Intent-Based Browse Activity Segmentation

Yury Ustinovskiy, Anna Mazur, and Pavel Serdyukov

Yandex
{yuraust,amfolity,pavser}@yandex-team.ru

Abstract. Users search and browse activity mined with special toolbars is known to provide diverse valuable information for the search engine. In particular, it helps to understand information need of a searcher, her personal preferences, context of the topic she is currently interested in. Most of the previous studies on the topic either considered the whole user activity for a fixed period of time or divided it relying on some predefined inactivity time-out. It helps to identify groups of web sites visited with the same information need. This paper addresses the problem of automatic segmentation of users browsing logs into logical segments. We propose a method for automatic division of their daily activity into intent-related parts. This segmentation advances the commonly used approaches. We propose several methods for browsing log partitioning and provide detailed study of their performance. We evaluate all algorithms and analyse contributions of various types of features.

1 Introduction

The classical approach to the information retrieval assumes that the user interacts with the search engine via a single query. This model favors the rapid development of various text retrieval methods which allow to construct rather effective retrieval systems. However, in the last decade it was realized that the performance of the classical approach is rather limited since it disregards many important aspects of search process, e.g. user personality, context of queries, pre-query actions. As a result, a number of recently proposed methods take into account all kinds of user-interaction data and lots of papers addressed to the problem of filling in the gap between classical approach and these new approaches. One of the most valuable sources of new data is represented by implicit user preferences expressed in various ways: her queries, clicks on search engine result page, pre- and post-query browsing trails [4,5,7,8,10].

As opposed to the text content of the web pages or standard link-graph of the Web, browsing data does not have any essential structure. That is why for further applications and studies this structure should be appropriately preprocessed and stored, so information extracted from it would become less noisy and more reliable.

In the current paper we focus on the problem of *browsing logs* processing and segmenting them into logically related parts (browsing logical sessions). The solution to this task tackles two different problems.

P. Serdyukov et al. (Eds.): ECIR 2013, LNCS 7814, pp. 242–253, 2013.

The first one was considered by several authors [1,3] and deals with automatic segmentation of *query logs*. It was mentioned in many studies that interaction with a search is a continuous process and restricting our attention to the sole query we lose valuable information about the context of the query. So consideration of a given query as a part of large search mission could significantly improve performance of the IR system. Since information derived even from simply-identified logical query sessions turned out to be quite useful, it is reasonable to improve quality of query segmentation. Exactly for this purpose query-flow-graph and hierarchical topic segmentation were developed in [1] and [3] respectively. Approaches proposed in these papers make it possible to understand how and when information need of a user changes, what exactly user has in mind addressing another query. As opposed to the previous studies, our method deals not only with search engine result pages but with the whole browsing logs. In particular, we simultaneously split both visited pages and queries into segments. So, the current research is in a sense continues these works: from the output of our algorithm we detect the switch of the information need on the browsed web-page navigational level instead of doing that at the coarse-grained search engine result page level.

The second problem concerns extraction of data from browsing logs [7,8]. Understanding the interaction nature of a user with her browser during a search process is crucial for further studies. To approach this problem researchers use the time-out to demarcate sessions in browsing logs and analyse obtained parts separately. In the literature such sessions are usually referred to as *browsing trails*. Such segmentation helps to realise how information need of a user drifts after a query, see why user comes to reformulation of a query, determine some common changes of search goals. Intent-based partition of the browsing log is essential for all these possible applications and browsing trails are just a rough approximation of ideal segmentation. Our framework allows to improve significantly the quality of common temporal grouping. To the best of our knowledge, this is the first paper addressing the problem of automatic intent-based browsing log segmentation.

To sum up, the main contributions of our paper are:

- Framework for automatic processing of browsing logs,
- Significant improvement of temporal segmentation, used in the most papers on browsing trails,
- Specification of query segmentation the web-site navigational level,
- Comprehensive analysis of various features and session extraction methods.

The rest of the paper is organized as follows. In Section 2 we discuss previous studies that are relevant to our research. In Section 3 we formulate our understanding of query and browsing logs and give several examples. In Section 4 the proposed algorithms for session segmentation are discussed. In Sections 5 and 6 we describe the data for the experimental part of our work and the results of these experiments. Finally, we conclude the paper in Section 7.

2 Related Work

Query and browsing logs were extensively studied in the last decade [11,14]. A large part of these studies is devoted to methods of division of URLs and queries into certain groups. The early studies used the idea of a temporal cut-off for session demarcation. Several studies addressed the problem of identification of the optimal time-out threshold, which mostly varies about 30 minutes [12,13].

In [1] Boldi et. al. introduced a *query-flow-graph* and proposed a framework for the automatic processing of the query log. The QFG is a graph, whose nodes are queries and edges connect intent-related queries. Authors divide the whole problem into several parts and solve them separately. They learn weights for the edges of graph, reorder original sequence of queries to cope with the interleaved sessions and finally use a tuned threshold to partition the reordered session. The first sub-problem is solved with the help of a classifier. The second sub-problem is NP-hard, so authors propose some greedy heuristic for its approximate solution. The paper [3] by Jones and Klinkner addressed rather similar problem, by proposing an algorithm for segmentation of search topics. As [1] they use a classifier to train weights of edges, so this step seems to be rather natural for the log segmentation methods. Unlike Boldi et. al. they describe a method for hierarchical segmentation, into search *goals* and *missions*. Such partitioning allows a more detailed look at the users' information needs. Methods of both papers significantly improve time-out based approach.

White et. al. in a series of papers [7,8,9,10] studies different aspects of browsing logs and its possible applications. In [7] Bilenko and White suggest that the information about the users interaction with search engine derived from post-query browsing trails should be used as implicit feedback. They try to predict and extrapolate intents of a user applying their models to searching and browsing activity of a range of users and propose algorithms for identifying relevant websites from the history of browsing and users' search behaviour. Search trails utilized in this paper are chains of pages connected via a sequence of clicked hyperlinks. In [8] White and Huang study the value of search trails for the original query and evaluate various metrics on it e.g. relevance, topic coverage, diversity. Since authors measure the value of trails for a particular query, they terminate the search trail either after a period of inactivity of 30 minutes, or with new query. They also show that users benefit more from full search trails. Singla et. al. in [9] solve the trailfinding problem: mine trails from raw logs. To identify segments of queries for further processing they use temporal cut-off. Several measures including coverage, diversity, relevance and utility are evaluated for extracted trails. In [6] Poblete et. al. authors in a unified way model two main aspects of users activity on the Web: browsing and searching. In the experimental part authors studied random walks on the new graph and show that its stationary distribution significantly outperforms the distributions for each of the hyperlink or bipartite query-click graphs.

3 Problem Statement

First we fix our notation and provide a number of definitions.

Definition. *Browsing log* $D_{\boldsymbol{u}} = \{d_1, \ldots, d_N\}$ is the recorded daily activity of a user $\boldsymbol{u}$ in the browser. Browsing log includes URLs of visited pages, queries submitted to a search engine, links followed by a user, timestamps of actions. Each d_i is a document viewed by a user with the available information: i.e URL, time of URL visit, text of the di crawled later during data preparation.

Definition. *Query log* $Q_{\boldsymbol{u}} = \{q_1, \ldots, q_n\}$ is a sequence of user's queries. Note that any query q_i corresponds to some record d_j in the browsing log.

Definition. *Browsing logical session* (or *logical session*) $D^{\boldsymbol{g}}_{\boldsymbol{u}} = \{d_{i_1}, \ldots, d_{i_K}\}$ is a subset of the browsing log, consisting of intent-related pages, i.e. pages visited with the same or similar search goal $\boldsymbol{g}$.

Definition. *Query logical session* $Q^{\boldsymbol{g}}_{\boldsymbol{u}} = \{q_{i_1}, \ldots, q_{i_k}\}$ is a subset of queries, unified into one search goal (=intent) $\boldsymbol{g}$.

As we have seen in the previous section, query and browsing logs have already been utilized in various fields of information retrieval and potentially have lots of new applications. Examples include personalisation of search results and query reformulations, integration of popular browsing trails into SERP, smoothing and generalisation of click data via query-flow-graph, session-oriented evaluation of a search engine performance, accurate modelling of user behaviour and so forth. Some of these applications do not need any special log partitioning and work pretty well on the raw data input, but the performance of most of them would considerably increase if browsing and query logs are supplied with additional logical partitioning. This fact was surely well understood in the previous studies, though for transparency of their models, authors (e.g. [8]) often used rather simple heuristics for logical segmentation. The most popular and very natural approach partitions browsing log $D_{\boldsymbol{u}}$ into consecutive series of queries relying on some time-out (typically, $T = 30$ min): a session continues only until it is followed by T minutes of inactivity. The disadvantages of this method are evident. First, often the answer to one question is immediately followed by a new query, even if the previous information need was completely satisfied and the user has another one in mind immediately. Generally, we do not want to glue such intents. Second, very often users return to some intent again and again during the day, so their intents could be *interleaved* — such repeated intents could not be joined together in principle. As we show further, even if the time-out threshold is tuned, the accuracy of such logical session identification on our data is upper-bounded by 72%.

To overcome these problems we propose a method for automatic segmentation of browsing logs. Our aim is to obtain a partition of pages visited by a user into intent related groups:

$$D_{\boldsymbol{u}} = \bigsqcup_{\boldsymbol{g}} D^{\boldsymbol{g}}_{\boldsymbol{u}} \tag{1}$$

Here $\boldsymbol{g}$ ranges over all goals user have in mind during her browsing session. Note that search goals are implicit in this decomposition — they are represented only

by the collection of underlying pages, not by any specific label. Naturally, Q_u is a subset of D_u , so decomposition 1 corresponds to the decomposition:

$$Q_u = \bigsqcup_g Q_u^g \tag{2}$$

which is exactly the well-known logical segmentation of a query log [1,3].

Table 1. Browsing log example and its partition

URL	session
http://www.some_search_engine/	1
http://www.some_search_engine/&q=ecir+wiki	1
http://en.wikipedia.org/wiki/ECIR	1
http://ecir2013.org/	1
http://en.wikipedia.org/wiki/Moscow	2
http://www.some_search_engine/	2
http://www.some_search_engine/&q=Moscow+attractions	2
http://ecir2013.org/cfpd.xml	1

To clarify our interpretation of query and browsing logical sessions we make one example[1]. In Table browsing log one sees the raw browsing log and its partitioning into two logical sessions (numbers 1 and 2 in the second column). The browsing log includes two queries submitted to the search engine: 'ecir wiki' and 'Moscow attractions', which reflect two different goals. We stress on several peculiarities of our approach to the problem. First, as follows from examination of real raw browsing logs, the logical sessions are naturally interleaved and this fact is crucial for our approach. Second, one URL could fall into different logical sessions depending on the context it occurs in, see occurrences of `http://www.some_search_engine/`. Third, very often the information need switches before, or even without any query being submitted to a search engine. In the example above intent 'Moscow attractions' supersede intent 'ECIR' before formulation of the query. This explains why the segmentation of browsing sessions is important and could very likely bring new information as compared to query logical sessions.

To evaluate our algorithms, we label raw browsing logs: split them into logical sessions manually. During labelling process the judge verifies visited pages iteratively and assigns some search intent to each page. To determine the intent of a page, the judge is allowed to look through several pages visited after the current one.

4 Methods of Browsing Log Segmentation

The proposed approach has some resemblance with the one described in [1], namely, first, we learn a classifier, which decides whether two records in the

[1] This example is not real record from our browsing logs.

browsing log share the same intent or not. Then we use the output of this classifier to glue visited pages into browsing logical sessions. We describe several methods of session extraction and evaluate them further in our experiments. As opposed to [1] or [3] we do not solve the rearrangement and session breaking problems separately, instead, we directly utilize the classifier and solve a certain clusterization problem.

4.1 Pairwise Classification

To train a classifier, we first build a learning and testing set: take a collection of pairs (d_1, d_2) of pages visited by one user and manually assign them labels $l(d_1, d_2, D_{\boldsymbol{u}}) \in \{0, 1\}$, choosing 1 if they belong to the same browsing logical session and 0 otherwise. The classifier is trained on the set of F of 29 features $f_i(d_1, d_2, D_{\boldsymbol{u}})_{i=1}^{F}$ extracted from the raw browsing log. Note that the labels as well as features depend on both pages d_1, d_2 themselves and the data extracted from raw browsing log, in particular we assign labels relying on the context (adjacent pages) of d_1 and d_2. This simplifies the labelling process and allows to implement additional contextual features. Using the learning set we train a GBDT (Gradient Boosted Decision Trees [2]) with logistic loss function, so, during the learning process, it assigns to each sample the probability $p(d_1, d_2, D_{\boldsymbol{u}})$ of getting the label 1 and maximizes the probability of correct classification of all samples.

As a result, we get a classifier, which is a function $p(d_1, d_2, D_{\boldsymbol{u}})$ (further we omit variable $D_{\boldsymbol{u}}$ in $p(\cdot)$) taking features of pages d_1 and d_2 and computing the probability that they belong to the same browsing logical session.

Basically implemented features fall into the following three classes:

URL Features measuring similarity of URLs u_1, u_2 of d_1 and d_2: cosine distance between vectors made of trigrams of u_1 and u_2, length of the longest common substring and its ratios to the lengths of URLs, match of hosts of u_1 and u_2.

Textual Features measuring similarity of texts t_1, t_2 of d_1, d_2: cosine distance between term vectors of t_1 and t_2, cosine distance between term vectors of titles t_1 and t_2, ratios of $tf.idf$ of common words to the lengths of each page, the same for top 10 words with the highest $tf.idf$ score.

Temporal Distances $|\tau(d_1) - \tau(d_2)|$ — time difference between moments d_1 and d_2 were opened in the browser, number of pages visited between d_1 and d_2.

Besides, we used context analogues of some features: context feature F is the same as F, but for the page preceding d_2 in the browsing log, instead of d_2.

We do not use any historical information about co-occurrence of pairs of documents in one browsing session, since it is very sparse for our task: two URLs visited sequentially by one user are not likely to be visited by significant number of other users.

4.2 Browsing Log Segmentation

Now we describe several approaches to browsing log segmentation using the outcome of the above-described classifier. Given the browsing log of one user $D\boldsymbol{u}$ we run our classifier for all such pairs d_{i_1} , d_{i_2} such that d_{i_1} precedes d_{i_2} in $D_{\boldsymbol{u}}$, so we know all the probabilities $p(d_i, d_j)$, $\tau(d_i) < \tau(d_j)$.

In general, we would like to obtain a partition 1 that maximizes the joint probability:

$$\mathcal{P} = \prod\nolimits' p(d_i, d_j) \prod\nolimits'' (1 - p(d_i, d_j)),$$

or equivalently the sum of logarithms of odds:

$$\Phi = \sum\nolimits' \log \frac{p(d_i, d_j)}{1 - p(d_i, d_j)}, \tag{3}$$

Here $\prod'$ and $\sum'$ run over all pairs of pages visited with the same intent and $\prod''$runs over all pairs visited with different intents. Unfortunately, the exact solution of this optimization problem is NP-hard. However, we propose several efficient algorithms to approach it, and provide detailed information on their performance on the real data. Some of our algorithms outperform significantly previous methods applied to the solution of the analogous problem.

Most our algorithms work iteratively. They are assumed to run online and model our labelling process. Namely, at every moment of time T we assume that the current browsing log $D_{\boldsymbol{u}}(T) = \{d|\tau(d) < T\}$ is already partitioned $D_{\boldsymbol{u}}(T) = \bigsqcup_{\boldsymbol{g}} D_{\boldsymbol{u}}^{\boldsymbol{g}}$ and decide what to do with the next n-th page d_n. We have two opportunities: 1) append d_n to the one of the existing logical sessions $D_{\boldsymbol{u}}^{\boldsymbol{g}}$ 2) start the new logical session $D_{\boldsymbol{u}}^{\boldsymbol{g}}$ with d_n. To choose between these opportunities, we use one of the following heuristics. The advantage of such approaches is the opportunity to use them as a part of online services with demands for real-time data processing.

1. **Last Page Maximal Likelihood.** We consider the last document $d_{\boldsymbol{g}} \in D_{\boldsymbol{u}}^{\boldsymbol{g}}$ for each current search goal $\boldsymbol{g}$. The page d_n is appended to the logical session $\boldsymbol{g}$, where $\boldsymbol{g} = \text{argmax} p(d_{\boldsymbol{g}}, d_n)$ if and only if $p(d_{\boldsymbol{g}}, d_n) > 1/2$, otherwise we start a new goal.
2. **All Pages Maximal Likelihood.** We consider all previous documents d_i. For the new page d_n we find d_i that maximizes probability $p(d_i, d_n)$ and append d_n to the logical session of d_i, if $p(d_i, d_n) > 1/2$, otherwise we start a new session.
3. **Greedy Appending.** We choose a logical session for document d_n that maximizes the sum of logarithms of odds3. That means we start a new session, if for all current goals $\boldsymbol{g}$ the sum

$$\Phi(\boldsymbol{g}) := \sum_{d_i \in D_{\boldsymbol{u}}^{\boldsymbol{g}}} \log \frac{p(d_i, d_n)}{1 - p(d_i, d_n)}$$

is negative, otherwise we append d_n to the $D_{\boldsymbol{u}}^{\boldsymbol{g}}$ with maximal sum of log-odds. It is not hard to see that this approach is the best solution to the problem of

log-odds optimization under natural constrains — 1) provide a partitioning at each moment during the current browsing session, and 2) do not permute documents in the detected segments, once they have been assigned to some partition.

We also propose an offline greedy algorithm, which works with the whole daily browsing log.

4. **Greedy Merging.** First attach each document to a separate session, then iteratively greedy glue a pair of sessions till it increases the sum of log-odds (3).

The advantage of the first approach is its efficiency — at each step we have to compute G probabilities, where G is the number of current goals, so all in all the classifier has to be ran $O(N \cdot G)$ times. Though, as we show further, its quality is relatively poor — any outlier or broken record in the raw browsing log will cut the logical session. The reminder methods, as method utilized in [1], require $O(N^2)$ runs of classifier, so, since in general $N \gg G$, they are less efficient, but turn out to be more effective.

The disadvantage of the latter approach is its nonapplicability online, since we need the whole browsing log to evaluate it, though it is still possible to utilize it for various offline applications of browsing logical sessions. As shown in Section 6.2, it outperforms others as an offline processing method.

5 Data

All the methods above were evaluated on the fully anonymized real browsing logs collected from one of the major search engines via the special browser toolbar, so, first, we discuss the data organization and information that is available for our purposes.

As opposed to the query log, the structure of the browsing log is much more complicated: the user could switch between multiple tabs in the browser, use 'back' button, follow a link or open a web page in a new window clicking a bookmark or a link from an external source (e.g. email client). In the browsing logs we store URL of every publicly available page visited by a user with the timestamp of its visit. If the user has followed a direct link, we also store the source URL along with the target. This data is not enough to recover complete user's behaviour in the browser — we are not able to detect a tab or window switch, 'back' button click. We believe that the lack of the complete information about all user's action does not harm our method, though we plan to verify it in the future work.

Given raw browsing log $D_{\boldsymbol{u}}$ we manually label all pages visited by a user $\boldsymbol{u}$ — assign a goal to each page d_i. The obtained partitioning $D_{\boldsymbol{u}} = \bigsqcup_{\boldsymbol{g}} D_{\boldsymbol{u}}$ serves two purposes. First, it is the source of the training and validation sets for our classifier: we pick all pairs of documents $d_i, d_j \in D_{\boldsymbol{u}}$ and assign to the pair label 1 if d_i, d_j fall into one logical session, and 0 otherwise. Second, it is used for

the evaluation of session extraction approaches, which are unsupervised and use only precomputed probabilities (see Section 4.2).

The main disadvantage of the labelling procedure is its complexity. Indeed, the judge, while assessing the raw session, has to keep in mind all the intents user had by the moment. On the other hand, it allows to construct large training set for the classifier, since raw users session of n documents gives $O(n^2) = \frac{n(n-1)}{2}$ samples.

Here are some statistics on the data. We have extracted 220 browsing logical sessions after labelling 50 raw logical sessions, which resulted into 151K labelled document pairs for the classifier, 78K of which were labelled positive positive, i.e. belong to the same session, and 73K were labelled negative. The average length of a logical session is 12 pages, the average number of logical sessions per raw browsing session is 4.4.

6 Evaluation

In this section we report results of the evaluation of our approaches on both classification and session extraction subtasks and compare them with the baseline — time-out approach with cut-off threshold tuned on the validation set. We also study contributions of various types of features to the final classifier and list the top of the most strong features.

Note that the segmentation $D_u = \bigsqcup) \boldsymbol{g} D_u$ automatically provides us with the underlying classifier — a pair (d_1, d_2) obtains label 1 iff d_1 and d_2 have occurred in one logical session. So it is possible to evaluate time-out based baseline not only on the session extraction task, but also on the classification task.

6.1 Pairwise Classification

Using the collected data we have learned a classifier with logistic loss function. The results of 50-fold cross-validation of the baseline and classifiers learned on various types of features are reported in Table 2. Standard F_1-score and accuracy of the classifier are chosen as the performance measures. Since in the previous studies [1,3] accuracy was chosen as the basic performance measure, we mostly focus on it and tuned parameters of the classifier relying on *Acc*.

$$Acc = \frac{\text{count of correctly classified samples}}{\text{total count of samples}}$$

Table 2. Perfomance of the classifier

Feature set	Baseline	All	without context	without text
F-measure	80%	83%	83%	82%
Accuracy	72%	82%	81%	81%

Rk.	Feature description	Score
1	Time difference bw. d_1 and d_2	1
2	LCS*	0.58
3	LCS/length(url_1)	0.40
4	LCS/length(url_2)	0.40
5	num. of pages visited bw. d_1 and d_2	0.33
6	URL trigramm match	0.32
7	context LCS/length(url_1)**	0.32
8	Same host	0.22
9	context LCS/length(url_2)	0.20
10	context LCS	0.20

Table 3. Top 10 features with their contributions to the final classifier. All scores are normalized to [0;1]. *LCS — length of the longest common substring of URLs. **context F is the same as feature F, but for the page preceding d_2 in the browsing log, instead of d_2.

For better understanding of the structure of our learning set we report performances of two trivial extreme classifiers (they are not our baselines). F_1-score and accuracy of 'All positive' classifier are 68% and 52%, F_1-score and accuracy of 'All negative' classifier are 73% and 48%.

We remind that in this section contextual features of documents d_1 and d_2 are ordinary features of d_1 and the page preceding d_2. For all experiments our classifiers significantly outperform the baseline. Interestingly, contextual and textual features do not improve quality of the classifier considerably. So, in certain cases, e.g. in presence efficiency constraints, context and textual features could be eliminated from the classifier without significant degradation of its performance.

Surprisingly textual features are not among the top 10 performing features, and they do not significantly improve our classifier. The best text-based feature is the cosine distance between term vectors of pages gets only to the 12th position. The probable reasons of poor performance of textual features are the change of content of considerable number of pages and high percentage of private pages (e.g. behind a login screen), which are not possible to access by a crawler.

6.2 Session Extraction

To evaluate the performance of the session extraction part 4.2 of our algorithms, we use Rand index, a conventional measure of similarity between two segmentations S_1, S_2 of the set of n elements. Namely,

$$R = \frac{n_1 + n_2}{\binom{n}{2}},$$

where n_1 is the number of pairs belonging to one segment both in S_1 and S_2, n_2 is the number of pairs belonging to different segments both in S_1 and S_2.

To evaluate the baseline, we have tuned its parameter — threshold of temporal cut-off on the labelled dataset and put $T = 20$ min. The Rand index of the baseline depending on the value of cut-off is plotted on Figure 1. Interestingly, it approximately confirms the optimal threshold of 30 minutes used in previous

researches, and shows that its performance does not change considerably in the interval [6;45] minutes. For the probabilities $p(d_i, d_j)$ learned from the classifier on the corresponding fold we evaluate our four methods of session extraction. Since the proposed clusterization methods are unsupervised, we could evaluate them directly on the test set of the classifier.

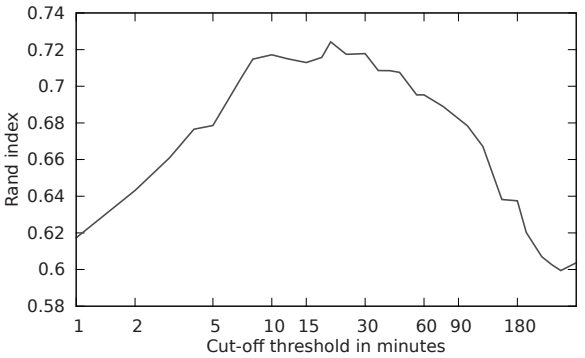

Fig. 1. Rand index for the temporal cut-off at T min on the log-scale

Table 4. Perfomance of session extraction methods

Method	Rand index
Baseline	0.72
Last page maximal likelihood	0.75
All pages maximal likelihood	0.79
Greedy appending	0.82
Greedy merging	0.86

The Rand indexes of these methods are reported in Table 4. As we have expected, all proposed algorithms significantly outperform the baseline. The most efficient 'Last page maximal likelihood' method does not considerably improve baseline in comparison with other our approaches, so for general purposes baseline seems to be more reasonable. The best performed method 'Greedy merging' could not be evaluated online, though it is still applicable only for offline processing of daily raw logs.

It is an interesting observation that appropriate choice of session extraction technique could significantly improve the quality of segmentation, especially considering that authors of [1,3] do not provide detailed study of the quality of greedy reordering algorithms.

7 Conclusions and Future Work

In this paper we have proposed and studied the problem of automatic raw browsing logs segmentation. Proposed framework allows significantly improve conventional time-out approaches and effectively identifies even interleaving sessions. We divided the whole problem into two subtasks classification and session extraction problems and evaluated performance of both of them separately. After being appropriately tuned our framework and its parts could be implemented for other segmentation tasks.

In the future we plan to conduct the detailed study of application of browsing logical sessions and estimate the influence of the quality of proposed segmentations on the quality of applications. Also, it is interesting to utilize complete knowledge about users actions in the browser for improvement and possible reorganization of logical sessions. Besides, we plan to examine interconnections between query and browsing logical sessions.

References

1. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., Vigna, S.: The Query-flow Graph: Model and Applications. In: CIKM (2008)
2. Friedman, J.H.: Greedy Function Approximation: A Gradient Boosting Machine
3. Jones, R., Klinkner, K.L.: Beyond the Session Timeout: Automatic Hi- erarchical Segmentation of Search Topics in Query Logs. In: CIKM (2008)
4. Hassan, A., Jones, R., Klinkner, K.L.: Beyond DCG: User Behavior as a Predictor of a Successful Search. In: WSDM (2010)
5. Ageev, M., Guo, Q., Lagun, D., Agichtein, E.: Find It If You Can: A Game for Modeling Different Types of Web Search Success Using Interaction Data. In: SIGIR (2011)
6. Poblete, B., Castillo, C., Gionis, A.: Dr. Searcher and Mr. Browser: a unified hyperlink-click graph. In: CIKM (2008)
7. Bilenko, M., White, R.W.: Mining the search trails of surfing crowds: iden- tifying relevant websites from user activity. In: WWW (2008)
8. White, R.W., Huang, J.: Assessing the scenic route: measuring the value of search trails in web logs. In: SIGIR (2010)
9. Singla, A., White, R., Haung, J.: Studying trailfinding algorithms for en- hanced web search. In: SIGIR (2010)
10. Guo, Q., White, R.W., Zhang, Y., Anderson, B., Dumais, S.T.: Why Searchers Switch: Understanding and Predicting Engine Switching Rationales. In: SIGIR (2011)
11. Wang, C.-J., Lin, K.H.-Y., Chen, H.-H.: Intent boundary detection in search query logs. In: SIGIR (2012)
12. Catledge, L., Pitkow, J.: Characterizing browsing strategies in the world-wide web. In: International World-Wide Web Conference on Technology, Tools and Applications (1995)
13. Anick, P.: Using terminological feedback for web search refinement — a log-based study. In: SIGIR (2003)
14. WSCD2012: Workshop on Web Search Click Data (2012)