Query Session Detection as a Cascade*

Matthias Hagen Benno Stein Tino Rüb

Bauhaus-Universität Weimar
99421 Weimar, Germany
<first name>.<last name>@uni-weimar.de

ABSTRACT

We propose a cascading method for query session detection, the problem of identifying series of consecutive queries a user submits with the same information need. While the existing session detection research mostly deals with effectiveness, our focus also is on efficiency, and we investigate questions related to the analysis trade-off: How expensive (in terms of runtime) is a certain improvement in F-Measure? In this regard, we distinguish two major scenarios where query session knowledge is important: (1) In an online setting, the search engine tries to incorporate knowledge of the preceding queries for an improved retrieval performance. Obviously, the efficiency of the session detection method is a crucial issue as the overall retrieval time should not be influenced too much. (2) In an offline post-retrieval setting, search engine logs are divided into sessions in order to examine what causes users to fail or to identify typical reformulation patterns etc. Here, efficiency might not be as important as in the online scenario but the accuracy of the detected sessions is essential.

Our cascading method provides a sensible treatment for both scenarios. It involves different steps that form a cascade in the sense that computationally costly and hence time-consuming features are applied only after cheap features "failed." This is different to previous session detection methods, most of which involve many features simultaneously. Experiments on a standard test corpus show the cascading method to save runtime compared to the state of the art while the detected sessions' accuracy is even superior.

Categories and Subject Descriptors: H.3.3 [Information Storage and Retrieval]: Query formulation, Search process
General Terms: Algorithms, Experimentation
Keywords: Web Search, Session Detection, Cascading Method

1. INTRODUCTION

We tackle the problem of query session detection, which aims at identifying consecutive queries a user submits for the same information need. Detecting such search sessions is of major interest as they offer the possibility to analyze potential techniques for supporting users stuck in longer sessions, to learn from the users' query reformulation patterns, or to obtain knowledge on how users behave when their initial queries were not satisfactory.

*Extended version of a paper presented at the ECIR 2011 Workshop on Information Retrieval over Query Sessions (SIR 11) [7].

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

Usually, the queries of a user form a stream ordered by submission time and are recorded in a query log by the search engines. Session detection for a single user then can be modeled as the task of deciding for each pair of chronologically consecutive queries whether the two queries were submitted for the same information need or not. This applies for an online scenario where the search engine is observing the user while searching, but also for an offline scenario where a recorded query log should be divided into the contained query sessions for further examination.

In an online scenario, session detection aims at supporting the user in satisfying her information need. Incorporating the knowledge of the queries that belong together may help the engine to present better results (e.g., the task of the TREC Sessions track) or to suggest queries that other users used in similar situations. Since online session detection is most useful when applied pre-retrieval, the efficiency of the detection method plays a crucial role. The overall retrieval time should not be increased too much in order not to cause a worse search experience at user site.

In an offline post-retrieval scenario, the goal of session detection is to identify typical reformulation patterns or to examine how search processes that finished satisfactory are different from such that are not. The efficiency of the session detection method is not as crucial as in the online scenario although faster methods are still favorable. However, the accuracy of the detected sessions is even more important than in the online scenario as conclusions drawn from examining large logs become more reasonable when the underlying data in form of detected query sessions is more reliable.

Our new approach for session detection—the cascading method—addresses both scenarios, online and offline detection. Unlike former approaches to the problem the cascading method does not require the simultaneous evaluation of all features. Instead, it processes the features in different steps one after the other by increasing computational costs and thus by increasing runtime requirements. Whenever a computationally cheap feature allows for a reliable decision, features with higher cost and runtime are not considered. If, however, the cheaper features do not provide a reliable decision basis, additional features are involved. For example, if a query contains the preceding query (e.g., `istanbul` and `istanbul archeology`), it is reasonable to assume that both queries belong to the same session. No other feature besides a simple query subset test is needed for that decision. For more complex situations like `istanbul archeology` and `constantinople`, the simple subset test fails but computationally more costly features that are able to identify semantic similarities can support the desired decision of also assigning these two queries to the same session (as Constantinople is the ancient name of Istanbul).

Query sessions are an interesting field of research with many published detection methods and also many studies that examine search behavior in sessions, query reformulation patterns, or improved retrieval techniques for known query sessions. In such stud-

ies, session detection is applied in a pre-processing step to obtain experimental data in form of sessions from a stored log. However, there is a huge gap between the state-of-the-art session detection methods and the ones that are practically used in the pre-processing of such studies. Most often the studies apply a simple time threshold: two queries belong to the same session iff the time gap between the queries is smaller than a given threshold (e.g., 5 or 10 minutes). The time threshold criterion is a very fast method but with a bad accuracy compared to more sophisticated methods. Is it the runtime alone that "bribes" so many studies to use time thresholds, although the bad accuracy is well known? In our opinion, there is an additional third critical issue that influences success of session detection methods besides the already mentioned efficiency and effectiveness: the simplicity of the method or, more precisely, the ease of re-implementation. In fact, time threshold methods are both very easy to be operationalized and extremely fast—two arguments that might give a reasonable trade-off for accepting inferior accuracy. In order to have a practical impact and to be applied by researchers, we argue that new detection methods should be equally fast and simple as time thresholds (and by far more accurate).

Our cascading method aims at bridging the described gap between developed and practically used methods. It is more easy to be implemented than many sophisticated state-of-the-art detection methods which often require intricate and not documented parameter tuning or training on data not available to the public. Furthermore, the cascading method is only about 4–5 times slower than a simple time threshold, making it faster than the current state of the art. These features come with an even superior accuracy of the detected sessions on a standard test corpus.

2. RELATED WORK

The earliest methods for session detection were mostly time based: two consecutive queries belong to the same session whenever the time elapsed is smaller than some threshold. Different time gaps have been tried and documented in the literature: 5 minutes [4], 10–15 minutes [8], 30 minutes [16], 60 minutes [2], or even 120 minutes [2]. The time thresholds were usually not intended to detect queries with the same information need but rather to study search behavior on bunches of queries. Nevertheless, time thresholds are an often used method to gather sessions in the "modern" notion of queries with the same information need (although the achievable accuracy is at most 70% [12]). As elaborated earlier, one reason for using time thresholds might be the very attractive simplicity compared to other existing session detection methods.

To overcome the bad accuracy of time thresholds, other methods also incorporate lexical features. For instance, queries that do not share any term often indicate a new session [10]. Other applied lexical features are the Levenshtein distance (how many operations are needed to transform one query into the other) [12] or the Jaccard coefficient (how large is the overlap) [13]. Some methods judge typical patterns like repeated query, specialization, or generalization as session continuations [11] or try to identify reformulations based on rules for changed words, deleted word suffixes, etc. [9].

However, all these lexical features are not able to determine semantic relatedness like in our introducing `istanbul archeology` vs. `constantinople` example. These two queries share no term and have very few overlapping character n-grams. For such cases several semantic features have been examined. An interesting idea is to enrich the short query strings in order to get a longer representation of the underlying information need. One possibility is to use the actual search results and researchers have tried the first 10 results [16], 50 results [18], 100 results [3], or even 500 results [14]. The methods then check the overlap of the set of URLs, the standard cosine similarity of the page titles and snippets, or even the cosine similarity of the result documents themselves [17]. In a very recent method the well-known ESA framework is applied for semantic similarity checks [13]. The ESA framework evaluates semantic similarity by comparing representations of the queries in a background collection (like Wikipedia documents) [5].

Table 1: Example query log. First the user had a Turkish history intent and then turned to a CIKM venue sports intent.

ID	Query	Click domain + rank	Time
42	`istanbul`	en.wikipedia.org 1	2011-05-22 20:34:17
42	`istanbul archeology`		2011-05-23 12:02:54
42	`istanbul archeology`	www.turizm.tr 6	2011-05-23 12:03:15
42	`istanbul archeology`	www.arkeoloji.tr 13	2011-05-23 18:24:07
42	`constantinople`		2011-05-23 19:12:40
42	`constantinople`	en.wikipedia.org 4	2011-05-23 19:13:02
42	`soccr glasgo`		2011-05-23 19:16:01
42	`soccer glasgow`		2011-05-23 19:16:11
42	`soccer glasgow`	www.soccer.uk 3	2011-05-23 19:16:15
42	`celtics vs rangers`		2011-05-23 20:33:04
42	`celtics vs rangers`	en.wikipedia.org 5	2011-05-23 20:33:12
42	`old firm`		2011-05-23 22:42:48

Most of the aforementioned features are not used alone (time thresholds being the main exception) but applied in different combinations. However, different features come with different computation costs and thus influence efficiency. For a query repetition it is obviously not necessary to check the search results in order to assign both queries to the same session. But none of the published session detection methods explicitly examines the efficiency aspect. Different to that line of research, we carefully design our new cascading method with respect to efficiency aspects. We also combine several features but involve time-consuming features only when cheaper features do not allow for a reliable decision.

We compare our method to the state-of-the-art geometric method by Gayo-Avello [6]. In his survey, Gayo-Avello compares most of the existing session detection approaches on a gold standard of human annotated queries and shows the geometric method's superiority in terms of detection accuracy. The geometric method is a very simple and fast method; it involves only the two basic features time threshold and lexical similarity. However, as a stand-alone approach, the geometric method is not able to detect semantic similarities. Hence, our cascading method builds upon the geometric method but equips it with simple additional steps. Our objective is threefold: a runtime performance comparable to the geometric method, a similar ease of implementation, and a further improved accuracy. In our cascading method we use an adapted version of the geometric method as the second of our four steps. In a first pre-processing step, simple patterns (repetitions, specializations, or generalizations) are identified and in two post-processing steps semantically related queries are detected. The pre-processing step ensures to save runtime compared to the original geometric method while the post-processing further improves accuracy.

3. PRELIMINARIES

A query q consists of a set of keywords. The search engine log additionally contains a user ID and a time stamp. If the user clicked on a result, the clicked result's rank and URL are also logged. We explain our cascading framework for the queries submitted by one user ordered by submission time. The problem of session detection is modeled as the problem of deciding for each consecutive pair q, q' of queries whether the session s that contains q continues with q' or whether q' starts a new session.

An example of a typical query log is given in Table 1 (the log

Table 2: Detected sessions after processing Step 1.

ID	Query	Click domain + rank	Time
42	istanbul	en.wikipedia.org 1	2011-05-22 20:34:17
42	istanbul archeology		2011-05-23 12:02:54
42	istanbul archeology	www.turizm.tr 6	2011-05-23 12:03:15
42	istanbul archeology	www.arkeoloji.tr 13	2011-05-23 18:24:07
42	constantinople		2011-05-23 19:12:40
42	constantinople	en.wikipedia.org 4	2011-05-23 19:13:02
42	soccr glasgo		2011-05-23 19:16:01
42	soccer glasgow		2011-05-23 19:16:11
42	soccer glasgow	www.soccer.uk 3	2011-05-23 19:16:15
42	celtics vs rangers		2011-05-23 20:33:04
42	celtics vs rangers	en.wikipedia.org 5	2011-05-23 20:33:12
42	old firm		2011-05-23 22:42:48

even contains a misspelled query right after the indicated intent boundary). Note that time is not a good indicator of session boundaries in the example as the user sometimes stopped working for longer time periods and then continued a pending session. For instance, in the last two queries, `old firm` is the name of the Glasgow `celtics vs rangers` soccer derby and both queries obviously have the same search intent. Regardless of the time passed, we assume that in such cases the user just continued the session. Another interesting observation is that at the highlighted search intent switch the passed time is quite short.

4. THE CASCADE: STEP BY STEP

Our cascading session detection method consists of four steps. From step to step the required features get more expensive in the sense of needed runtime but later steps are invoked only if the previous steps were not able to come to a reliable decision.

Step 1: Simple Query String Comparison

The most simple patterns that two consecutive queries q and q' may form can be detected by a simple comparison of the keywords: repetition ($q = q'$), generalization ($q' \subset q$), and specialization ($q \subset q'$). Whenever two consecutive queries represent one of these three cases, the cascading method assigns the queries to the same session regardless of the time that has passed between their submission. The rationale is that in case of a longer time gap, we assume the user to have continued a pending session. The effect of Step 1 in the example scenario is shown in Table 2. Note that Step 1 reliably detects generalizations and specializations, and captures all the interactions for the same queries. No other features have to be involved "inside" of the sessions that Step 1 detected. However, all the session boundaries cannot be trusted. Hence, for these cases, our cascading method invokes the geometric method [6] in Step 2 as a more sophisticated comparison of the query strings.

Step 2: Geometric Method

The geometric method [6] relaxes Step 1's query overlap condition with respect to the elapsed time between the queries. Let t and t' be the submission times of a pair q and q' of consecutive queries. Using the offset $t' - t$, the geometric method computes the time feature $f_{\text{time}} = \max\{0, 1 - \frac{t'-t}{24h}\}$. Thus, chronologically very close queries achieve scores near to 1 whereas longer time periods between query submissions decrease the score until it gets 0 for queries with a gap of 24 hours or larger. The lexical similarity f_{lex} for q' is computed as the cosine similarity of the character 3- to 5-grams of the query q' and the session s whose current last query is q. The geometric method votes for a session continuation

Table 3: Detected sessions after processing Step 2.

ID	Query	Click domain + rank	Time
42	istanbul	en.wikipedia.org 1	2011-05-22 20:34:17
42	istanbul archeology		2011-05-23 12:02:54
42	istanbul archeology	www.turizm.tr 6	2011-05-23 12:03:15
42	istanbul archeology	www.arkeoloji.tr 13	2011-05-23 18:24:07
42	constantinople		2011-05-23 19:12:40
42	constantinople	en.wikipedia.org 4	2011-05-23 19:13:02
42	soccr glasgo		2011-05-23 19:16:01
42	soccer glasgow		2011-05-23 19:16:11
42	soccer glasgow	www.soccer.uk 3	2011-05-23 19:16:15
42	celtics vs rangers		2011-05-23 20:33:04
42	celtics vs rangers	en.wikipedia.org 5	2011-05-23 20:33:12
42	old firm		2011-05-23 22:42:48

iff $\sqrt{(f_{\text{time}})^2 + (f_{\text{lex}})^2} \geq 1$. This inequality can be geometrically interpreted as plotting the point $(f_{\text{time}}, f_{\text{lex}})$ in the $\mathbb{R}^2$ and checking whether it lies inside or outside the unit circle (cf. Figure 1 (a)).

The effect in our example scenario is shown in Table 3 (remember that Step 2 is invoked only when Step 1 decided for new session). The geometric method correctly decides to remove the intent switch between the queries `soccr glasgo` and `soccer glasgow` as the elapsed time is rather short and many character n-grams overlap. This way, the geometric method is able to catch many typos and their corrections as often the corrected queries are lexically very similar and submitted within a small time gap.

Nevertheless, there are problematic cases where the geometric method cannot be trusted. There are three such session boundaries in Table 3 where this happens. One example is the pair `istanbul archeology` and `constantinople`. The only overlapping 3- to 5-grams of `constantinople` with the previous session are `sta`, `stan`, and `tan` such that the geometric method decides for a new intent. However, knowing that the ancient name of Istanbul was Constantinople, one would expect both queries to belong to the same session as semantically they are very similar. The other pairs in our example scenario are `soccer glasgow` and `celtics vs rangers`, and `celtics vs rangers` and `old firm`. As the Celtics and the Rangers are soccer teams from Glasgow and their derby is named "Old Firm," there should be no intent breaks.

Common to all three cases in the example scenario is that for the query pairs f_{time} is large (i.e., queries are chronologically close) but the lexical similarity reflected by f_{lex} is rather low. We conducted a pilot experiment to further examine where the geometric method fails. Therefore, we took a 10% sample of consecutive query pairs from the standard session detection corpus by Gayo-Avello [6] and analyzed the accuracy of the geometric method. Note that the remaining 90% of the corpus form our test set also used for evaluation in Session 5. Especially for pairs with $f_{\text{lex}} < 0.4$ and $f_{\text{time}} > 0.8$ the sample contained many wrong decisions (feature values tested in steps of 0.2).

To show that the derived bounds $f_{\text{lex}} < 0.4$ and $f_{\text{time}} > 0.8$ are also reasonable for the test set, Figures 1 (b) and (c) show the points in the $\mathbb{R}^2$ that the geometric method derives on the test set. Obviously, most pairs within a session are close to the right vertical border of the diagram, while pairs with a session break are close to the lower border. Cases where the geometric method decides wrongly are all the black dots in Figure 1 (b) that lie inside the unit circle and all the red dots in Figure 1 (c) that lie outside the unit circle. Figure 1 (d) gives the frequencies of the geometric method's errors. In the lower right corner ($f_{\text{lex}} < 0.4$ and $f_{\text{time}} > 0.8$) there are $583 + 14$ query pairs wrongly assigned to different sessions

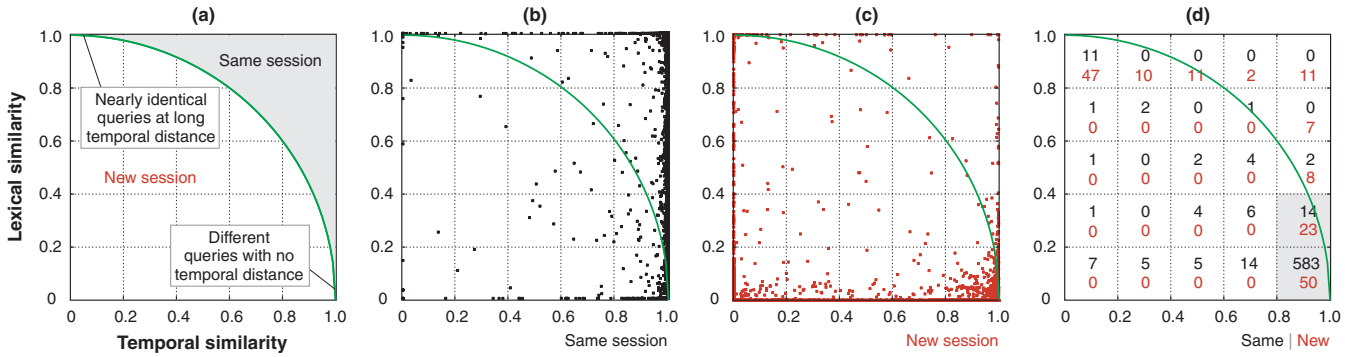

Figure 1: (a) Geometric interpretation of the geometric method. (b) Feature values of the geometric method for query pairs with "same session" annotation in the test corpus. (c) Feature values of the geometric method for query pairs with "new session" annotation in the test corpus. (d) Error frequency distribution of the geometric method's decisions. Black frequencies are query pairs wrongly assigned to different sessions. Red frequencies are pairs wrongly assigned to the same session.

although semantically related and $23 + 50$ query pairs wrongly assigned to the same session although semantically different.

Hence, for query pairs that are chronologically very close and lexically very different, the decisions of the geometric method are not reliable. Some of the respective pairs will not be related and correspond to real intent switches but some will have high semantic similarity and thus correspond to session continuations with different wordings. To detect these semantic relatedness or difference of queries, our cascading method drops the geometric method's decision for pairs that fall in the lower right corner and invokes Step 3 to further analyze semantic similarity.

Step 3: Explicit Semantic Analysis

An elegant way to compare semantic similarity of two short texts is the explicit semantic analysis (ESA) introduced by Gabrilovich and Markovitch [5]. The idea is to not compare the given two texts directly but to use an indexed background collection against which similarities are calculated. Since the index collection (e.g., the Wikipedia articles) can be pre-processed and stored, invoking ESA is not too expensive compared to the basic session detection features such as n-gram overlap or query submission time.

The ESA principle works as follows. During a pre-processing step, a $tf \cdot idf$-weighted term-document-matrix of the Wikipedia articles is stored as the ESA matrix. During runtime, the two to-be-compared texts represented as vectors are multiplied with the ESA matrix and the cosine similarity of the resulting vectors yields the ESA similarity feature f_{esa}. In our setting, the two texts that should be ESA-compared are the keywords of q' and all the keywords of the queries in the session s to which the previous query q belongs. As Anderka and Stein [1] showed that the ESA performance varies only very little with the size of the index collection, we conducted a pilot experiment on the 10% subsample of Gayo-Avello's corpus also used in the analysis of Step 2. We randomly sampled English Wikipedia articles (10 000, 100 000, 1 million, all) and could verify Anderka and Stein's observation that the size of the index collection did not really matter for the accuracy of the session decisions except that 10 000 articles performed a little worse than the larger sizes. Hence, the cascading method's Step 3 implementation uses a random sample of 100 000 Wikipedia articles as the ESA index collection. Our pilot experiment also reveals that for $f_{\text{esa}} \geq 0.35$ (we tested in steps of 0.05) a quite reliable decision can be made that the two examined queries belong to the same session.

Table 4 shows the effect of Step 3 in our example scenario. The intent switches between the queries `istanbul archeology` and `constantinople` and between `soccer glasgow` and `celtics vs rangers` are correctly removed. However, the session break between `celtics vs rangers` and `old firm` still remains as the corresponding ESA similarity is to low. Hence, for $f_{\text{esa}} < 0.35$ the decision is still questionable (remember that `old firm` is just another name for the `celtics vs rangers` soccer match). Accordingly, the cascading method does not immediately view q' as the start of a new session in this case but invokes Step 4 that aims at further enlarging the representation of the queries.

Table 4: Detected sessions after processing Step 3.

ID	Query	Click domain + rank	Time
42	`istanbul`	en.wikipedia.org 1	2011-05-22 20:34:17
42	`istanbul archeology`		2011-05-23 12:02:54
42	`istanbul archeology`	www.turizm.tr 6	2011-05-23 12:03:15
42	`istanbul archeology`	www.arkeoloji.tr 13	2011-05-23 18:24:07
42	`constantinople`		2011-05-23 19:12:40
42	`constantinople`	en.wikipedia.org 4	2011-05-23 19:13:02
42	`soccr glasgo`		2011-05-23 19:16:01
42	`soccer glasgow`		2011-05-23 19:16:11
42	`soccer glasgow`	www.soccer.uk 3	2011-05-23 19:16:15
42	`celtics vs rangers`		2011-05-23 20:33:04
42	`celtics vs rangers`	en.wikipedia.org 5	2011-05-23 20:33:12
42	`old firm`		2011-05-23 22:42:48

Step 4: Search Result Comparison

Step 4 compares the search results of the queries. As retrieving such results requires time-consuming index accesses at search engine site or even the submission of two web queries from an external client, the web results are applied only when all previous steps failed to provide a reliable decision. Using web search results to detect semantically similar queries is applied in different variants in the literature (cf. the discussion in Section 2). We evaluated different settings in a pilot study on the 10% sample of Gayo-Avello's test corpus also used in the analyses of Steps 2 and 3. As a result, we chose to compare the sets of URLs of the top-10 retrieved documents via the Jaccard coefficient. Whenever q' returns at least one of the top-10 results of q, we view this as an argument for a session continuation. Otherwise, q' starts a new session.

In our running example scenario, Step 4 correctly removes the intent switch between the queries `celtics vs rangers` and `old firm` as both return the "Old Firm" Wikipedia article in their top-10 Google results. Hence, the decisions after processing all four steps of the cascading method match the desired outcome depicted in Table 1. But note that a "new session" decision after Step 4 still cannot be guaranteed to be correct as semantically related queries may exist with low f_{esa}-value and no shared top-10 search results.

The Complete Cascade

Putting all the pieces together, the cascading method's pseudo code is given as Algorithm 1.

Algorithm 1 Cascading method for one query pair q and q'

```
1: if q ⊆ q' or q ⊃ q' then "same session"                    ▷ Step 1
2: else if f_lex ≥ 0.4 or f_time ≤ 0.8 then
3:     if sqrt((f_time)^2 + (f_lex)^2) ≥ 1 then "same session" ▷ Step 2
4:     else "new session"
5: else if f_esa ≥ 0.35 then "same session"                   ▷ Step 3
6: else if shared top-10 result then "same session"           ▷ Step 4
7: else not sure, but probably "new session"
```

5. EXPERIMENTAL EVALUATION

We conduct a two-part experimentation to examine the performance of our new cascading method. In a first experiment, we compare against the state-of-the-art geometric method on a standard session detection test corpus. In a second experiment, we show the scalability of our method on a large query log.

Comparison with the State of the Art

To ensure comparability, we evaluate our cascading method on the annotated gold standard query corpus that Gayo-Avello used in his experiments showing the geometric method to be superior to other approaches for session detection [6]. Note that using the same corpus, we have to compare our cascading method only with the best performing method of Gayo-Avello's comprehensive study to ensure comparability with all other approaches as well. The corpus contains 11 484 queries of 223 users sampled from the 2006 AOL query log [15]. The queries are manually subdivided into 4 254 sessions with an average of 2.70 queries per session. As we have sampled 10% of the corpus for parameter tuning of Steps 2–4, the experimental evaluation is done on the remaining 90% of the corpus. For evaluation purposes we use the F-Measure $F_\beta = \frac{(1+\beta^2)\cdot prec \cdot rec}{\beta^2 \cdot prec + rec}$, where precision and recall of the detected sessions are measured against the human gold standard. We follow Gayo-Avello and set $\beta = 1.5$, which emphasizes wrong session continuations as the bigger problem compared to wrong breaks.

The results of this experiment can be found in Table 5. First of all, we could verify Gayo-Avello's reported good performance of his geometric method achieving an F-Measure of 0.9181. Note however, that the cascading method improves upon this value and achieves an F-Measure of 0.9327 (the difference is statistically significant according to a paired t-test with confidence level $p = 0.05$). The better accuracy is due to an improved session recall that more than compensates for a slightly worse precision.

We further analyze the impact step by step. The results are given in Table 6 and should be read as follows. After Step 1 (the subset test) about 40% of the test set's query pairs are reliably judged (column "reliable") and the F-Measure is 0.8303 if one would stop the processing after Step 1. Note that the runtime for invoking Step 1 on a pair of queries is about 0.08 ms on average (experimentation done on a standard quad-core desktop PC with 8 GB RAM). After Step 2 another 35% of all queries are judged reliably (altogether about 75%) with an F-Measure of 0.9292. Hence, applying the pre-processing Step 1 improves the accuracy of the geometric method.

Table 5: Accuracy values on the Gayo-Avello corpus.

	Precision	Recall	F-Measure
Geometric method	**0.8667**	0.9430	0.9181
Cascading method	0.8622	**0.9679**	**0.9327**

Table 6: Step by step performance on the Gayo-Avello corpus.

	reliable	F-Measure	time	factor
Step 1	40.49%	0.8303	0.08 ms	1.0
Step 2	35.15%	0.9292	0.20 ms	2.5
Step 3	2.05%	0.9316	0.27 ms	3.4
Step 4	0.85%	0.9327	9.85 ms	123.1

That the pre-processing also significantly speeds up the whole process can be seen as Step 2 requires about 0.20 ms per query pair (2.5 times more than Step 1 (column "factor")).

Step 3 is invoked on about 25% of all queries and runs with a pre-computed ESA matrix in main memory. Nevertheless, Step 3 is a little slower than Step 2 with an average time of about 0.27 ms per query pair. Step 3 reliably assigns "same session" for another 2.05% of all queries raising the F-Measure to 0.9316 if one would stop the cascading method after Step 3. Note that after Step 3 the cascading method is still faster than the original geometric method: 2.5 times faster on about 40% of the queries and 1.35 times slower on about 25% (0.27 ms vs. 0.20 ms).

The only crucial issue for runtime is Step 4 which is invoked on about 22% of all query pairs. For our experimental setting we modeled Step 4 as follows. All the queries on which Step 4 would be invoked were submitted to the Bing-API and the top-10 result URLs stored. On runtime of the cascading method, these result lists were held in memory to simulate a very fast index lookup at search engine site. Nevertheless, even in this artificial setting, Step 4 on average needs about 9.35 ms per query pair. This yields a factor of more than 100 compared to the time for Step 1. Applying Step 4 against the API's of commercial search engines would be even inferior as then a typical query needs at least 200 ms. Furthermore, note that Step 4 only assigns a "same session" decision to about 0.85% of all queries (column "reliable") and only slightly increases the resulting F-Measure by 0.011. That the improvement is so small is due to the fact that the ESA step already catches many of the semantically related query pairs that can be detected automatically. Hence, when efficiency is an issue, Step 4 is the obvious bottleneck even in the simulated environment of having the results in memory.

Improved Cascading Method: Drop Last Step

As Step 4 is the clear efficiency "problem" within the cascading method, we also suggest a very fast three-step variant of the cascade by removing Step 4. As described above, the resulting method is about 15% faster than the original geometric method while the accuracy of the detected sessions is even improved from 0.9181 to 0.9316 (again found to be a statistically significant difference).

Furthermore, consider the scenario in which session detection is applied offline and not at search engine site with access to a web index etc. Consider for instance some academic non-industry researcher whose use case is not to break a complete search engine log into all contained sessions but just to sample sessions with very high detection accuracy for further experiments. This might for instance be the case in the pre-processing step of studies that want to evaluate new retrieval ideas for improved handling of query sessions etc. In this scenario, Step 4 of the original cascade would have to be implemented against the public API of some commercial search engine with time requirement of at least 200 ms per query. For larger logs this is prohibitive and for such cases we suggest an additional tweak to the three-step cascading method. This second variant of the three-step cascading method assigns a "not sure, maybe new session" decision when Step 3 votes for a new session (about 22% of all query pairs in the Gayo-Avello test corpus). In a post-processing, all sessions starting with such a "not sure" decision can be deleted. In case of the Gayo-Avello test corpus this removes about 30% of the sessions but the remaining ones then

achieve an almost perfect F-Measure of 0.9755. Hence, this version of the three-step cascading method with post-processing can be used as a very fast and very accurate session detection method that extracts very high quality sessions for further experimentation.

Scalability on a Large Log

We tested the suggested three-step variants of the cascading method on the 2006 AOL query log [15]. To this end, we cleaned the original log of about 36.4 million interactions collected in 3 months from about 650 000 users in order to remove users that probably are bots or that are not relevant for session detection. We had three simple rules for the cleaning: (1) we removed all users that only had one interaction (session detection would be senseless on these), (2) we removed all users that on average had less than 10 seconds between two consecutive interactions (probably bots), and (3) we removed all users whose median query length is more than 100 characters (probably bots). All in all, the cleaning removed about 1 million interactions and 130 000 users. Hence, the remaining log contains about 35.4 million interactions. With the figures from the experiments on Gayo-Avello's corpus one would expect about 8 million queries (22% of 35.4 million) to have to be submitted to a search engine when applying the full four-step cascade. Further assuming an average time of about 300 ms per web query against a typical API of some commercial search engine would yield 1 month of processing time. Of course, some queries could be parallelized etc.; but nevertheless the time consumption remains in the order of days compared to several minutes for the other three steps. Hence, these figures emphasize the fact that for larger analyses the three-step cascading method should be applied.

The three-step cascade on the cleaned AOL log takes about 27 minutes on a standard quad-core desktop PC with 8 GB RAM (compare this to several days for the full four step variant). We also tested the effect of removing all the sessions starting with a "not sure" decision after Step 3: as in the Gayo-Avello test corpus this affects about 30% of the sessions.

Altogether, our experiments show that for analyses of smaller logs the full four-step cascade can be applied although the accuracy is only slightly better than for the three-step cascade. If, however, the focus is on larger logs or the online scenario, the efficiency issue comes into play and the three-step cascading method should be chosen. It is about 15% faster than the original geometric method and comes with an even improved accuracy. Furthermore, for the task of sampling a few sessions with very high detection accuracy, an adapted version of the three-step cascade can remove the about 30% of sessions on which it is not sure. The remaining sessions then have a very good quality compared to human judgments.

6. CONCLUSION AND OUTLOOK

We have presented the cascading method for query session detection, which is based on the combined use of the geometric method and the well-known ESA model. The cascading method achieves a very high detection accuracy against a human gold standard and is superior to the current state-of-the-art geometric method. In contrast to other published methods, the cascading method sensibly invokes time-consuming features only if cheaper features fail to provide a reliable session detection. In its three-step variant our method is more accurate than the geometric method while saving 15% of the runtime. If equipped with a post-processing step that drops sessions with "not sure" decisions, the achieved accuracy is almost perfect on Gayo-Avello's gold standard. Hence, the cascading method is an ideal pre-processor for many evaluations that need to extract high quality sessions from query logs as experimental data. Another salient characteristic of the cascading method is its simplicity, which renders costly training data or an intricate hyperparameter tuning superfluously: the three-step cascade just involves a basic pre-processing, the simple geometric method, and the standard ESA framework.

An interesting aspect for future research is the integration of a post-processing that accounts for multitasking at user site, or for hierarchies of different search goals and missions [12, 13, 19, 20]. The idea of multitasking is that users may shortly interrupt longer search sessions for some completely different intent, and, after the short interruption, continue the original session. The idea of search goals and missions is that larger user search missions like planning the next vacation comprise of smaller goals like booking a flight, searching a nice hotel, etc. In future work, we plan to enhance the cascading method to also find search hierarchies or goals/missions. A potential way to achieve this would be to run the current cascade and then carefully merge the detected sessions into an appropriate hierarchy also respecting multi-tasking issues.

7. REFERENCES

[1] M. Anderka and B. Stein. The ESA retrieval model revisited. In *SIGIR 2009*, pp. 670–671.

[2] N. Buzikashvili and B. J. Jansen. Limits of the web log analysis artifacts. In *WWW 2006 Workshop on Logging Web Activity*.

[3] S.-L. Chuang and L.-F. Chien. A practical web-based approach to generating topic hierarchy for text segments. In *CIKM 2004*, pp. 127–136.

[4] D. Downey, S. T. Dumais, and E. Horvitz. Models of searching and browsing: languages, studies, and application. In *IJCAI 2007*, pp. 2740–2747.

[5] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI 2007*, pp. 1606–1611.

[6] D. Gayo-Avello. A survey on session detection methods in query logs and a proposal for future evaluation. *Information Sciences*, 179 (12):1822–1843, 2009.

[7] M. Hagen, B. Stein, and T. Rüb. Query Session Detection as a Cascade. In *ECIR 2011 Workshop on IR over Query Sessions*.

[8] D. He and A. Göker. Detecting session boundaries from web user logs. In *BCS-IRSG Colloquium 2000*, pp. 57–66.

[9] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *CIKM 2009*, pp. 77–86.

[10] B. J. Jansen, A. Spink, C. Blakely, and S. Koshman. Defining a session on web search engines. *JASIST*, 58(6):862–871, 2007.

[11] B. J. Jansen, A. Spink, and B. Narayan. Query modifications patterns during web searching. In *ITNG 2007*, pp. 439–444.

[12] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM 2008*, pp. 699–708.

[13] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Identifying task-based sessions in search engine query logs. In *WSDM 2011*, pp. 277–286.

[14] D. Metzler, S. T. Dumais, and C. Meek. Similarity measures for short segments of text. In *ECIR 2007*, pp. 16–27.

[15] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Infoscale 2006*, paper 1.

[16] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD 2005*, pp. 239–248.

[17] M. Sahami and T. D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In *WWW 2006*, pp. 377–386.

[18] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM 2005*, pp. 824–831.

[19] A. Spink, H. C. Özmutlu, and S. Özmutlu. Multitasking information seeking and searching processes. *JASIST*, 53(8):639–652, 2002.

[20] A. Spink, M. Park, B. J. Jansen, and J. O. Pedersen. Multitasking during web search sessions. *Inf. Process. Manage.*, 42(1):264–275, 2006.