

- Rich features are designed not only to cover different switching causes such as user preference (user-based features) and specific search task (query- and url-based features), but also to learn search failure from user behavior (session- and action-sequence-based features); As shown by our experiments, all these different groups of features are useful to the switching detection task, and are complementary to each other.
- To better capture switching signals, we use datasets that are disjoint from the training datasets to compute rich aggregate features over (user specific) switch

[1] http://switchdetect.yandex.ru/en/

WSCD '2013 Rome, Italy

sessions and non-switch sessions. As shown by our experiments, these statistics (including personalized statistics) actually contain many of the most important features.

- To better deal with the imbalanced data, we tried two techniques [11]. The first one is to increase the weight (cost) of positive examples in the training set so that committing errors for the positive class becomes less likely as it is associated with higher cost. The second technique is to subsample negative examples to make a balanced dataset. As shown by our experiments, both techniques lead to performance improvements.
- To alleviate the overfitting problem, we trained multiple models over different time splits and used the average of scores obtained from each individual model for the final prediction. As shown by our experiments, the combined model achieves better performance.

The rest of the paper is organized as follows: Section 2 shortly discusses the related work and Section 3 describes the task, dataset and evaluation metric of the challenge. Next, we describe our approach in Section 4 (feature design), Section 5 (feature importance) and Section 6 (performance improvement). Finally, Section 7 concludes the paper.

2. RELATED WORK

Understanding and detecting search engine switching has recently attracted much research attention [10][12][17][5]. Previous work on detecting search engine switching behavior is done mostly in industry using proprietary data. The 2012 Yandex Switching Detection Challenge is the first one to share a public dataset for this prediction task, which allows academic researchers to experiment with naturalistic large scale log data. However, in order to avoid privacy issues the data has to be anonymized prior to its public release. Thus, only numeric identifiers of queries and URLs are available and can be used for analysis and deriving features. As per challenge guidelines we aim at offline prediction of user switching using complete session information instead of more difficult online scenario where only partial session information is available at the time of prediction [17].

The related work is primarily focused on understanding user behavior prior to switching and predicting searcher's success. Heath et al. [10] have analyzed search engine logs and mined frequent patters of searcher behavior in the search session represented as a sequence of possible actions. Their alphabet of actions included query issued (Q), clicked result link (S), clicked non result link, usage of browser back button, pagination and search engine switching. They have also encoded types of page pages user visits - results page or other page, with associated time duration - short, medium and long. Laxmant et al. [12] use Hidden Markov Model (HMM) to analyze searcher behavor and identify frequent predictive subsequences. Closest to our paper in spirit are the works of R.White and Q.Guo et al. [17], [5]. As one of the most popular switching rationale is dissatisfaction with search engine, another relevant research area is predicting searcher success. Not exhaustive list of relevant papers include [7][1][4][8][9].

Predicting query difficulty [3] [2] [6] as part of search success is relevant as well, in particular in part of query features described in later sections.

3. TASK DESCRIPTION

3.1 Task and Dataset

The goal of the challenge is to predict switches that cannot be monitored in a standard way. Standard ways to monitor switching actions include detecting navigational queries that users type in an old search engine to open a new one (e.g. a user might type a query "google" in Yandex in order to switch to Google), detecting clicks on the "search with another search engine" links which Yandex has at the bottom of its SERP (Search Engine Results Page), and detecting direct search engine switches using Yandex's browser toolbar. Unfortunately a great number of switches cannot be monitored by the above mentioned means but the knowledge that the users switched search engines in these sessions might help better understand user satisfaction and the complexity of search; therefore, it is important to identify such unmonitored switch sessions.

The dataset released by Yandex includes search sessions extracted from Yandex logs[2], with user ids, queries, URL rankings, clicks and search engine switching actions. To allay privacy concerns the user data is fully anonymized. So, only meaningless numeric IDs of users, queries, sessions, and URLs are released. In total, the dataset includes 8,595,731 sessions, 10,139,547 unique queries, 49,029,185 unique URLs, and 956,536 unique users.

More specifically, two files are given in the dataset: train and test. Yandex took all users that have done at least one switch in a period of 27 days and collected all their sessions in this period as the train file. The test file consisted of the sessions of these users for the next 3 days with switch actions removed.

The task of the challenge is to predict for each of the test sessions how likely the user switched to another search engine during the search task.

3.2 Quality metric

All submissions are evaluated using the AUC (Area Under Curve) measure [13]. AUC is a popular measure used for evaluation of predictors and represents the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.

If the results of the classifier are presented as a ranked list based on the beliefs that each instance belongs to the positive class, then we can use a simple formula (from [13]) to calculate AUC:

$$AUC = \frac{S_0 - n_0(n_0+1)/2}{n_0 n_1}$$

where $S_0 = \sum r_i$, r_i are the ranks of truly positive examples in the results list, and n_0, n_1 are the number of positive and negative examples respectively. AUC was shown to be a more consistent and discriminating measure than accuracy.

4. OUR APPROACH

Since the task of the challenge was to predict how likely each test session contains a switching action we decided to solve this as a binary classification problem. The idea was to train a classifier which outputs a confidence that a session

[2]The logs are about 1.5 years old and do not contain sessions with queries that have commercial intent detected with Yandex's proprietary query classifier.

Table 1: Statistics of the evaluation sets created for our experiment

	#Sessions	%Pos
Official train file (d1-27)	7,856,734	18.6%
Official test file (d28-30)	738,997	unknown
Split 8 (d1-24 for statistics)	6,769,082	18.2%
Split 8 (d25-27 for validation)	737,421	10.7%
Split 7 (d1-21 for statistics)	5,676,130	18.1%
Split 7 (d22-24 for training)	556,502	10.0%
Split 6 (d1-18,22-24 for stats)	5,985,972	18.3%
Split 6 (d19-21 for training)	542,917	10.4%
...	...	...
Split 0 (d4-24 for statistics)	5,871,491	17.5%
Split 0 (d1-3 for training)	538,506	9.3%

contains a switch and then to rank all sessions by this value to produce the final list.

4.1 Evaluation set creation

First of all, we needed to set up a reasonable evaluation environment so as to evaluate the performance of our classification models. Therefore, we held out the sessions in day 25-27 as our validation set, and used the sessions in day 1-24 to create training sets as follows. First, we splitted the sessions into two sets: a statistics set according to a statistics period (e.g., d1-21) and a training set according to a training period (e.g., d22-24). Then, we removed from the training set sessions by users who never switched during the statistics period[3]. We also applied this filtering step to the validation set. Therefore, the process of creating our evaluation set is similar to how the official train and test files were created. Table 1 shows the statistics of the multiple data splits we created. One can see that they have very similar distribution as the official train and test files. Based on our experimental results, the performance of classification models on this evaluation set is in general consistent with their performance on the Leaderboard test set.

In the following sections of the paper, unless explicitly stated the statistics/results are computed using the dataset of Split 7 in Table 1.

4.2 Feature description

As suggested by the previous research [5] users typically switch because they are not fully satisfied with the current search results, they need more relevant results or they assume the other search engine is better for the given task. Therefore, the probability that user switches to different search engine during his search session depends on user's personal habits and the search task. This means that some users switch more often than the others and for some tasks switching is more likely to happen than for the others.

Inspired by the previous effort on search engine switching prediction [10][12][17][5], we developed a broad set of features, which can be splitted into 3 groups: features based on the current session only, features based on statistics computed from all switch/non-switch sessions in the statistics period and features based on user-specific switch/non-switch sessions in the statistics period. Our intuition was that using aggregated statistics over all switch/non-switch sessions on a separate period should help an algorithm to distinguish between switch and non-switch sessions on a training set. In addition, statistics calculated for each user separately represents some information on personalized user behavior patterns and may help to train a better model, but on the other hand this data is very sparse (most of the users in the dataset have just a few sessions and statistics may not be that trustworthy).

The final feature set we used for our best submission includes 414 features. Some of them are very similar, only with different smoothing, normalization or other variations. Table 5 provides a summary of these features (not a complete list). The Python code generating the features is available online [4].

4.2.1 Session Features

Session features include features that only depend on the current session, e.g., session duration (in time units and in actions), number of unique and abandoned queries (queries without clicks), average/min/max click dwell time and pause between actions (pause between issuing a query and the next action was also considered), average click position, number of SAT/DSAT clicks, and some more. By SAT (DSAT) clicks we considered clicks with dwell time more than 500 (less than 200) time units.

4.2.2 (User-Specific) Statistics Features

The rest of the features are calculated using the statistics dataset. They can be grouped into three subsets. The first subset of statistics features include all the above session features normalized by the corresponding average statistics calculated across each user's switch/non-switch sessions as well as across all users' switch/non-switch sessions. This gives us 4 features with different normalization. For example, we calculated the average duration of all switch/non-switch sessions in the statistics period separately, and included as two features the session duration divided by the two average values separately. Another two features were generated by repeating the same operations for the two average values calculated for the current user. In addition to that we also included user's personalized average statistics as separate features which could allow a learning algorithm to learn more complicated dependencies than just normalization.

The second subset of statistics features include the switching probability of a user/query/url. Different users have different probability of switching. The same is true for different search tasks. A user's switch probability was calculated as $(\#user\_switches + 1)/(\#user\_sessions + 10)$. To include search task related switch probability we considered queries and urls separately. For each query we calculated the number of times the query was issued before switch in a session. There are two cases: we can include all queries from the beginning of the session to the last switch in a session, or just queries issued immediately before switching. We used both cases for the feature generation. The same approach was takes for urls, where only clicked urls were considered.

The third subset of statistics features was computed based on action sequences, sometimes called search trails. Each action sequence of a session was encoded in three ways, denoted as type-I, type-II and type-III sequences respectively.

[3]We forgot to remove such sessions from the statistics set by mistake.

[4]http://mathcs.emory.edu/~dsavenk/switch_detect/

Table 2: Markov model transition probabilities

Non-switch sessions

action	q	K	Q	D	P	S
q	0.169	0.006	0.014	0.272	0.099	0.440
K	0.438	0.038	0.066	0.135	0.064	0.259
Q	0.579	0.038	0.091	0.078	0.032	0.182
D	0.168	0.005	0.019	0.399	0.096	0.313
P	0.327	0.012	0.039	0.228	0.105	0.289
S	0.439	0.016	0.066	0.131	0.064	0.284

Switch sessions

action	q	K	Q	D	P	S
q	0.212	0.016	0.040	0.323	0.115	0.294
K	0.532	0.069	0.129	0.097	0.041	0.132
Q	0.642	0.055	0.157	0.047	0.018	0.081
D	0.162	0.009	0.030	0.412	0.104	0.283
P	0.533	0.069	0.128	0.097	0.041	0.132
S	0.525	0.031	0.129	0.107	0.047	0.161

The type-I sequence alphabet includes queries (Q) and clicks (C) only. Considering that not all clicks are the same, to include click dwell time, in type-II sequences we used the following actions, similarly as those used in [7]: DSAT click (D), SAT click (S) and other (P), where SAT/DSAT clicks are defined in the same way as in Section 4.2.1. So, the alphabet for the type-II sequences is {Q,D,P,S}. The last type (type-III) of the sequences took into consideration different lengths of pause between issuing a query and the next action in a session. Using the same thresholds as for short and long dwell time we assigned different symbols to queries with small pause (q), long pause (Q) and others (K). The alphabet for this type of sequences is {q,K,Q,D,P,S}.

For each of the three types of sequences, we generated the following features based on the statistics dataset: probability of this sequence appearing in switch/non-switch sessions, probabilities of this sequence under the Markov model with transition probabilities estimated on switch/non-switch sessions, and finally statistics based on all n-grams (2,3,4-grams) of this sequence, namely average ratio of their frequencies in switch and non-switch sessions[5]. Table 2 provides the estimated Markov model transition probabilities for type-III sequences. As we can see in switch sessions transitions from other actions to S (SAT click) are less likely while transitions to queries (q, K, Q) are more likely than in non-switch sessions. Table 3 shows some 3-grams over type-III sequences ranked by the ratio of their frequencies in switch and non-switch sessions. We can see that 3-grams with Q (queries with long pause) tend to occur more frequently in switch sessions, and on the contrary 3-grams with S (SAT clicks) are indicators of non-switch sessions. An intuitive but nevertheless interesting finding is that 3 queries with small pause (qqq) has almost the same ratio as 3 DSAT clicks in a row (DDD) and the ratio is much less than for queries with long pause (e.g, QQQ).

4.2.3 Hidden-state Condition Random Fields

So far we have focused on feature aggregation at multiple levels in order to derive good features for the whole session. However, a more principle way of capturing sequence information is to use a sequence model that naturally incorporates information from the sequences of user action . Among many existing sequence models we have chosen conditional random field with latent variables (HCRF [15]) for two reasons: (i) it allows discriminative training and (ii) the hidden layer of variables conditions the entire sequence on single a label, i.e. whether user switched or not. Figure 1 shows an example graphical model of HCRF. The model has three types of variables: x_i represents features for the i-th action, h_i represents the hidden variable for i-th action and y is the label for the entire sequence. In our implementation actions correspond to either queries or clicks. We trained the HCRF model by optimizing the log-likelihood on training data with respect to model parameters, i.e., weights defining node potentials, as well as pairwise potentials between hidden variables and sequence label y (for more details see [15]). As there is no closed formed solution optimizing the log-likelihood, the training is performed iteratively with gradient based methods such as LBFGS. Because of this reason we limited the number of features to 49 to fit the model. Table 4 summarizes the features used for training HCRF.

Table 3: 3-grams ranked by the ratio of frequencies to appear in switch ans non-switch sessions

3-gram	Ratio	3-gram	Ratio
QKQ	1.6332	qSS	0.2096
KQQ	1.5983	SSS	0.2107
QQQ	1.5895	SSP	0.2449
QQK	1.5259	SPS	0.2568
KQK	1.4143	SSD	0.2649
KKQ	1.4041	PSS	0.2697
QKK	1.3872	qqS	0.2739
qQQ	1.2804	DqS	0.2786
QQq	1.2591	SDS	0.2809
QqQ	1.2502	KSS	0.2829
...	...	...	...
qqq	0.4384	DDD	0.4194

Table 4: Features used in HCRF model

time_pause - time elapsed from previous action *time_to_action* - time elapsed from the start of the session *queries_before* - number of queries before this action *clicks_before* - number of clicks before this action *is_query* - 1 if this action is a query, 0 if it is a click
is_repeated_query - 1 if this query repeats in the session *query_freq* - query frequency *query_switch_prob* - switch probability for this query *urls_switch_prob* - switch probability for each result of this query (10 features) *urls_ctr* - CTR for each result of this query (10 features) *urls_clicked_positions* - 1 if a result of this query displayed on a given position receives a click (10 features)
url_ctr - CTR for the result receiving this click *url_clicked_positions* - 1 if this click is on the i-th result (10 features)

4.3 Learning algorithm

For our experiments we tried 2 different implementations of the gradient boosting tree algorithm: pGBRT[6] [16] and

[5]Unfortunately at the time of the challenge we had a bug in the feature generation script which resulted in incorrect values for all n-gram features in our submissions.

[6]http://machinelearning.wustl.edu/pmwiki.php/Main/Pgbrt

Table 5: Features used in switch prediction

Features based on current session only
q_count - number of queries
c_count - number of clicks
time_to_1click - time to first click in a session (or large value if no clicks)
q_abandoned - number of abandoned queries
dwell - average, max, min clicks dwell times
duration - session duration (in time units and in actions - queries and clicks)
pause - mean, min, max pause between actions in session
unique_queries - number of unique queries issued in a session
(d)sat_clicks - number of sat/dsat clicks
ave_click_pos - average click position (11 if no clicks)
time_between_clicks - average time between clicks
last_action - is the last action a query or a click
Features based on all switch/non-switch sessions
all above features normalized by the corresponding average statistics calculated across all switch/non-switch sessions
q_switch_freq - frequency of a query occuring before switch in a session (max, mean, min over queries in a session)
q_ctr - ctr (sat, dsat, last) of a query (max, mean, min over queries)
q_ave_clickpos - average click position for a query (max, mean, min over queries)
q_freq - query frequency (max, mean, min over queries)
clicked_url_ctr - ctr (sat, dsat, last) of a clicked url (max, mean, min over all clicked urls in a session)
url_switch_freq - frequency of a clicked url apperaring before switch in a session (max, mean, min over all clicked urls)
markov_models - probability of the action sequence by Markov models estimated on switch/non-switch sessions
n-grams - average ratio of frequencies of the sequence n-grams in switch and non-switch sessions
seq_prob - probability of this exact action sequence appearing in switch/non-switch sessions
crf - conditional random fields model prediction for the session
Features based on user-specific switch/non-switch sessions
all above features with statistics calculated on this user's switch/non-switch sessions instead of on all users' sessions
all average statistics of session features computed over this user's switch/non-switch sessions
user_switch_prob - number of user sessions with a switch + 1 divided by the total number of user sessions + 10
user_switch_prob_periods - similar to *user_switch_prob* but computed for all the 3-day periods within the statistics period separately
user_last(middle,toolbar,serp)_switch_prob - number of user sessions with a switch (last/middle in session or of toolbar/serp type) divided by the total number of user sessions
user_session_count - number of sessions of this user in the statistics period
ave_time_to_switch - average time to switch for this user

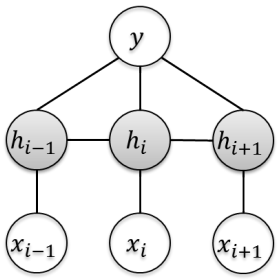

Figure 1: Hidden Variable Conditional Random Field Model

scikit-learn[7] [14]. The first implementation is written in C++ and uses MPI for parallel computations. By default RMSE loss function is optimized. The second implementation is a Python module which has a variety of tuning parameters, e.g. loss function, subsampling rate (each new stage uses only a fraction of examples), maximum number of features to use on each iterations and some more. In our experiments we used logistic loss function.

[7]http://scikit-learn.org/

In the experiments we made the scikit-learn implementation achieved better performance (0.8572 AUC on training and 0.8440 AUC on validation, compared to 0.8473 and 0.8403 AUC scores correspondingly for pGBRT). Modifying pGBRT to use logistic loss didn't improve the score. Unfortunately, it takes about 6 hours to train a model with scikit-learn, compared to 30 minutes with pGBRT, thus we were unable to use scikit-learn for our final submissions.

5. FEATURE IMPORTANCE

5.1 Individual feature importance

Table 6 presents importance scores for some of the features. We chose two scores: information gain and Gini index based score calculated on the boosting trees ensemble (denoted as boosting score). Only a couple of features from each group are presented in the table. The ranks of the features in the list are computed by the boosting score.

The analysis shows that the most important signals for switch detection are sequence n-gram statistics, user switching frequency, average click position. The average click position in a session is ranked 3rd by the boosting score but much worse (123th) by the information gain. This feature can be

an indicator of both difficult search tasks and tasks requiring more coverage. Sequence features placed at the top of the ranked list, suggesting that Markov models, whole sequence frequencies and n-gram statistics are all useful. Comparing the three types of sequences, we found that features calculated based on type-III sequences are ranked best, which means that encoding different dimensions of time information (i.e., distinguishing different lengths of pause after a query and of pause after a click separately) is useful. It was surprising that url switch statistics (probability that a session contains a switch after a click on the given url) are more important than query switching statistics. The possible explanation is that some urls "suggest" users to try another search engine.

Only 291 out of 414 features (70%) have non-zero boosting scores, which means that the learning algorithm didn't use the rest of the features (e.g., day of week features, most of the query and url ctrs). This is not a surprise as our feature set has a variety of similar features with different normalization or smoothing.

5.2 Feature group importance

To investigate how features interact with each other, we divided features into groups and analyzed the prediction performance by each feature group. Two ways of feature grouping were considered. The first way included three groups as shown in Table 5: 1) features based on current session only; 2) features based on all switch/non-switch sessions; and 3) features based on user-specific switch/non-switch sessions.

A special interest is to see which of the following is the most important: frequency of switches for users, queries, urls or sequences. Therefore, we considered another way to group features into five disjoint sets: 1) features based on session statistics[8] (duration, number of queries, number of clicks, etc); 2) features based on user statistics[9] (user switch probability and some modification); 3) features based on query statistics (frequency of switches after a given query, ctr of queries in a session, etc); 4) features based on url statistics (frequency of switches after a given url is clicked in a session, ctr of clicked urls in a session, etc); and 5) features based on action sequence statistics (Markov model probability, n-gram statistics, etc).

Based on the above two ways of feature grouping, we conducted a series of feature ablation experiments. Table 7 shows the prediction performance when providing a single feature group or when providing all feature groups but this feature group. Among the first set of feature groups, overall statistics features alone are more effective than session features alone, which verifies our intuition that aggregated statistics on a separate period helps with switch prediction. Moreover, user-specific statistics features gave even more signals for predicting switches. This single feature group run achieved 98.1% of the validation AUC obtained by training on all features. And removal of this feature group resulted in a significant drop in prediction quality (from 0.8413 to 0.7782). One reason which makes user-specific statistics features so effective could be attributed to how the dataset was generated: all users from the validation set were present in the training set and had at least one switch there.

[8]Note that session statistics features here included all the normalized versions by the corresponding average statistics computed in the statistics period.

[9]Note that user statistics features here did not include those relying on queries, urls, or action sequences in the session.

Table 7: Feature ablation experiments

Single feature group runs

Feature group	Training AUC	Validation AUC
Session features	0.7563	0.7491
Overall statistics features	0.7853	0.7777
User-specific statistics features	0.8381	0.8253
Session statistics features	0.8298	0.8183
User statistics features	0.7503	0.7273
Query statistics features	0.6480	0.6352
Url statistics features	0.7488	0.7396
Sequence statistics features	0.8062	0.7952
All features	0.8534	0.8413

Without feature group runs

Feature group	Training AUC	Validation AUC
Session features	0.8532	0.8326
Overall statistics features	0.8371	0.8290
User-specific statistics features	0.7882	0.7782
Session statistics features	0.8453	0.8348
User statistics features	0.8490	0.8289
Query statistics features	0.8529	0.8380
Url statistics features	0.8466	0.8273
Sequence statistics features	0.8517	0.8397
All features	0.8534	0.8413

Among the second set of feature groups, the most effective ones are session statistics features and sequence features. Here we also see that url switch statistics features alone (frequency of switches occurred after a click on a given url) obtained a little better AUC than user statistics features alone (e.g. user switching frequency). It would be interesting to see, what urls are "triggering" search engine switching behavior. Query statistic features alone resulted in significantly worse performance in predicting switches. On the other hand, results of runs with a feature group removed are very close to each other, indicating that signals in each of the five feature groups could be covered to a certain extent by other four feature groups. Nevertheless, all the five groups of features are useful to the switching detection task. In particular, when sequence features were removed the prediction quality did not drop much, despite the fact that features from this group have high importance according to information gain and boosting Gini index scores in Table 6. This shows that sequence features contain strong and unique signals for switch detection, but also many duplicate signals from session, user, query, and url based statistics features.

6. EXPERIMENTS

6.1 Handling imbalanced datasets

As you can see from Table 1 the training set is imbalanced, only 10% of the examples contain switches and the rest are sessions without switch. One way to take this into account is to increase the number of positive examples (session with switches), we can do this for example by increasing the weight of the positive examples during training. Another approach is to subsample negative examples (sessions without switches). We were aware that sessions in negative class are more noisy because they might still contain

Table 6: Feature importance scores (ranks are computed based on the boosting score)

Rank	Feature	Information gain	Boosting score
1	*user_typeIII_2gram* - average ratio of type-III sequence 2-gram frequencies in user's switch and non-switch sessions	0.0350	0.0230
2	*user_switches_count* - total number of switches in the user sessions in statistics period	0.0194	0.0222
3	*ave_click_pos* - average position of clicks in a session	0.0149	0.0219
4	*user_switch_prob* - smoothed frequency of switches in user sessions in statistics period	0.0309	0.0184
11	*typeIII_2gram* - average ratio of type-III sequence 2-gram frequencies in all switch and non-switch sessions	0.0290	0.0134
12	*time_to_1click_user_normalized* - time to the first click in a session normalized by average time to first click in user non-switch sessions	0.0103	0.0129
14	*abandoned_to_ave_abandoned* - number of abandoned queries normalized by average number of abandoned queries in user's non-switch sessions	0.0229	0.0114
15	*unique queries* - number of unique queries in a session	0.0204	0.0114
21	*ave_clickedurl_switchprob* - average frequency of switches after clicking the given url (score is averaged over all clicked urls in the session)	0.0100	0.0103
22	*min_pause_user_switch* - minimum pause in a session normalized by average minimum pause in user's switch sessions	0.0016	0.0101
27	*ave_time_to_switch* - average time before switch in user's sessions	0.0026	0.0092
30	*time_1click* - time to first click in a session	0.0078	0.0082
39	*crf* - conditional random field score for the session	0.0191	0.0070
42	*sat_clicks* - number of sat clicks in a session	0.0046	0.0069
51	*user_ave_query_switchprob* - average frequency of the user to switch after typing a given query (the score is averaged over all queries in the session)	0.0013	0.0059
64	*session_duration* - duration of the session in time units	0.0182	0.0050
291	*query_nonswitchprob* - average frequency of a query to be issued in a non-switch session (score is averaged over all queries in a session)	0.0001	0.0000

switches not detected by standard means. The results for both approaches are summarized in Table 8. As you can see both strategies show some performance improvement over training on the imbalanced dataset. The differences in performances are small, but the results are consistent over different splits and experiments we did. In addition, subsampling resulted in a smaller size of the dataset (e.g. 110981 vs 556502 examples for split 7) and training was faster without loss of prediction quality. For example, on our server training of pGBRT on a whole split of dataset lasted about 30 minutes, compared to 3 minutes on a subsampled dataset.

Table 8: Experiments results (dataset balancing, model averaging)

	Train AUC	Validation AUC
No balancing	0.8473	0.8403
Pos weight = 4	0.8534	0.8413
Subsampling	0.8490	0.8412
Split 7 only (21-24)	0.8534	0.8413
best individual model	0.8568	0.8426
worst individual model	0.8546	0.8410
Averaging 8 models	-	0.8450

6.2 Multiple models averaging

To reduce variance of the trained models we decided to use a strategy similar to bagging, but instead of sampling from the training set we used different time splits of the original dataset. As shown in Table 1, since only 3-day period was used for training (and the rest days are used for aggregated statistics) we could produce all different training sets, train model on each of them, make predictions for a test session and obtain the final result by averaging individual scores. Table 8 summarizes results of this experiment. As you can see the averaged model outperformed all individual models.

Using this best model computed by averaging multiple models, we plot the precision and recall curve for the positive class (switch sessions) in Figure 2. It shows that for this switch prediction task, high precision (e.g., 0.78) could be achieved at the cost of recall (e.g., 0.1). Moreover, precision decreases roughly linearly with the increase of recall.

6.3 Our best submission

For our best submission, all 414 features described above were collected for different splits (Table 1), and we used the pGBRT implementation of gradient tree boosting with the RMSE loss function. Parameters were fixed at 400 iterations with tree depth equal to 5 and learning rate equal to 0.1. We used a combination of classifiers trained on different periods as it was shown to give some performance improvement. The final submission was based on averaging predictions of models trained on splits 3 (d10-12), 6 (d19-21), 7 (d22-24), 8 (d25-27). Please note, that for the final submission the validation dataset (d25-27) was used for training as well as for statistics when training on other splits. For a bunch of reasons we used a mixed balancing strategy: models trained on splits 7 and 8 were trained with weight of positive class set to 4, but models 3 and 6 were trained on subsampled datasets. As a final score we used the average of predictions obtained from each individual model.

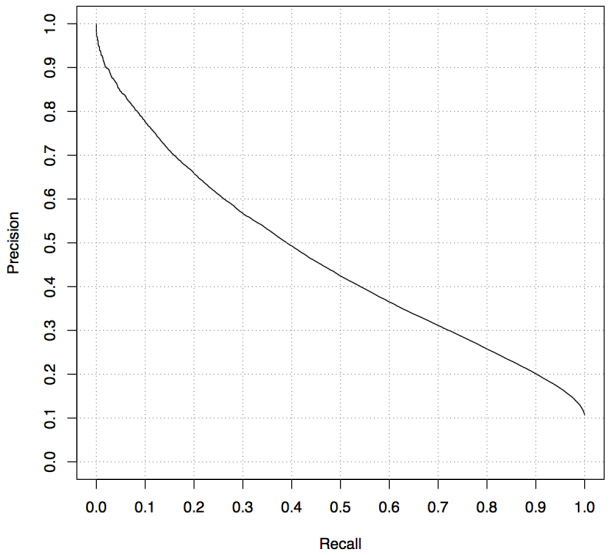

Figure 2: Precision-Recall curve for the positive class (switch sessions)

7. CONCLUSIONS

In this paper we presented a machine learning based approach to web search engine user switching detection. Our approach utilizes several kinds of personalized and aggregated statistics, collected on a period separate from the training set. All the features were designed to help learning different user switching habits and behavior patterns as well as different reasons for switching (i.e. dissatisfaction and necessity in better coverage). We demonstrated that personalized statistics and session feature normalization allow to improve switching prediction quality significantly. Modeling action sequences is an important source of information about switching behavior. We showed that n-gram model features are ranked high by both information gain and Gini index scores. Incorporating time information into actions (i.e. queries with short/long pauses before the next action) has shown to be very useful for switching prediction as well.

Our system scored first among all participants of the Yandex Switching Detection Challenge[10].

8. ACKNOWLEDGMENTS

We would like to thank Mikhail Ageev and Yu Wang for their discussions. We would also like to thank Dr. Eugene Agichtein for his support of this work.

9. REFERENCES

[1] M. Ageev, Q. Guo, D. Lagun, and E. Agichtein. Find It If You Can: A Game for Modeling Different Types of Web Search Success Using Interaction Data. In *Proc. of SIGIR'11*, pages 345–354, New York, New York, USA, July 2011. ACM Press.

[2] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In *Proc. of SIGIR'06*, pages 390–397, New York, NY, USA, 2006. ACM.

[3] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proc. of SIGIR'02*, pages 299–306, New York, NY, USA, 2002. ACM.

[4] H. A. Feild, J. Allan, and R. Jones. Predicting searcher frustration. In *Proc. of SIGIR'10*, pages 34–41, New York, New York, USA, July 2010. ACM Press.

[5] Q. Guo, R. White, Y. Zhang, B. Anderson, and S. Dumais. Why searchers switch: understanding and predicting engine switching rationales. In *Proc. of SIGIR'11*, pages 335–344, 2011.

[6] Q. Guo, R. W. White, S. T. Dumais, J. Wang, and B. Anderson. Predicting query performance using query, result, and user interaction features. In *Adaptivity, Personalization and Fusion of Heterogeneous Information*, RIAO '10, pages 198–201, Paris, France, France, 2010.

[7] A. Hassan. A semi-supervised approach to modeling web search satisfaction. In *Proc. of SIGIR'12*, pages 275–284, New York, NY, USA, 2012. ACM.

[8] A. Hassan, R. Jones, and K. L. Klinkner. Beyond DCG: User Behavior as a Predictor of a Successful Search. In *Proc. of WSDM'10*, pages 221–230, New York, New York, USA, Feb. 2010. ACM Press.

[9] A. Hassan, Y. Song, and L.-w. He. A task level metric for measuring web search satisfaction and its application on improving relevance estimation. In *Proc. of CIKM'11*, pages 125–134, New York, New York, USA, Oct. 2011. ACM Press.

[10] A. Heath and R. White. Defection detection: Predicting search engine switching. In *Proc. of WWW'08*, pages 1173–1174, 2008.

[11] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449, Oct. 2002.

[12] S. Laxman, V. Tankasali, and R. W. White. Stream prediction using a generative model based on frequent episodes in event sequences. In *Proc. of SIGKDD'08*, pages 453–461, New York, New York, USA, Aug. 2008. ACM Press.

[13] C. Ling, J. Huang, and H. Zhang. Auc: a statistically consistent and more discriminating measure than accuracy. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 519–526, 2003.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[15] A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1848–1852, 2007.

[16] S. Tyree, K. Weinberger, K. Agrawal, and J. Paykin. Parallel boosted regression trees for web search ranking. In *Proc. of WWW'11*, pages 387–396. ACM.

[17] R. White and S. Dumais. Characterizing and predicting search engine switching behavior. In *Proc. of CIKM'09*, pages 87–96. ACM, 2009.

[10] http://switchdetect.yandex.ru/en/results