From Search Session Detection to Search Mission Detection

Matthias Hagen Jakob Gomoll Anna Beyer Benno Stein

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

ABSTRACT

Search mission detection aims at identifying those queries a user submits for the same information need. Such knowledge offers interesting insights into behavioral usage patterns and often can help to better support a user. However, most existing query log studies focus on search *sessions* only (consecutive queries for the same need) and ignore multitasking behavior (interleaved information needs) as well as hierarchies of short-term search goals in multiple sessions that form a long-term search task such as vacation planning. To better understand the dialog between user and search engine we distinguish between (1) physical search sessions, characterized by the time gap between queries, (2) logical search sessions, characterized by consecutive queries for the same information need within a physical session, and (3) search *missions*, characterized by logical sessions, multitasking behavior, and hierarchical goals.

Our contributions are threefold. First, we present a new algorithm for logical session detection, which follows the state-of-the-art cascading method's rationale of combining effectiveness with efficiency. Our approach is applicable within the time-critical online scenario, where a search engine tries to support users by incorporating knowledge about their search history on the fly, as well as within the offline scenario, where the objective is to accurately partition a collected log. We improve several steps of the cascading method, among others by exploiting Linked Open Data information. Second, we demonstrate our new algorithm's applicability to accurately detect search missions. Third, we introduce a new publicly available corpus of 8800 queries labeled with session and mission information.

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

1. INTRODUCTION

Session detection has a long history in the field of query log analysis for web search. A major goal of session detection is the identification of typical patterns the users follow in their search processes, and, based on these patterns, to develop search support tools such as smart query suggestions learned from the reformulations of other users. In the early days of session detection the rather simplistic understanding of *physical* sessions was prevailing: all consecutive queries of a user without some longer time gap (typically 30 minutes or more) were considered a search session. Physical sessions can be easily detected, but the obtained insights are limited to basic statistical analyses such as the number of queries submitted per time frame. Similarly, exploiting queries from a physical session for query reformulation purposes is limited as well: often, pairs of consecutive queries do not target the same information need and hence are not related (e.g., the lines 4–7 in the example query log given in Table 1). As a consequence, the focus of session detection research turned to a content-related view, called *logical* sessions, which designate series of consecutive queries within a physical session that target the same information need. Logical sessions can provide us with deeper insights on how users refine their searches, or help us to identify cases when users encounter problems, indicated for example by the formulation of numerous related queries accompanied by few result clicks.

Note, however, that relying on series of consecutive queries alone will miss important connections, even if the targeted information need is considered (as is done by a logical session analysis). This shortcoming is rooted in the fact that behavioral patterns of search engine users include interleaving searches in a multitasking manner [23, 24], as well as hierarchies of different search goals within so-called search *missions* [15]. For example, a user may shortly interrupt a longer search task for checking the weather forecast (e.g., line 5 in the example log in Table 1), or she may follow subordinate search goals spanning several days, which altogether form a larger mission (e.g., all the queries on Turkish history in the example). Ideally, different logical sessions targeting the same information need should be connected, which actually is the task of search mis-

Table 1: Example query log (clicks omitted) divided into physical search sessions with a threshold of 30 minutes.

Line	Query	Timestamp
1	`ancient turkey`	2012-12-20 20:02:44
2	`history istanbul`	2012-12-20 20:24:17
3	`istanbul archeology`	2012-12-21 12:02:54
4	`istanbul archeology`	2012-12-21 18:31:21
5	`weather new york`	2012-12-21 18:45:23
6	`constantinople`	2012-12-21 18:45:36
7	`footbal lisbon`	2012-12-21 19:14:01
8	`football lisbon`	2012-12-21 19:14:11
9	`benfica vs sporting`	2012-12-21 20:23:04
10	`derby eterno`	2012-12-21 22:42:48
11	`constantinople`	2012-12-21 23:09:02
12	`constantinople`	2012-12-21 23:27:38

Table 2: **Example query log divided into search missions.**

	Query	Timestamp	Search mission
1	`ancient turkey`	2012-12-20 20:02:44	M1: Turkish history
2	`history istanbul`	2012-12-20 20:24:17	
3	`istanbul archeology`	2012-12-21 12:02:54	
4	`istanbul archeology`	2012-12-21 18:31:21	
5	`weather new york`	2012-12-21 18:45:23	M2: Weather NYC
6	`constantinople`	2012-12-21 18:45:36	M1: Turkish history
7	`footbal lisbon`	2012-12-21 19:14:01	M3: Lisbon football
8	`football lisbon`	2012-12-21 19:14:11	
9	`benfica vs sporting`	2012-12-21 20:23:04	
10	`derby eterno`	2012-12-21 22:42:48	
11	`constantinople`	2012-12-21 23:09:02	M1: Turkish history
12	`constantinople`	2012-12-21 23:27:38	

sion detection: identify those queries that a user submits for the same information need. The example log in Table 1 should hence be divided into the three missions shown in Table 2. Note that the labeling of the missions with an expressive phrase in the rightmost column is not part of the mission detection problem but provided only for readability purposes. In the mission detection problem as understood here, assigning an information need ID such as M1, M2, or M3 to each query is sufficient.

The paper in hand presents three main contributions to the state of the art of search mission detection. First, it provides a new method for logical session detection as a building block of mission detection. This method follows the stepwise paradigm of the recent cascading method [8]. By improving the single steps and adding a new step that employs Linked Open Data analyses to detect a possible semantic relatedness of queries, the new approach achieves a better efficiency and effectiveness in extensive experiments. Second, we show that applying our new method in a two-phase strategy (first logical session detection, then merging sessions into missions) provides a very efficient means to tackle the mission detection problem. Third, we publicly release a new large scale corpus of 8800 queries annotated with search session and search mission information.

2. FRAMEWORK AND RELATED WORK

For each query q a search engine query log contains the query string, a user ID and a time stamp denoting when q was submitted. We assume that the query log for one user is ordered by time stamp. Information on clicked search results (e.g., rank or domain), shown snippets, etc. might be additionally available but is not necessary for running our proposed method. We view a query as a set of keywords, such that for instance $q \subset q'$ for a query pair means that the keywords from q form a subset of the keywords from q'.

The problems we are dealing with can be formally described as follows. Given the query log of one user, the problem of physical session detection is to split the log at those pairs of consecutive queries q, q' where the time elapsed between submission of q and submission of q' exceeds some given threshold. Given the query log of one user, the problem of logical session detection is to split the log into those series of consecutive queries within physical sessions that are submitted for the same information need. Given the query log of one user, the problem of search mission detection is to identify which queries are submitted for the same information need. Note that an alternative formulation of search mission detection is to connect those logical sessions of one user that concern the same information need. Hence, search missions can be built upon logical sessions which in turn can be built upon physical sessions.

Our two formulations of session detection distinguish between the simple time based notion of a physical session and the more contextually oriented logical sessions. Published studies on query log analysis often do not explicitly state which notion of a search session they follow. Even worse, often logs are just split into physical sessions when the study context suggests that indeed logical sessions should be used. One reason might be that typically detecting physical sessions is much more easy (just check some time threshold) than logical session detection. However, with our newly developed efficient logical session detection method this should be no argument anymore. Note that some studies even follow another different concept of a "search session" viewing them as all the consecutive queries that are submitted for the same information need (regardless of the time gaps!). However, we believe that in these cases the term "session" with its typical flavor of some temporal connection should be replaced by the more general notion of a search mission (not just consecutive queries) established by Jones and Klinkner [15].

The earliest methods for session detection aimed at the detection of physical sessions (without explicitly saying so) and hence were 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 [9], 30 minutes [20], 60 minutes [2], or even 120 minutes [2]. But note that with the detection of physical sessions it is very likely that queries for different information needs are mixed. The logical session accuracy achievable with any time threshold as the only feature is rather low in the range of 70–80% [15]. Nevertheless, many researchers unfortunately still apply time thresholds to gather sessions in the "modern" notion of logical sessions; as elaborated earlier, one reason might be the very attractive simplicity compared to more sophisticated methods.

To overcome the bad logical session detection accuracy of time thresholds, other methods also incorporate lexical features. For instance, queries that do not share any term often indicate a new session [13]. Other applied lexical features are the Levenshtein distance [15] (how many characters need to be changed to transform one query into the other) or the Jaccard coefficient [17] (how large is the character or word overlap). Some methods judge typical patterns like repeated query, specialization, or generalization as session continuations [14] or try to identify reformulations based on rules for changed words, deleted word suffixes, etc. [12]. The geometric method by Gayo-Avello [6] is a very fast and simple combination of a time threshold and lexical query similarity.

However, note that lexical features are not able to determine semantic relatedness. Consider for instance the two queries `istanbul archeology` and `constantinople`. Both queries share no term and have very few overlapping character n-grams; still they are related as Constantinople is the ancient name of Istanbul. 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 [20], 50 results [22], 100 results [3], or even 500 results [18]. 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 [21]. Still, obtaining actual search results can be quite costly. Thus, in a very recent method [17] the well-known ESA framework [5] is used to check the semantic similarity of query representations in a background collection (e.g., Wikipedia articles).

Most of the aforementioned features are not used alone (time thresholds being the main exception) but often applied simultaneously 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. This is the idea of the cascading method by Hagen et al. [8] that was designed with respect to efficiency aspects. It also combines several features but involves time-consuming features only when cheaper features do not allow for a reliable decision. In their paper, Hagen et al. show their method to outperform the previous state-of-the-art geometric method in terms of detection accuracy while at the same time even being faster. As the cascading method sensibly involves different features in different steps considering the increasing computational costs, it is much faster than other methods that combine all features simultaneously. However, the cascading method also follows the notion of a "session" as the consecutive series of queries (very long inactivity gaps allowed) and thus can neither detect interleaved information needs nor can it cope with long-term search missions spread over several sessions. In both cases the cascading method would just identify the smaller (interleaved) pieces of longer missions in isolation without recognizing their connections.

Only three methods for multitasking and mission detection are published yet [15, 16, 17]. They apply the entire feature set at once and do not follow the cascading method's idea of deferring the exploitation of computationally costly features [8]. In contrast, our proposed new approach adopts the cascading framework for the first time in the context of multitasking and mission detection and improves the cascading method in three respects: (1) by adding a new first step for physical session detection, (2) by tailoring the cascading method's steps for logical session detection, and (3) by adding a new step for the semantic similarity check of two queries. The new step is based on an analysis of Linked Open Data connections, a technique previously used for the identification of typical query reformulation patterns [10, 11]. We show that a two-round protocol of running the improved cascade twice is able to identify multitasking behavior and search missions with good accuracy.

As only one small scale corpus for multitasking detection evaluation is publically available [17], we develop a new one order of magnitude larger corpus with about 8800 queries based on the standard corpus for session accuracy evaluation by Gayo-Avello [6].

3. LOGICAL SESSION DETECTION

As for logical session detection, we improve the original cascading method that had four steps with increased feature (computation) cost from step to step [8]. However, we have to carefully adapt the original cascading method as its underlying concept of a "session" significantly differs. The original cascade tries to identify sequences of consecutive queries for the same information need without explicitly looking at the intermediate time gaps. For instance, whenever two queries are identical, the original cascade puts them in the same "session" even when weeks have passed. Our conceptual framework differs in the sense that such long gaps indicate different physical sessions whose contained logical sessions might then be merged in a mission detection phase. Hence, the notion of a search mission is more general than Hagen et al.'s "session" concept that only looks for sequences of consecutive queries. Missions instead also allow multitasking behavior in physical sessions and even several other physical sessions between queries for the same information need.

Even though our concept of a search session differs, we follow the cascading method's paradigm of combining effectiveness and efficiency considerations (i.e., we also use expensive features only when cheaper features fail to provide reliable decisions). In the first step of the original cascade, query pairs that are repetitions, generalizations, or specializations are put in the same session. The second step involves the geometric method [6] that combines both lexical similarities and the time elapsed between two queries. Whenever the time is below a predefined threshold and the lexical similarity is high, the two consecutive queries are assigned to the same session. However, when only a short period of time passed between two consecutive queries and the lexical similarity is low, the second step will miss semantically related queries. To address this problem, the cascade involves an ESA step (Explicit Semantic Analysis) [5] to identify semantically related query pairs. In a final fourth step, the top-10 search results of the two queries are compared.

Table 3: Example after Step 1 (asterisks tag trusted decisions).

Line	Query	Timestamp
1	`ancient turkey`	2012-12-20 20:02:44
2	`history istanbul`	2012-12-20 20:24:17
*		
3	`istanbul archeology`	2012-12-21 12:02:54
*		
4	`istanbul archeology`	2012-12-21 18:31:21
5	`weather new york`	2012-12-21 18:45:23
6	`constantinople`	2012-12-21 18:45:36
7	`footbal lisbon`	2012-12-21 19:14:01
8	`football lisbon`	2012-12-21 19:14:11
9	`benfica vs sporting`	2012-12-21 20:23:04
*		
10	`derby eterno`	2012-12-21 22:42:48
11	`constantinople`	2012-12-21 23:09:02
12	`constantinople`	2012-12-21 23:27:38

We improve the individual steps and adapt the cascade to our framework of first detecting physical sessions and only then tackling logical sessions. Therefore, we will use a simple time-based first step splitting a log into physical sessions and add a new step following ESA that is based on Linked Open Data connections.

Finally, we propose a two-phase protocol of the improved cascade for mission detection: a first phase on query level detects logical sessions, a second phase on session level merges them into missions.

Step 1: Time-based Physical Sessions

As for physical session detection, we stick to a quite conservative approach and view sequences of consecutive queries without a time gap of more than 90 minutes as one physical session (i.e., the log is split into physical sessions at gaps of more than 90 minutes). The rationale for not using a shorter gap (e.g., 30 minutes is employed in many studies) is that shorter gaps still might split at points where the user was just on a longer reading + click trail which should be seen as the same physical session although not every interaction is with the search engine. But not interacting with the search engine for 90 minutes could be easily regarded as having finished one physical session; it could for instance be interpreted as the typical duration of a lunch break. However, any time threshold is debatable and our suggested method does not really depend on the exact value. Further note that our focus is not on physical session detection but on logical session detection and mission detection. Any queries for the same information need, that might have been split into two physical sessions by whatever threshold value applied, should finally be merged into one mission such that the exact physical session threshold in the end does not really matter.

The example log after applying Step 1 is shown in Table 3. When turning to logical sessions it becomes clear that any split decision made by Step 1 is correct as logical sessions are defined as parts of physical sessions. Hence, any query pair where Step 1 decides to break does not have to be handed to Step 2: it is already decided with high confidence (depicted by an * in the table). All the non-split decisions of Step 1 cannot be trusted (for logical session detection) but have to go deeper down the cascade to decide whether both queries are submitted for the same information need or not.

Table 4: Example after Step 2 (asterisks tag trusted decisions).

	Line	Query	Timestamp
	1	`ancient turkey`	2012-12-20 20:02:44
	2	`history istanbul`	2012-12-20 20:24:17
*	3	`istanbul archeology`	2012-12-21 12:02:54
*	4	`istanbul archeology`	2012-12-21 18:31:21
	5	`weather new york`	2012-12-21 18:45:23
	6	`constantinople`	2012-12-21 18:45:36
	7	`footbal lisbon`	2012-12-21 19:14:01
	8	`football lisbon`	2012-12-21 19:14:11
	9	`benfica vs sporting`	2012-12-21 20:23:04
*	10	`derby eterno`	2012-12-21 22:42:48
*	11	`constantinople`	2012-12-21 23:09:02
	12	`constantinople`	2012-12-21 23:27:38

Step 2: Simple Patterns

The most simple patterns that two consecutive queries q and q' for the same information need 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 within a physical session represent one of these three cases, our method assigns the queries to the same logical session. For efficiency reasons, the step is not implemented on keyword sets (as in the original cascade) but as a substring test on the query strings.

The effect of Step 2 in the example scenario is shown in Table 4. Note that for all the pairs that Step 2 assigns to the same logical session (only the pair in lines 11 and 12 in the example scenario) no other features have to be involved as the decisions already are reliable (again, all high confidence decisions so far that are not handed to later steps are depicted by a *). However, all the logical session boundaries detected by Step 2 within physical sessions cannot be trusted. Hence, for these cases, our method invokes a more sophisticated lexical similarity check in Step 3.

Step 3: Lexical Similarity and Time

The original cascade uses the geometric method by Gayo-Avello [6] to detect lexical similarity. This method computes a time feature $f_{\text{time}} = \max\{0, 1 - \frac{t'-t}{24h}\}$ for the submission times t, t' of a pair q, q' of consecutive queries as well as the cosine similarity f_{lex} between the character 3-/4-/5-grams of the query q' and the session s whose current last query is q. The geometric method votes for a session continuation iff $\sqrt{(f_{\text{time}})^2 + (f_{\text{lex}})^2} \geq 1$. Whenever $f_{\text{lex}} < 0.4$ and $f_{\text{time}} > 0.8$, the original cascade does not trust the geometric method's decision but invokes a semantic similarity check (ESA) in a next step. The rationale is that for query pairs relatively close in time that have a low lexical n-gram similarity, still some semantic relatedness could be found showing both queries to be related (and probably having the same intent).

We will not directly use the geometric method as described above. In our improved lexical similarity step we change the feature computation, the session continuation condition, and the above described "trust" range for invoking the ESA step. First note that the original cascade and the geometric method allow two queries to fall in one session even when the time gap is more than one day (24h)—due to the definition of f_{time}. But our notion of a session is different and Step 3 is guaranteed to be called only on pairs where $t' - t \leq 90\text{min}$.

Table 5: Example after Step 3 (asterisks tag trusted decisions).

	Line	Query	Timestamp
	1	`ancient turkey`	2012-12-20 20:02:44
	2	`history istanbul`	2012-12-20 20:24:17
*	3	`istanbul archeology`	2012-12-21 12:02:54
*	4	`istanbul archeology`	2012-12-21 18:31:21
	5	`weather new york`	2012-12-21 18:45:23
	6	`constantinople`	2012-12-21 18:45:36
*	7	`footbal lisbon`	2012-12-21 19:14:01
	8	`football lisbon`	2012-12-21 19:14:11
	9	`benfica vs sporting`	2012-12-21 20:23:04
*	10	`derby eterno`	2012-12-21 22:42:48
*	11	`constantinople`	2012-12-21 23:09:02
	12	`constantinople`	2012-12-21 23:27:38

Hence, we can easily change $f_{\text{time}} = 1 - \frac{t'-t}{90\text{min}}$. This even saves some operations (= runtime) as no maximum has to be computed.

From our newly developed query corpus (cf. Section 5), we use a 25% sample as the training set for tuning parameters. We evaluated the original geometric method with our new f_{time}-definition and made the following observations. For f_{lex} we can use only character 3- and 4-grams saving runtime and not impairing the session accuracy. Furthermore, for $f_{\text{lex}} > 0.15$ (all feature values identified by a grid search on the 25% training set with steps of 0.05) we can simply ignore the time aspect for a moment (saving another four arithmetic operations per query pair) and assign both queries to the same session. The reason again is the guarantee that our new method invokes Step 3 only for query pairs pretty close in time. The original geometric method would often also have assigned such pairs to the same session due to the existing lexical similarity compared to the short time gap in its underlying 24h-frame. However, when $f_{\text{lex}} \leq 0.15$ the time comes in again. Such query pairs with low lexical similarity are split into two logical sessions when $f_{\text{time}} < 0.6$.

The remaining query pairs with $f_{\text{lex}} \leq 0.15$ and $f_{\text{time}} \geq 0.6$ constitute the range where semantic similarity should be checked. All these pairs have low lexical similarity but were submitted within 36 minutes such that semantic relatedness could be supposed. Hence, our new method invokes the original cascade's next step (ESA) on all these pairs and otherwise trusts the third step's decisions. Due to the much simpler feature computation and decision criteria, our lexical similarity step is much more efficient than the original cascade's geometric method—while not harming effectiveness.

The effect of Step 3 in the example scenario is shown in Table 5. Note that the geometric method is especially good at detecting typo corrections (lines 7 and 8) that typically come with a very short time gap and high lexical similarity. All the decisions with low lexical similarity and small time gap cannot be trusted. Hence, for these cases, our method invokes a semantic similarity check in Step 4.

Step 4: Semantic Similarity Using ESA

Gabrilovich and Markovitch [5] suggested an explicit semantic analysis (ESA) method to compare the semantic similarity of two short texts. Basically, the two texts are not compared directly but similarities against documents in an indexed background collection are calculated. The indexed collection (often Wikipedia articles) can be preprocessed and stored, such that using ESA is not too expensive compared to for instance lexical cosine similarity.

Table 6: Example after Step 4 (asterisks tag trusted decisions).

	Line	Query	Timestamp
	1	`ancient turkey`	2012-12-20 20:02:44
*			
	2	`history istanbul`	2012-12-20 20:24:17
*	— — —	— — — — — —	— — — — — —
	3	`istanbul archeology`	2012-12-21 12:02:54
*	— — —	— — — — — —	— — — — — —
	4	`istanbul archeology`	2012-12-21 18:31:21
	— — —	— — — — — —	— — — — — —
	5	`weather new york`	2012-12-21 18:45:23
	— — —	— — — — — —	— — — — — —
	6	`constantinople`	2012-12-21 18:45:36
	— — —	— — — — — —	— — — — — —
	7	`footbal lisbon`	2012-12-21 19:14:01
*			
	8	`football lisbon`	2012-12-21 19:14:11
	— — —	— — — — — —	— — — — — —
	9	`benfica vs sporting`	2012-12-21 20:23:04
*	— — —	— — — — — —	— — — — — —
	10	`derby eterno`	2012-12-21 22:42:48
	— — —	— — — — — —	— — — — — —
	11	`constantinople`	2012-12-21 23:09:02
*			
	12	`constantinople`	2012-12-21 23:27:38

We employ the ESA method as follows. In a preprocessing 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 queries q and q' represented as term vectors are multiplied with the ESA matrix and the cosine similarity of the resulting vectors yields the ESA similarity feature f_{esa}. Using all the keywords of the queries in the logical session s of query q instead of only keywords from q did not show much difference in pilot experiments on our 25% training set.

Anderka and Stein [1] showed that the ESA performance varies only very little with the size of the index collection. In our pilot experiments on the 25% training set we also tried different numbers of randomly sampled English Wikipedia articles (10 000, 100 000, 1 million, all) and verified that the accuracy of the logical session decisions did not really differ—except that 10 000 articles performed a little worse. Hence, our implementation of Step 4 uses a random sample of 100 000 Wikipedia articles as the ESA index collection. Our pilot experiments also revealed that for $f_{\text{esa}} \geq 0.28$ (we tested in steps of 0.01) a high confidence decision can be made that the two examined queries belong to the same session.

Table 6 shows the effect of Step 4 in the example scenario. The intent switch between the queries `ancient turkey` and `istanbul archeology` is correctly removed. However, the session break between `football lisbon` and `benfica vs sporting` still remains as the corresponding ESA similarity is too low. Hence, for $f_{\text{esa}} < 0.28$ the decision is still questionable (remember that both Benfica and Sporting are football clubs from Lisbon). Accordingly, the cascading method does not immediately view q' as the start of a new session in the case of $f_{\text{esa}} < 0.28$ but invokes Step 5 that checks semantic similarity based on Linked Open Data connections.

Step 5: Semantic Similarity Using LOD

In addition to semantic similarity from the ESA step described above, we equip the cascading method with a new fifth step. Within this new step, the semantic similarity of a pair q, q' of consecutive queries from a physical session is analyzed via the Linked Open Data (LOD) graph of DBpedia.[1] Hollink et al. [10, 11] used Linked Open Data for detecting typical reformulation patterns in an image search query log and suggested to test the potential for web search as well—an idea that we now pick up.

DBpedia contains RDF triples representing entities and relationships between entities mined from Wikipedia. Viewing entities as nodes and relationships as edges, the data defines a graph. Finding a path in this graph from one entity to another is a means of establishing some semantic relationship between these two entities. Typically, the more paths between two entities exist and the shorter these paths are, the more related (semantically similar) are the two entities. There is for instance a two step path from `benfica` to `sporting` since both are football clubs in Lisbon.

To efficiently find paths in the DBpedia graph, a preprocess stores the entities and the relationship RDF triples in an index as follows. For each entity e there is a postlist that contains all entities occurring in an RDF triple with e. An entity e has an idf-inspired associated value $idf_e = \log\left(\frac{pl}{pl_e}\right)$, where pl is the total number of postlists and pl_e is the number of postlists that contain e (which equals the length of e's postlist). The weight of an entity e is set to the $[0, 1]$-normalized value $weight_e = \frac{idf_e}{idf_{max}}$, where max is the entity with largest idf-value. The idea of these idf-style weights follows the usage of idf-values in information retrieval: paths containing entities that occur in a lot of RDF triples (i.e., that have long postlists) are probably less descriptive of a semantic similarity than paths with less frequently occurring entities. Note for instance that all person entities have a relationship to their respective home country such that a path including a country is a rather weak evidence of semantic similarity between two persons. However, when both have played for the same football club this can be viewed as a "stronger" relationship and thus a much better similarity indicator.

In practice, the new LOD step is implemented as follows. First, the main DBpedia entities in the queries q and q' are identified via a slightly modified version of the Wikipedia title query segmentation algorithm [7] (the modification is to also allow 1-word segments that are ignored by traditional segmentation algorithms). Thereby, the main entities of a query are the Wikipedia title segments chosen by the Wikipedia title segmentation approach (basically, in case of overlapping Wikipedia titles the ones with higher web frequency are favored). These main entities are mapped to their corresponding DBpedia entities by employing a dictionary that unifies different "names" of an entity to the same generic entity in the LOD graph (e.g., `benfica` is unified to `s.l. benfica`). The LOD step then identifies all one- and two-step paths (i.e., at most one intermediate entity) from one main entity e in q to one main entity e' in q'. This can be accomplished efficiently by merging the indexed postlists of e and e'. The feature value f_{lod} is obtained by the maximum weight of any of the found paths. The weights of one-step paths are set to 1.0 and the weight of a two-step path is set to $weight_p$ of the intermediate entity p connecting e and e'. The queries q, q' are assigned to the same logical session iff $f_{\text{lod}} > 0.6$ (value determined on our 25% training set by a search with 0.01 steps). Otherwise, q' is viewed as the start of a new session.

Table 7 shows the effect of Step 5 in the example scenario. The intent switch between `football lisbon` and `benfica vs sporting` is correctly removed as there are short paths from both football clubs to the football entity and the Lisbon entity in Wikipedia. But note that still none of the logical session split decisions made by Step 5 is a high confidence split. In the example query log all the splits of Step 5 (all the splits not tagged with a *) are correct but in real world scenarios there often exist cases of query pairs with low ESA and low LOD similarity that still belong to the same logical session (e.g., in our newly developed search mission corpus presented in Section 5). These might for instance be pairs without LOD entities. Hence for all split decisions of Step 5, we adopt the final step of the original cascading method that compares the queries' actual search results.

[1] `http://dbpedia.org/`

Table 7: **Example after Step 5 (asterisks tag trusted decisions).**

	Line	Query	Timestamp
	1	`ancient turkey`	2012-12-20 20:02:44
*			
	2	`history istanbul`	2012-12-20 20:24:17
*		— — — — — — — — — —	— — — — — — —
	3	`istanbul archeology`	2012-12-21 12:02:54
*		— — — — — — — — — —	— — — — — — —
	4	`istanbul archeology`	2012-12-21 18:31:21
		— — — — — — — — — —	— — — — — — —
	5	`weather new york`	2012-12-21 18:45:23
		— — — — — — — — — —	— — — — — — —
	6	`constantinople`	2012-12-21 18:45:36
		— — — — — — — — — —	— — — — — — —
	7	`footbal lisbon`	2012-12-21 19:14:01
*			
	8	`football lisbon`	2012-12-21 19:14:11
*			
	9	`benfica vs sporting`	2012-12-21 20:23:04
*		— — — — — — — — — —	— — — — — — —
	10	`derby eterno`	2012-12-21 22:42:48
		— — — — — — — — — —	— — — — — — —
	11	`constantinople`	2012-12-21 23:09:02
*			
	12	`constantinople`	2012-12-21 23:27:38

Step 6: Comparing search results

Using web search results to identify semantic similarity of a query pair q, q' is part of many session detection algorithms (cf. Section 2). We evaluated different settings in a pilot study on our 25% training sample. As a result, Step 6 compares 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 same information need and thus as a continuation of a logical session. Otherwise, q' is viewed as starting a new logical session. However, retrieving search results requires time-consuming index accesses at search engine site or even the submission of two web queries from an external client. Hence, the web results are applied only when all previous steps fail to provide a trusted decision. In our running example scenario, Step 6 does not remove any intent switches such that the result stays the same as after Step 5.

Discussion

Note that after the six steps there are "new session" decisions that still are not guaranteed to be correct logical session splits. It is always more difficult to automatically decide that a query pair is *not* related than finding some evidence for lexical or semantic similarity. In real world settings, related queries do exist that have low f_{lex}-, f_{esa}-, and f_{lod}-values, and no shared top-10 search results. Our current cascade will wrongly split such pairs (but not with high confidence). Developing sophisticated steps that can identify such involved related information needs is a very interesting future research direction—but probably also rather challenging when efficiency issues have to be considered.

Despite the difficulty of establishing trustable splits, note that all "same session" decisions after Step 6 are high confidence decisions (from the definition of the single step's trust ranges). This means that the connections within the logical sessions detected at the end of the cascading process always come with high confidence evidence for their underlying same information need—which is what logical session detection should aim for. Merging related logical sessions into search missions forms the second phase of our proposed algorithm.

4. TWO-PHASE MISSION DETECTION

The above described six step scheme can be viewed as the first phase of a mission detection algorithm. Assume that a query log is already split into logical sessions after this first phase. By applying a similar cascade on the detected logical sessions, they can be merged into missions also accounting for multitasking behavior. Hence, the first phase runs the six step cascade on query level, the second phase then runs a similar cascade on logical session level.

A straightforward idea is to merge two logical sessions s and s', when an adapted variant of the cascade would assign the last query q of s and the first query q' of s' to the same logical session—this way masking out all the queries from intermediate sessions. One might argue that q' could be checked against all queries from s and its assigned mission m or even a pairwise comparison of all queries from s' to all from m could be performed. However, we restrain from such a protocol as it is not practical in the online scenario (live observing one user's interactions) due to runtime issues coming with the large number of potential comparisons. Furthermore, our experiments will show that comparing the last and first query of two to-be-merged logical sessions very often suffices.

Note that for mission detection, time gaps between queries do not play the same role as for logical session detection. Hence, Step 1 (physical sessions) and Step 3 (lexical similarity) where time is involved as a feature have to be changed: we want to be able to merge logical sessions with more than 90 minutes time gap (e.g., lines 2 and 3 in the example). Step 1 is omitted completely and from Step 3 time is removed. All remaining steps remain unchanged.

In practice, for some logical session s' the cascade first checks against the preceding mission m. When s' is assigned to the same mission, the mission detection cascade can move on to the logical session following s'. Otherwise, s' is checked against the mission preceding m etc. On the training set, we recognized that when a mission is picked up in a later logical session there are hardly more than ten intermediate other logical sessions (more than ten in only 15% of the cases). For efficiency reasons, we thus use the following cascading mission detection protocol. The first query q' of s' is first checked for simple patterns (Step 2 of the session detection cascade) against the last queries of the preceding ten logical sessions one after the other. Whenever a high confidence merge decision is possible, the mission detection cascade does not invoke more costly steps. If no simple pattern merge is detected, the lexical similarity step (Step 3 of the session detection cascade) runs against the last queries of the preceding ten logical sessions etc. This way, the mission detection misses all continuations with more than ten intermediate sessions but also saves a lot of runtime (especially for sessions that have no related previous session).

In the example scenario, mission detection is done on the logical sessions given in Table 7. The first two logical sessions (lines 1–2 and line 3) are merged by the ESA step as the query pair in lines 2 and 3 has high enough f_{esa}. Line 4 is merged into the same mission because it is a query repetition of line 3. The first four lines now form one intermediate mission. Line 5 is checked against that mission but no cascade step detects a similarity from line 5 to 4. Line 6 is first checked against line 5 without success but then checked against the line 1–4 mission and merged because ESA detects a similarity of lines 6 and 4. The lines 7–9 logical session is checked (using line 7) without success against lines 6 and 5. Then line 10 is merged with lines 7–9 as the LOD step finds paths from `benfica` and `sporting` to `derby eterno` (which is the name of that classic Lisbon football derby). Finally, line 11 is checked without success against line 10 but then against line 6 a query repetition is triggered such that the final outcome is the mission splitting from Table 2 as desired.

Practical Remarks

Having established the stepwise cascading process of using one feature after the other and the two phases for mission detection, one could assume that using the cascade would require several runs over a data set. Note however, that all steps can be applied one after the other on one query pair until a decision is reached before moving on to the next pair. In fact, reusing computed time gaps from Step 1 in Step 3 even saves some operations. Also the second phase for mission detection can be implemented directly after the cascade finished logical session detection on one query pair: whenever a split decision is made, the current query can directly be checked against the last queries of previous missions before moving with the first phase to the next query pair.

5. EXPERIMENTAL EVALUATION

The accuracy of detected logical sessions or missions is usually evaluated against corpora of manually labeled query logs. So far, only two such corpora are publically available. Gayo-Avello's corpus [6] consists of 11 484 queries from 215 users sampled from the 2006 AOL query log [19] with respect to the representativeness of typical querying behavior (ratio of repeated queries, click through rate, etc.). A single human annotator subdivided the sample into 4040 sessions with an average of 2.70 queries per session. The main drawbacks of Gayo-Avello's corpus are (1) that the underlying notion of a "session" is only focused on series of consecutive queries submitted for the same information need (e.g., very long periods of inactivity are contained in the sessions and no relationships between different sessions on the same information need are annotated), (2) that query submission times and clicks are not included but have to be reconstructed from the original AOL log, (3) that almost half of the sampled users submitted less than 4 queries such that session detection for them makes no real sense, (4) that sometimes the ordering of the queries in the sample does not reflect the original ordering in the AOL log, and finally, (5) that some queries from some users are left out with no reason (not even privacy).

Lucchese et al.'s corpus [17] consists of 1424 queries from 13 users also sampled from the AOL log. However, not all queries from the sampled users are contained in the corpus but only very few selected periods of querying where no gap between two subsequent queries is longer than 26 minutes (i.e., only few physical sessions but not the whole query stream). This invalidates any attempt at identifying long term tasks spanning several physical sessions as about 97% of the original queries from the 13 users in the AOL log are discarded from the sample. As a result, the rather small Lucchese et al. sample does not reflect typical querying behavior with respect to long-term or reoccurring tasks.

New Corpus for Session and Mission Analysis

The described drawbacks render the available corpora not applicable to the evaluation of search mission detection. Hence, to reliably evaluate our method, we do not use the existing corpora but create a new one. To still ensure some level of comparability with previous studies, we choose the large Gayo-Avello sample as our basis. From the AOL log we extracted all queries of the 215 users contained in the Gayo-Avello sample. We removed the few queries that are empty or just a URL (probably submitted by users mixing up the search field with the address bar) and all queries from the 88 users that submitted less than 4 queries in total (too few queries for reasonable logical sessions). The final query sample contains 8840 queries from 127 users. Two human annotators divided this sample into 2881 logical sessions and 1378 missions. Cases where the annotators did not agree initially were discussed to reach a consensus. This new corpus is made freely available as the Webis Search Mission Corpus 2012 (Webis-SMC-12).[2]

On average, a user in our corpus conducts 10.85 missions with 6.42 queries each. A mission on average contains 2.09 logical sessions. Missions and sessions are interrupted by a different session or mission and later picked up again in 1505 cases by 93 users. About 76.3% of all the missions are finished within one day (about 34.5% contain just one query); the longest non-trivial mission runs 88 days (a user comparing banks in St. Louis).

[2]`http://www.webis.de/research/corpora`

Table 8: Stepwise performance on the Webis-SMC-12 test set.

	decided	$F_{1.5}$-Measure	time	factor
Step 1	23.87%	0.807	0.033 ms	16.5
Step 2	48.72%	0.845	0.002 ms	1.0
Step 3	13.28%	0.925	0.178 ms	89.0
Step 4	0.60%	0.930	0.237 ms	118.5
Step 5	0.11%	0.930	12.770 ms	6385.0
Step 6	2.03%	0.946	13.359 ms	6679.5

Performance of the Improved Cascade

Our evaluation test set is formed by the 75% of our new mission detection corpus that were not used as the training set for tuning the cascade's parameters in Sections 3 and 4.

Session detection.

For evaluation of logical session accuracy, we employ 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 logical session breaks are measured against the human gold standard. We set $\beta = 1.5$, which emphasizes wrong logical session continuations as the bigger problem compared to wrong breaks.

The stepwise results of the cascading process are shown in Table 8 and should be read as follows. After Step 1 (physical sessions) about 24% of the test set's query pairs are decided with high confidence (column "decided") such that they will not be touched by later steps. The logical session accuracy however is rather low as expected (F-Measure of 0.807). After Step 2 (substring test) another 49% of the query pairs are decided and the F-Measure is 0.845 if one would stop the processing after Step 2. Note that the runtime for invoking Step 2 on a pair of queries is really fast—around 0.002 ms on a standard quad-core PC with 8 GB RAM; runtime factors of other steps in column "factor". After Step 3 (lexical similarity) another 13% of all queries are decided (altogether about 86% after three steps) with an F-Measure of 0.925.

Step 4 (ESA) is invoked on about 14% of all queries and runs with a pre-computed ESA matrix in main memory. Nevertheless, Step 4 is a little slower than Step 3 with an average time of about 0.24 ms per query pair. Step 4 with high confidence assigns "same session" for another 0.40% of the pairs, thus only slightly raising the F-Measure to 0.930 if one would stop the cascading method after Step 4. Note that after Step 4 our improved cascade is still faster than the original three step cascade or the geometric method.

The only crucial issue for runtime are Steps 5 and 6 which are invoked on about 13.5% of all query pairs. Both are more than 50 times slower than ESA even though the search result step was modeled as follows for our experimental setting. All the queries on which Step 6 would be invoked were submitted to a commercial search engine 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 6 on average needs about 13 ms per query pair. This yields a factor of almost 6700 compared to the time needed for Step 2 (substring test). However, Step 6 assigns a reliable "same session" decision for about 2% of all queries (column "decided") and increases the resulting F-Measure to 0.946. But when efficiency is an issue, Steps 5 and 6 are the obvious bottlenecks.

Note that the original cascading method [8] achieves an F-Measure of only 0.853 on the test set (when equipped with the 90 minute physical session detection step for a fair comparison it still is just 0.93). Our improved cascade is faster (more pairs decided with fast steps) and after Step 6 achieves an F-Measure of 0.946. However, the main reason for the improvement is not the new LOD step (Step 5) but the improved variant of Step 3. There is no change in the F-Measure from Step 4 to Step 5 (the LOD step correctly identifies six continuations and has one wrong continuation). The main problem is the characteristic of the web queries. A detailed analysis shows that most query pairs on which the LOD step is invoked, are hard pairs with no LOD entity at all—the LOD step cannot do anything then. Thus, to show the potential effectiveness of the LOD step, we also compare it to the ESA step on a manually developed sample of 100 pairs of person and place names that are semantically related. On this set, the LOD analysis identifies 77 of the semantically related pairs while the ESA step can only identify 59. Despite its rather weak performance on the Webis-SMC-12, the LOD analysis potentially has a much better coverage than ESA (only the Webis-SMC-12 query sample does not reflect that). Possible further improvements of the LOD step's time efficiency and effectiveness could be pruning the LOD graph or indexing complete Wikipedia articles (cf. Section 6).

Mission detection.

To evaluate the mission detection accuracy, we run the second phase of the mission detection cascade on the logical sessions annotated in the test set (75% of the new corpus). The cascade correctly identifies 865 of the 1134 mission continuations missing only 269 continuations (157 due to the horizon effect of just checking against the previous ten sessions). This clearly shows the improved cascade's applicability to mission detection. However, there also is an error rate of 307 sessions that are wrongly assigned to be a continuation. Most of the errors are due to the semantic steps that for instance identify a similarity between health related queries that the annotators split due to different diseases tackled. Without the three semantic steps, the mission detection would still correctly identify 807 continuations and only vote for 113 wrong ones. As a side remark, note that this is a strong evidence for the hypothesis that users often pick up an abandoned mission by using a lexically very similar query. Since the runtime without the semantic steps is much faster and the accuracy is much better, we suggest the restricted cascade for mission detection.

6. CONCLUSION AND OUTLOOK

We have presented an improved cascading method for logical session detection that can also be applied to mission detection. The improvements relate to the original cascade's geometric method step and a new step that checks the semantic similarity of two queries based on a Linked Open Data (LOD) analysis. Just like with the original cascade, time-consuming features are applied only when the cheaper features failed to provide a reliable session detection.

As for the evaluation, we have developed a new publicly available corpus of 8800 queries annotated with logical session and mission information. On this corpus, our improved cascade outperforms the original method with respect to the detected logical sessions' accuracy while being comparably efficient. Our experiments have also demonstrated the potential of the improved cascade for the detection of multitasking behavior at user side and for merging shorter logical sessions into longer search missions.

An interesting topic for future research are efficient semantic steps to identify more of the "hard" relations. One idea could be to speed up the LOD step by pruning the underlying graph in a preprocessing. Another step might have an index of the complete Wikipedia to check the pages of the LOD entities for semantic similarity instead of just the connections in the LOD graph. Also the search engine result step could be improved. In the online scenario, the results (or snippets) of the previous query are known and it would be quite cheap to for instance check whether the keywords of the following query are related to these results (no additional index access necessary). With logged snippets and top-10 URLs, such a step could also be very efficient in the offline scenario.

7. REFERENCES

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

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

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

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

[5] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proc. of 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, M. Potthast, A. Beyer, and B. Stein. Towards optimum query segmentation: In doubt without. In *Proc. of CIKM 2012*, pp. 1015-1024.

[8] M. Hagen, B. Stein, and T. Rüb. Query session detection as a cascade. In *Proc. of CIKM 2011*, pp. 147–152.

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

[10] V. Hollink, T. Tsikrika, and A. Vries. The semantics of query modification. In *Proc. of RIAO 2010*, pp. 176–181.

[11] V. Hollink, T. Tsikrika, and A. Vries. Semantic search log analysis: A method and a study on professional image search. *JASIST*, 62(4):691–713, 2011.

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

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

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

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

[16] A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais, and J. Teevan. Modeling and analysis of cross-session search tasks. In *Proc. of SIGIR 2011*, pp. 5–14.

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

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

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

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

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

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

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

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