Identifying Task-based Sessions in Search Engine Query Logs

Claudio Lucchese‡, Salvatore Orlando†,
Raffaele Perego‡, Fabrizio Silvestri‡, Gabriele Tolomei‡†
‡ISTI-CNR, Pisa, Italy – e-mail: {firstname.lastname}@isti.cnr.it
†Dip.to di Informatica, Università Ca' Foscari Venezia, Italy – e-mail: {lastname}@dsi.unive.it

ABSTRACT

The research challenge addressed in this paper is to devise effective techniques for identifying *task-based sessions*, i.e. sets of possibly non contiguous queries issued by the user of a Web Search Engine for carrying out a given *task*. In order to evaluate and compare different approaches, we built, by means of a manual labeling process, a *ground-truth* where the queries of a given query log have been grouped in tasks. Our analysis of this ground-truth shows that users tend to perform more than one task at the same time, since about 75% of the submitted queries involve a multi-tasking activity. We formally define the *Task-based Session Discovery Problem* (TSDP) as the problem of best approximating the manually annotated tasks, and we propose several variants of well known clustering algorithms, as well as a novel efficient heuristic algorithm, specifically tuned for solving the TSDP. These algorithms also exploit the collaborative knowledge collected by *Wiktionary* and *Wikipedia* for detecting query pairs that are not similar from a lexical content point of view, but actually *semantically* related. The proposed algorithms have been evaluated on the above ground-truth, and are shown to perform better than state-of-the-art approaches, because they effectively take into account the multi-tasking behavior of users.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering, Query formulation, Search process*

General Terms

Algorithms, Design, Experimentation

Keywords

Query log analysis, Query log session detection, Task-based session, Query clustering, User search intent

WSDM'11, February 9–12, 2011, Hong Kong, China.

1. INTRODUCTION

There is a common belief that the Web is increasingly used to simplify the accomplishment of various everyday activities. Since nowadays Web Search Engines (WSEs) are the most important and used Web portals, such users' behaviors can be revealed by analyzing and mining WSE query logs [2, 26, 14, 8, 19, 30]. A very important piece of information we can extract from a query log is represented by "*query sessions*", i.e. specific sets/sequences submitted by a user while interacting with a WSE. Sessions represent the basic unit of information for tasks like query suggestion [1], learning to rank [23], enhancing interactions with the Web Search Engine [34], etc.

In the literature, there are many definitions of query sessions. In this paper, we are interested in identifying sessions composed of queries issued by users having in mind a particular task/goal [31]. Unfortunately, the well-known time-based detection methods fail in revealing such *task-based sessions*, i.e., *Web-mediated tasks*, due to the *multi-tasking* users' behavior. Multi-tasking refers to the way users interact with a WSE, by intertwining different tasks within the same time period. Therefore, the extraction of such task-based sessions requires to detect whether pairs of users' queries are *similar* and, thus, related to the same task/goal.

The main contributions of our work are the following.
(i) We start by showing that users perform multi-tasking search activities in the query streams issued to a WSE. This makes it unsuitable to identify task-based sessions by only exploiting techniques that simply split the stream of queries. Then, we investigate three well-known *clustering-based* approaches and we propose a new heuristic for detecting Web-mediated tasks. The obtained results show that the new algorithm performs similarly to the best clustering-based approach (i.e., weighted connected components) but it is computationally lighter on average.
(ii) We use a *query distance function*, exploited by those algorithms, which combines classical lexical content distance measures, with the collaborative knowledge provided by *Wiktionary*[1] and *Wikipedia*[2]. This knowledge is used to enrich the meaning of each issued query and, thus, to make more accurate decisions during clustering.
(iii) Finally, we compare and evaluate the quality of all those methods by exploiting a manually generated *ground-truth*, i.e. a set of task-based sessions manually detected over the queries submitted by several users.

[1] http://www.wiktionary.org
[2] http://www.wikipedia.org

2. RELATED WORK

Analysis of query logs collected by most Web Search Engines (WSEs) has increasingly gained interest across Web mining research community. Roughly, query logs record information about the *search activities* of users and so they are a suitable source of information for understanding how people search the Web or, in other words, the real intent behind issued queries [30].

Previous work on session identification can be classified into: 1) *time-based*, 2) *content-based*, and 3) *mixed-heuristics*, which usually combine both 1) and 2).

1) Time-based. Usually, time-based techniques have been adopted for their simplicity in previous research work. Silverstein *et al.* [29] firstly defined a concept of "*session*" as follows: two consecutive queries are part of the same session if they are issued at most within a 5-minutes time window. According to this definition, they found that the average number of queries per session in the data they analyzed was 2.02. He and Göker [6] used different timeouts to split user sessions of Excite query log, ranging from 1 to 50 minutes. Radlinski and Joachims [23] observed that users often perform a sequence of queries with a similar information need, and they referred to those sequences of reformulated queries as *query chains*. Their paper presented a method for automatically detecting query chains in query and click-through logs using 30 minutes threshold for determining if two consecutive queries belong to the same search session.
Another definition of session, i.e. *search episode*, was given by Jansen and Spink [8]. They described a session as the period of time occurring from the first to the last recorded time-stamp on the WSE server from a particular user in a single day, so that session length might vary from less than a minute to a few hours. Moreover, using the same concept of search episode, Spink *et al.* [31] investigated also *multi-tasking* behaviors while users interacting with a WSE.
In this paper, we show the presence of multi-tasking also within shorter user activities.

2) Content-based. Some work suggested to exploit the lexical content of the query themselves for determining a possible topic shift in the stream of issued queries and, thus, a session boundary [12, 7, 20]. To this extent, several *search patterns* have been proposed by means of lexical comparison, using different string similarity scores (e.g., Levenstein, Jaccard, etc.). However, approaches relying only on content features suffer of the so-called *vocabulary-mismatch problem*, namely the existence of topically-related queries without any shared terms. In order to overcome this issue, Shen *et al.* [28] compared "expanded representation" of queries, instead of the actual queries themselves. Each individual expanded query was obtained by concatenating the titles and the Web-snippets for the top 50 results provided by a WSE for the specific query. Thus, the relatedness between query pairs was computed using cosine similarity between the corresponding expanded queries.

3) Mixed heuristics. Jansen *et al.* [9] assumed that a new search pattern always identifies the start of a new session. Moreover, He *et al.* [7] showed that statistical information collected from query logs could be used for finding out the probability that a search pattern actually implies a session boundary. In particular, they extended their previous work [6] to consider both temporal and lexical information. Boldi *et al.* [1] introduced the *query-flow graph* as a model for representing data collected in WSE query logs. They exploited this model for segmenting the query stream into sets of related information-seeking queries, leveraging on an instance of the Asymmetric Traveling Salesman Problem.
Finally, Jones and Klinkner [11] addressed a problem that appears to be similar to ours. In particular, they argue that within a user's query stream it is possible to recognize particular hierarchical units, i.e., *search missions*, which are in turn subdivided into disjoint *search goals*. A search goal is defined as an atomic information need, resulting in one or more queries, while a search mission is a set of topically-related information needs, resulting in one or more goals. Given a manually generated ground-truth, Jones and Klinkner [11] investigated how to *learn* a suitable binary classifier, which is aimed to precisely detect whether two queries belong to the same task or not. Among various results, they realized that timeouts, whatever their lengths, are of limited utility in predicting whether two queries belong to the same goal, and thus to identify session boundaries. Indeed, authors did not to explore how such binary classifier could be exploited for actually segmenting users' query streams into goals and missions.

3. THEORETICAL MODEL

A WSE query log stores queries submitted by users, along with other information, such as userIDs, time-stamps, etc. We denote with $\mathcal{QL}$ a WSE log of the queries submitted by a set of users $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ during a given observation period. Moreover, let $q_i \in \mathcal{QL}$ be a generic query issued by user u_i, and $q_{i,j} \in \mathcal{QL}$ be the j-th query issued by user u_i.

The methods that extract meaningful user sessions from $\mathcal{QL}$ have to analyze all the queries issued by each user u_i. Let $\mathcal{S}_i$ be the sequence of *all* the queries $q_i \in \mathcal{QL}$ issued by user $u_i \in \mathcal{U}$, chronologically ordered during the period of observation recorded in the query log: $\mathcal{S}_i = \langle q_{i,1}, q_{i,2}, \ldots, q_{i,K} \rangle$. Therefore,

$$\mathcal{QL} = \bigcup_{i=1}^{N} \mathcal{S}_i$$

Since users tend to issue *bursts* of queries for relatively short periods of time, which are usually followed by longer periods of inactivity, the *time gap* between queries plays a significative role in detecting session boundaries. According to [29], we detect the session boundaries by considering the user's inactivity periods, i.e. the time gaps between consecutive queries in each $\mathcal{S}_i$.

DEFINITION 3.1 (TIME-GAP SESSION $\phi_{i,k}$). *Let $\tau(q_{i,j})$ bet the time at which the query $q_{i,j}$ is issued, and t_ϕ be the maximum* time gap *threshold. The ordered set of consecutive queries $\phi_{i,k} = \langle q_{i,s_k}, \ldots, q_{i,e_k} \rangle \subseteq \mathcal{S}_i$, with $s_k < e_k$, is said to be a* time-gap session *if it holds that:* (i) $\tau(q_{i,j+1}) - \tau(q_{i,j}) \leq t_\phi$ *for every j, $s_k \leq j < e_k$, and* (ii) *there is no time-gap session being a superset of $\phi_{i,k}$.* □

It is worth noticing that this splitting technique makes no restrictions on the total elapsed time between the first and the last query of the sequence $\phi_{i,k}$. Moreover, even if the inactivity threshold is usually fixed arbitrarily, in our tests we set $t_\phi = 26$ minutes by analyzing the distribution of the time gaps in the query log $\mathcal{QL}$ used for the experiments (see Section 4.1).

In this paper, we are interested in studying to which extent in such time-gap sessions we can further recognize *task-based sessions*, i.e. sets of queries aimed at performing some Web-mediated tasks. Queries within the same task-based session do not necessarily occur consecutively in the time-gap session $\phi_{i,k}$. Indeed, we will show that a generic user u_i usually interleaves many different information needs and related queries in each $\phi_{i,k}$.

DEFINITION 3.2 (TASK-BASED SESSION $\theta^j_{i,k}$). *Let $\phi_{i,k}$ be a time-gap session included in $\mathcal{S}_i$, and let $\theta^j_{i,k} \subseteq \phi_{i,k}$ be a* task-based session*, i.e., a set of (not necessarily consecutive) queries issued by user u_i for performing a given Web-mediated task. Such tasks form a disjoint partitioning of a time-gap session.* □

We denote with $\Theta_{i,k} = \cup_j \theta^j_{i,k}$ all the task-based sessions in a given time-gap session $\phi_{i,k}$, and with $\Theta = \cup_{i,k}\Theta_{i,k}$ the set of all the task-based sessions in the query log $\mathcal{QL}$, i.e. the set-union of $\Theta_{i,k}$ for all users i and associated time-gap sessions k.

The problem of finding Θ in a given query log can thus be formulated as the *Task-based Session Discovery Problem* (TSDP), whose goal is to find the best *query partitioning strategy* π that, when used to segment each time-gap session $\phi_{i,k}$ in $\mathcal{QL}$, approximates the *actual* user task-based sessions $\Theta_{i,k}$.

DEFINITION 3.3 (TSDP). *Given a query log $\mathcal{QL}$, let $\mathcal{C}_{i,k} = \{c^1_{i,k}, c^2_{i,k}, \ldots\}$ be the task-based sessions determined by the* query partitioning strategy *π, when applied onto $\phi_{i,k}$, i.e. $\pi(\phi_{i,k}) = \mathcal{C}_{i,k}$. Let $\Theta = \cup_{i,k}\Theta_{i,k}$ and $\mathcal{C}_\pi = \cup_{i,k}\mathcal{C}_{i,k}$. The TSDP requires to find the best partitioning $\bar{\pi}$ such that*

$$\bar{\pi} = \arg\max_{\pi} \xi(\Theta, \mathcal{C}_\pi)$$

where ξ is a given function that measures the quality of partitioning $\mathcal{C}_\pi$ with respect to Θ. □

Several quality measures can be used to evaluate the accuracy of a task-based session extraction, and consequently, several ξ functions can be devised. In Section 6 we instantiate ξ in terms of *F-measure*, *Rand index* and *Jaccard index*.

4. DATA ANALYSIS

We used the 2006 *AOL* query log as our testing data set. This query log is a very large and long-term collection consisting of about 20 million of Web queries issued by more than 657000 users over 3 months (from 03/01/2006 to 05/31/2006)[3].

First of all, we removed query log records containing both empty and "non-sense" query strings (e.g., query strings composed of only punctuation symbols). Also, we removed all the *stop-words* from each query string. Then, we run the *Porter stemming algorithm* [21] for removing the most common morphological and inflexional English endings from the terms of each query string. Finally, the data cleaning phase involved removing the long-term user sessions containing too much queries, which were probably generated by *robots*, instead of human users. Then, we considered as a sample the 1,000 user sessions with the highest number of queries (*top*-1000).

[3] http://sifaka.cs.uiuc.edu/xshen/aol_querylog.html

Hence, according to Def. 3.1, we split each user session into several time-gap sessions. To this end, we had to devise a suitable time threshold t_ϕ, which can be obtained by analyzing the distribution of time gaps between all the consecutive query pairs in our data set. We divided all the time gaps into several *buckets*, 60-seconds each. Therefore, we analyzed the query inter-arrival times distribution, which is revealed to be a power-law (Fig. 1(a)).

This model tightly fits user behaviors during Web search activities, when consecutive queries issued within a short period of time are often not independent because they are also task-related.

More formally, given the following general form of a power-law distribution $p(x)$,

$$p(x) = \frac{\alpha - 1}{x_{min}} \left(\frac{x}{x_{min}} \right)^{-\alpha}$$

where $\alpha > 1$ and x_{min} is the minimum value of x from which the law holds, we were interested in finding the value $\bar{x}$, such that two consecutive queries whose time gap is smaller than $\bar{x}$ are considered to belong to the same time-gap session. When the underlying distribution is unknown, it makes sense to assume a Gaussian distribution and use a threshold $\bar{x} = \mu + \sigma$ being equal to mean μ plus standard deviation σ, which results in "accepting" $\lambda = 84.1\%$ of the samples. This is equivalent to consider the cumulative distribution $P(\bar{x}) = Pr(X \leq \bar{x})$ and to determine $\bar{x}$, such that $P(\bar{x}) = \lambda$. Since we know the underlying distribution, we map the threshold λ into our context as follows:

$$P(\bar{x}) = C \int_{-\infty}^{\bar{x}} p(X)\, dX = \frac{\alpha - 1}{x_{min}^{-\alpha+1}} \int_{-\infty}^{\bar{x}} X^{-\alpha}\, dX = \left(\frac{\bar{x}}{x_{min}} \right)^{-\alpha+1}$$

Hence, for our purpose we had to solve the following equation w.r.t. $\bar{x}$:

$$P(\bar{x}) = \left(\frac{\bar{x}}{x_{min}} \right)^{-\alpha+1} = \lambda = 0.841 \qquad (1)$$

The value x_{min} represents the minimum query pair time gap and corresponds to the first interval, i.e., 60 seconds. Therefore, we estimated $\alpha = 1.58$ and finally we can solve Eq. 1 finding $\bar{x} \simeq 26$ minutes. This means to assume 84.1% of consecutive query pairs are issued within 26 minutes. We used this value, $\bar{x}$, as the threshold t_ϕ for splitting each long-term user session of the the query log.

4.1 Ground-truth construction

In order to approach the Task-based Session Discovery Problem, according to our Def. 3.3, we need to find the query partitioning strategy that best approximates the *actual* task-based segmentation. Such optimal task-based partitioning can be manually built from real WSE query log data. To this end, we developed a Web application that helps human assessors to manually identify the optimal task-based query sessions from the previously prepared AOL query log, thus producing a *ground-truth* that can be used for evaluating automatic task-based session discovery methods.

Human annotators grouped together queries that they claimed to be task-related within each time-gap session. Also, they had chance to discard meaningless queries from those sessions. For each manually identified task (i.e., set of task-related queries), evaluators had to add a *tag* and, optionally,

a longer description. Such data source could possibly represent a semantic knowledge base of users search goals (i.e., taxonomy of tasks).

Since long-term sessions in AOL query log were too long, we only consider the first week of activities for each *top*-1000 user session. Finally, human evaluators were people selected from our laboratory, but not directly involved in this work.

4.2 Ground-truth statistics

Manual annotation procedure concerned a total of 2,004 queries, from which 446 time-gap sessions were extracted automatically. A total of 139 time-gap session were discarded as meaningless by the annotators, and therefore they were removed from the ground-truth. Eventually, 1,424 queries were actually clustered from 307 time-gap sessions.

Fig. 1(b) shows the distribution of time-gap session length, using a discretization factor of 60 seconds. While there are many session being less than 1 minute long, probably short sessions with one or two queries, the average length of a time-gap session is about 15 minutes. It is not infrequent to have sessions lasting for 40 minutes. Also in this case, the length of these sessions suggests that the interaction of the user with the Web Search Engine is non trivial, and it is likely to involve multi-tasking. Finally, the longest time-gap session lasts 9207 seconds, i.e. about 2 hours and a half.

In Fig. 1(c) we report the time-gap session size distribution. On average, each time-gap session contains 4.49 queries, the sessions with at most 5 queries cover slightly more than half of the query log. The other half of the query log contains longer sessions with high probability of having multiple tasks being carried on in the same session.

The total number of human annotated task-based sessions is 554, with an average of 2.57 queries per task. The distribution of the task-based sessions size is illustrated in Fig. 1(d). The number of tasks accomplished in a time-gap session is 1.80, see Fig. 1(e). In particular, only 162 out of 307 time-gap sessions contain one task only. We found that this 50% split between single-tasking and multi-tasking sessions is consistent across the various users. Interestingly enough, this shows that a good algorithm should be able to handle efficiently both single- and multi-tasking sessions. If we consider the queries included in each task, then 1,046 out of 1,424 queries are included in multi-tasking sessions, meaning that about 74% of the user activity is multi-tasking.

Finally, we also evaluated the degree of multi-tasking by taking into account the number of overlapping task-based sessions. We say that a *jump* occurs whenever two queries in a manually labelled task-based session are not consecutive. For instance, let $\phi = \langle q_1, q_2, \ldots, q_9 \rangle$ be a time-gap session and let $\pi(\phi) = \{\theta_1, \theta_2, \theta_3\}$ be the result of the manual annotation procedure for ϕ, where $\theta_1 = \{q_1, q_2, q_3, q_4\}$, $\theta_2 = \{q_5, q_7\}$, and $\theta_3 = \{q_6, q_8, q_9\}$. In this case, the number of jumps observed in ϕ is 2, because there are two query pairs $(q_5, q_7) \in \theta_2$ and $(q_6, q_8) \in \theta_3$, which are not consecutive. The number of jumps gives a measure of the *simultaneous* multi-tasking activity. We denote with $j(\phi)$ the simultaneous multi-tasking degree of ϕ as the ratio of task-based sessions in ϕ having at least one jump. In the previous example $j(\phi) \simeq 0.67$, since 2 out of 3 tasks contain at least one jump. In Fig. 1(f), we show the distribution of the multi-tasking degree over all the time-gap sessions. Note that the result for $j(\phi) = 0$ is omitted, because we already know that 50% of the sessions are single-tasking.

5. SESSION DISCOVERY METHODS

In this section, we address the Task-based Session Discovery Problem (TSDP) introduced in Section 3 by proposing and comparing several approaches and techniques. We group the session discovery mechanisms into two broad families: *(i) TimeSplitting-t* and *(ii) QueryClustering-m*.
Basically, *TimeSplitting-t* consists of splitting each session when the time between two query submissions is greater than a threshold t. Besides, *QueryClustering-m* aims to detect task-based sessions using a given clustering method m.

5.1 TimeSplitting-t (TS-t)

Intuitively, the simplest techniques for identifying sets of task-related queries from a WSE's log take *only* into account query submission time. Time splitting techniques simply break the stream of queries as long as the time gap between two adjacent queries is greater than a certain threshold t. This is based on the assumption that if two consecutive queries are far away enough than they are also likely to be unrelated. Note that time splitting techniques differ one from each other only for the actual value of t.

According to Def. 3.1, time splitting techniques are used for detecting time-gap sessions. In particular, we use a time threshold $t = 26$ minutes for identifying time-gap sessions of our query log (i.e., TS-26). This threshold have been figured out from the testing data set using the methodology described in Section 4.

Moreover, according to Def. 3.3, TSDP requires to find the best partitioning strategy over all the time-gap sessions available in the query log. A trivial partitioning strategy is the one that simply consider each time-gap session as a task-based session. In our case, this is equivalent to use only TS-26 for addressing the TSDP. However, other partitioning strategies might be figured out simply by applying different time splitting techniques to each identified time-gap session. In this regard, there are several time thresholds that have been extensively proposed in literature [29, 6]. In this work, we used TS-5 and TS-15, i.e., 5 and 15 minutes thresholds, respectively.

The main drawback of time splitting methods is that they are unable to properly deal with multi-tasking sessions, since identified sets of task-related queries are actually composed of temporarily ordered consecutive queries. Moreover, according to the analysis we provided in Section 4.2, multi-tasking sessions represent a significative sample of the total available sessions, at least for our testing data set.

In Section 6, we compare the results provided by TS-5, TS-15, and TS-26. Also, we show that they alone are not suitable for identifying task-based sessions.

5.2 QueryClustering-m (QC-m)

We study three algorithms derived from well-known clustering methods: QC-Means, QC-Scan, and QC-wcc. Moreover, we propose a novel algorithm as a variation of QC-wcc, named QC-htc. All clustering algorithms have been applied to time-gap sessions, which in turn have been preliminary identified using TS-26 time splitting technique.

As for any other clustering problem, most important choices involve both the features selected for computing the *distance function* used by the algorithms and how such features might be composed, as we show in the following.

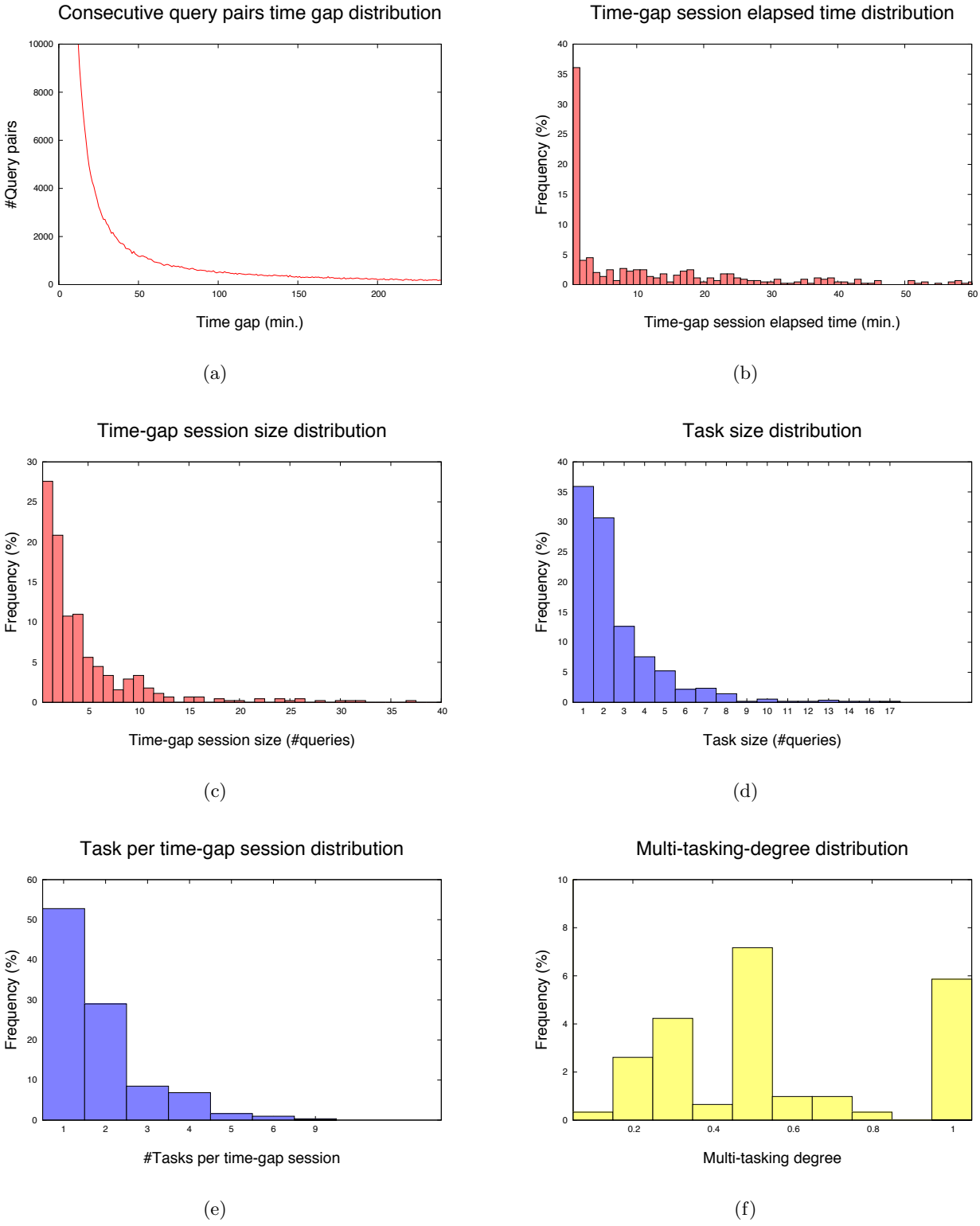

Figure 1: Statistics about the ground-truth data.

5.2.1 Feature Selection

Evaluating the similarity between two queries is a very complex issue. Most of the previous approaches are based on distance between query lexical content [27]. The precision of those approaches results to be quite low due to the short length of queries [29] and the lack of the contextual information in which queries are issued [33]. Thus, some approaches try to expand those short queries by exploiting resulting URLs returned by WSEs [5], or the returned Web-snippets [16], or the documents themselves [24]. Two queries might be considered similar if they return similar results, or similar documents. Unfortunately, it might be the case for unrelated queries to share some results.

In our work, we propose two features and two similarity measures for assessing the relatedness of two queries both in terms of their lexicographical content and their semantics.

Content-based ($\mu_{content}$). Two queries that share some common terms are likely related. Sometime, such terms may be very similar, but not identical, due to mispelling, or different prefixes/suffixes. To capture content distance between queries, we adopt a Jaccard index on tri-grams [10]. Let $T(q)$ be the tri-grams resulting from the terms of query q, we define the distance $\mu_{jaccard}$ as follows:

$$\mu_{jaccard}(q_1, q_2) = 1 - \frac{|T(q_1) \cap T(q_2)|}{|T(q_1) \cup T(q_2)|}.$$

In addition, we exploit a normalized Levenstein distance $\mu_{levenstein}$, which Jones and Klinkner [11] claimed to be the best edit-based feature for identifying goal boundaries. Finally, the overall content-based distance is computed as follows:

$$\mu_{content}(q_1, q_2) = \frac{(\mu_{jaccard} + \mu_{levenstein})}{2}.$$

Semantic-based ($\mu_{semantic}$). We are interested in finding a measure of the *semantic relatedness* between query pairs. Typically, humans can easily judge the semantic relatedness between two terms. This human ability is backed by their experience and knowledge, which makes it a hard task for machines. If a machine should solve this task, it also needs some source of knowledge. Usually, this knowledge comes from: *(i)* large text collections (i.e., *corpora*) or from *(ii)* semantic resources. Thus, we figured out that we could expand each query with its "*wikification*". Basically, we exploit both Wiktionary and Wikipedia data sources for increasing the meaningfulness of each query, trying to overcome its lack of semantic information.

Several semantic relatedness metrics dealing with semantic resources have been proposed in the past. They can be classified into: *(i) path-based*, in which knowledge is modeled as a graph of concepts and the metrics rely on paths over that graph [22, 13], *(ii) information content-based* that takes into account the information content of a concept [25], *(iii) gloss-based*, which is based on term overlaps between definitions of concepts [15], and *(iv) vector-based* that models each concept as a vector of anchor links [18] or terms [4].

Following the last approach, we assume that a Wiktionary or a Wikipedia article describes a certain concept and that the presence of a term in a given article is an evidence of the correlation between that term and that concept. Thus, we describe the *wikification* $\overrightarrow{C}(t)$ of a term t as its representation in a high dimensional concept space $\overrightarrow{C}(t) = (c_1, c_2, \ldots, c_W)$, where W is the number of articles in our collections and c_i scores the relevance of the term t for the i-th article. We measure this relevance by using the well known *tf-idf* score [27].

In order to "*wikify*" the whole string associated with a query q, we sum up the contribution from its terms, i.e.:

$$\overrightarrow{C}(q) = \sum_{t \in q} \overrightarrow{C}(t).$$

Then, we compute the relatedness $rel(q_1, q_2)$ between two queries as the cosine of their corresponding concept vectors:

$$rel(q_1, q_2) = \frac{\overrightarrow{C}(q_1) \cdot \overrightarrow{C}(q_2)}{|\overrightarrow{C}(q_1)||\overrightarrow{C}(q_1)|}.$$

Thus, the distance score $\mu_{wikification}$ can be written as follows:

$$\mu_{wikification}(q_1, q_2) = 1 - rel(q_1, q_2).$$

Of course, we use the same approach both for computing $\mu_{wiktionary}$ and $\mu_{wikipedia}$ distances, taking into account Wiktionary and Wikipedia corpora, respectively. Finally, the overall semantic-based distance is obtained as follows:

$$\mu_{semantic}(q_1, q_2) = min(\mu_{wiktionary}, \mu_{wikipedia}).$$

5.2.2 Distance Function

An immediate way to put together *(i)* the lexical content ($\mu_{content}$) and *(ii)* the semantic expansion ($\mu_{semantic}$) distance is via a *convex combination*:

$$\mu_1 = \alpha \cdot \mu_{content} + (1 - \alpha) \cdot \mu_{semantic} \quad (2)$$

In addition, we propose a novel *conditional* distance function μ_2 based on the following heuristic: if the content-based distance between two queries does not exceed a certain threshold then we can be confident that queries are also task-related; otherwise, we look at the semantic expansion of the queries and we compute the final distance score as the minimum between content-based and semantic-based distance values.

$$\mu_2 = \begin{cases} \mu_{content} & \text{if } \mu_{content} < \mathbf{t} \\ \min(\mu_{content}, \mathbf{b} \cdot \mu_{semantic}) & \text{otherwise.} \end{cases} \quad (3)$$

Both μ_1 and μ_2 relies on the estimation of some parameters, i.e., α for μ_1 and $\mathbf{t}$, and $\mathbf{b}$ for μ_2. We exploited the ground-truth for learning such parameters but here we do not show the whole procedure we followed because it is not part of the contributions we wanted to highlight in this work.

The rationale for introducing the *conditional* distance function μ_2 is the following: we conjecture that if two queries are close in term of lexical content, the semantic expansion could be unhelpful. Vice-versa, nothing can be said when queries do not share any common content feature (e.g., consider the two queries "kobe bryant" and "nba").

5.2.3 Algorithms

In the following we describe four clustering algorithms that exploit the above functions. The first three are insipired to well-known clustering algorithms, while the last is a novel algorithm aiming at reducing the computational cost of the clustering.

QC-Means. QC-MEANS is a *centroid-based* algorithm and represents a variation of the well-known K-MEANS [17]. We replaced the usual K parameter, i.e. the number of clusters to be extracted, with a ρ threshold that defines the maximum radius of a centroid-based cluster. This allows us to better deal with the variance in length of the various user sessions as well as to avoid specifying the number of final clusters *apriori*.

QC-Scan. QC-SCAN is the *density-based* DB-SCAN algorithm [3], specifically tailored for extracting task-based sessions from WSE query logs. The rationale for evaluating also a variation of DB-SCAN is that centroid-based approach may suffer the presence of noise in query logs.

QC-wcc. QC-WCC, query clustering based on *weighted connected components*, is a *graph-based* algorithm. Given a time-gap based session ϕ, it builds a complete graph $G_\phi = (V, E, w)$, whose nodes V are the queries in ϕ, and whose E edges are weighted by the similarity of the corresponding nodes. The weighting function w is a similarity function $w : E \longmapsto \mathbb{R} \in [0, 1]$ that can be easily instantiated in terms of the distance functions μ_1 or μ_2, described in the previous section (i.e., $w = 1 - \mu_1$ or $w = 1 - \mu_2$). The graph G_ϕ describes the similarity between any pair of queries in the given tim-gap based session.
The rationale of QC-WCC is to drop *weak edges*, i.e. with low similarity, since the corresponding queries are not related, and to build clusters on the basis of the *strong edges*, i.e. with high similarity, which identify the related query pairs. The algorithm preforms two steps. In the first step, given the graph G_ϕ all the edges $e \in E$ whose weight is smaller than a given threshold, that is $w(e) < \eta$, are removed, thus obtaining a pruned graph G'_ϕ. In the second step, the connected components of G'_ϕ are extracted. Such connected components identify the clusters of related queries which are returned by the algorithm.
Note that this step could have been accomplished by adopting the classification-based method proposed by Jones and Klinkner [11]. However, our aim was to detect task-based sessions using heuristics that extract suitable features directly from the data we had, thus avoiding any kind of preliminary step, which involves the training of a classifier.
Indeed, assuming a robust similarity function, the QC-WCC algorithm is able to handle the multi-tasking nature of users sessions. Groups of related queries are isolated by the pruning of weak edges, and links with large similarity identify the generalization/specialization steps of the users, as well as restarts from a previous query when the current query-chain is found to be unsuccessful.
The cost of QC-WCC is dominated by the construction of the graph G_ϕ. Indeed, the similarity between any pair of edges must be computed, resulting in a number of similarity computations quadratic in the number of nodes, i.e. queries, in the session.

QC-htc. In this section, we propose QC-HTC, query clustering based on *head-tail* components, a variation of the connected components based algorithm, which does not need to compute the full similarity graph. Since queries are submitted one after the other by the user, the QC-HTC algorithms takes advantage of this sequentiality to reduce the number of similarity computations needed by QC-WCC. The algorithm works in two phases as follows.
The first step aims at creating an approximate fine-grained clustering of the given time-gap session $\phi = \langle q_1, q_2, \ldots, q_m \rangle$. Every single Web-mediated task generates a sequence of queries. Due to the multi-tasking behavior of users, multiple Web-mediated tasks are carried on at the same time, and the query log records such overlapping tasks, and the corresponding query sequences. As a consequence, each Web-mediated task is observed as a set of *fragments*, i.e. smaller sets of consecutive queries, and fragments of different tasks are interleaved in the query log because of multi-tasking.
The algorithm exploits the sequentiality of user queries, and tries to detect the above fragments, by partitioning the given time-gap session into *sequential clusters*, where a sequential cluster, denoted with $\tilde{c}^i$, must contain only queries that occur in a row within the query log, and such that each query is sufficiently similar to the chronologically following one. Since we are only focussing on the similarity between one query and the next, the detection of such *sequential clusters* can be done in linear time.
The second step of the algorithm merges together those fragments when they are related, trying to overcome the interleaving of different tasks. Here we introduce another assumption that reduces the computational cost of the algorithm. We assume that a cluster of queries can be described well by just the chronologically first and last of its queries, respectively denoted with $head(\tilde{c}^i)$ and $tail(\tilde{c}^i)$. Therefore, the similarity s between two clusters $\tilde{c}^i$,$\tilde{c}^j$ is computed as:

$$s(\tilde{c}^i, \tilde{c}^j) = \min_{\substack{q \in \{head(\tilde{c}^i), tail(\tilde{c}^i)\} \\ p \in \{head(\tilde{c}^j), tail(\tilde{c}^j)\}}} w(e(q, p))$$

where w weights the edge $e(q, p)$ linking the queries p and q on the basis of their similarity, analogously to QC-WCC.
The final clustering is produced as follows. The first cluster c^1 is initialized with the oldest sequential cluster $\tilde{c}^1$, which is removed from the set of sequential clusters. Then, c^1 is compared with any other sequential cluster $\tilde{c}^i$ (ordered chronologically) by computing the similarity as above. Given a threshold η, if $s(c^1, \tilde{c}^i) < \eta$, then $\tilde{c}^i$ is merged into c^1, the *head* and *tail* queries of c^1 are updated consequently, and $\tilde{c}^i$ is removed from the set of sequential clusters. The algorithm continues comparing the new cluster c^1 with the remaining sequential clusters. When all the sequential clusters have been considered, the oldest sequential cluster available is used to build a new cluster c^2.
The algorithm iterates this procedure until no more sequential clusters are left.
In the worst case, the complexity of QC-HTC is still quadratic in the number of queries. However, there are frequent cases in which the complexity is much smaller. We have seen that 52.8% of the time-gap bases sessions contain one task only. In this case, it is very likely that this task is found after the first step of the algorithm, if each query is sufficiently similar to the next one, with a cost that is only linear in the number of nodes. For multi-tasking sessions, the complexity in the second step is quadratic in the number of sequential clusters extracted. Also in this case, the cost reduction may be significant. Suppose that the number of sequential clusters is $\rho|\phi|$, with $0 \leq \rho \leq 1$, then the complexity of the algorithm is $O(\rho^2|\phi|^2)$. Suppose that the number of sequential clusters is half the number of queries, than the algorithm is four times cheaper than QC-WCC. Still, this is an upper bound of the cost, since the QC-HTC algorithm does not compute the pair-wise similarities among sequential clusters in advance.

6. EXPERIMENTS

In this section we analyze and compare the results obtained with all the methods we described in Section 5 for approaching the TSDP. Moreover, we compare our results with the ones provided by two other methods: *(i)* the simple time splitting technique TS-26, which is considered as the *baseline* solution and *(ii)* the session extraction method based on the *query-flow graph* model proposed by Boldi *et al.* [1], which represents the state of the art.

6.1 Measures of validity

In order to evaluate and compare all the methods we mentioned, we need to measure the degree of correspondence between manually-extracted tasks of the ground-truth and tasks produced by our algorithms. To this end, we use both *classification-* and *similarity-oriented* measures [32]. In the following, we thus refer to "*predicted classes*" as the task-based sessions detected by a specific algorithm, whereas the "*actual classes*" just correspond to the task-based sessions of the ground-truth.

Classification-oriented approaches measure the degree to which predicted classes correspond to actual classes and *F-measure* is one of the most popular measure in this category. It combines both *precision* and *recall*: precision measures the fraction of a task that consists of objects of a specified class, while recall measures the extent to which a task contains all objects of a specified class. Thus, globally F-measure evaluates the extent to which a task contains *only* objects of a particular class and *all* objects of that class. Given $p(i,j)$, $r(i,j)$ the precision and recall of task i with respect to class j respectively, the F-measure is computed as: $F(i,j) = \frac{2 \times p(i,j) \times r(i,j)}{p(i,j)+r(i,j)}$.

We have to compute F-measure for each task-based session detected by an algorithm, where discarded queries are considered as singleton tasks (i.e., tasks containing only one query). Finally, an overall F-measure value is computed by a weighted average over all the tasks, by considering the size of each task as weight.

Besides, similarity-oriented measures consider pairs of objects instead of single objects. Hence, given f_{00} = number of pairs of objects having a different class and a different task, f_{01} = number of pairs of objects having a different class and the same task, f_{10} = number of pairs of objects having the same class and a different task, and f_{11} = number of pairs of objects having the same class and the same task, two measures are defined: *(i) Rand index* $R = \frac{f_{00}+f_{11}}{f_{00}+f_{01}+f_{10}+f_{11}}$ and *(ii) Jaccard index* $J = \frac{f_{11}}{f_{01}+f_{10}+f_{11}}$.

When computing both Rand and Jaccard index we did not consider time-gap sessions containing only one singleton task. Anyway, we still took into account time-gap sessions that are composed of a single task with more than one query.

6.2 Evaluation

TimeSplitting-t. In this work, we compare three different time splitting techniques: TS-5, TS-15, and TS-26, which use 5, 15, and 26 minutes thresholds, respectively. Tab. 1 shows the results we obtained using those techniques on the ground-truth. The best result in terms of F-measure was found considering the whole time-gap sessions identified with TS-26, without additionally splitting them into shorter time-gap sessions.

Hence, we consider TS-26 as the *baseline* approach for addressing the TSDP. Roughly, according to Def. 3.1 and Def. 3.2, this is equivalent to identify task-based sessions with time-gap sessions.

Table 1: TS-5, TS-15, and TS-26.

	F-measure	Rand	Jaccard
TS-5	0.28	**0.75**	0.03
TS-15	0.28	0.71	0.08
TS-26	**0.65**	0.34	**0.34**

Query Flow Graph. In order to better evaluate our proposed approaches we decided to compare them to the *query-flow graph* (QFG) presented by Boldi *et al.* [1].

QFG has been constructed over a training segment of the AOL *top*-1000 user sessions. This method uses *chaining probabilities* measured by means of a machine learning method. The initial step was to extract those features from the training log, and storing them into a compressed graph representation. In particular, we extracted 25 different features (i.e., time-related, session and textual features) for each pair of queries (q, q') that are consecutive in at least one session of the query log.

The validity of QFG has been tested on the ground-truth and the results we obtained are showed in Tab. 2. We found the best values using a threshold $\eta = 0.7$. In fact, results do not improve using a greater threshold value.

QFG significantly improved the baseline TS-26. In particular, F-measure is improved with a gain of $\approx 16\%$. Furthermore, QFG gained $\approx 52\%$ in terms of Rand and $\approx 15\%$ in terms of Jaccard.

Table 2: QFG: varying the threshold η.

	η	F-measure	Rand	Jaccard
QFG	0.1	0.68	0.47	0.36
	0.2	0.68	0.49	0.36
	0.3	0.69	0.51	0.37
	0.4	0.70	0.55	0.38
	0.5	0.71	0.59	0.38
	0.6	0.74	0.65	0.39
	0.7	**0.77**	**0.71**	**0.40**
	0.8	0.77	0.71	0.40
	0.9	0.77	0.71	0.40

QueryClustering-m. We now compare all the clustering methods described in Section 5.2.3.

We start evaluating QC-Means algorithm using both the distance functions μ_1 and μ_2. We empirically set the radius of this centroid-based algorithm to 0.4 for both distance functions and the results are showed in Tab. 3.

Concerning μ_1, the best results were obtained by using only the content-based distance, i.e., with $\alpha = 1$. However, the very best results for QC-Means were found when using μ_2. Here, we significantly improved the baseline TS-26 in terms of F-measure ($\approx 10\%$) and Rand ($\approx 54\%$), while we lost $\approx 21\%$ in terms of Jaccard. Moreover, if we compare the best QC-Means with the best QFG we can notice that QC-Means lost $\approx 6\%$ for F-measure, $\approx 33\%$ for Jaccard but it gained $\approx 4\%$ in terms of Rand.

Then, we analyzed QC-Scan algorithm, again using both the distance functions μ_1 and μ_2. We used several combinations of the two density-based parameters, i.e., $minPts$ and eps, and we found the best results with $minPts = 2$ and $eps = 0.4$.

Tab. 4 highlights that QC-Scan provided globally better results than QC-Means for both μ_1 and μ_2. Still, for μ_1

Table 3: QC-MEANS: μ_1 vs. μ_2.

QC-MEANS μ_1				
		F-measure	Rand	Jaccard
α	$(1-\alpha)$			
1	0	0.71	0.73	0.26
0.5	0.5	0.68	0.70	0.14
0	1	0.68	0.70	0.13

QC-MEANS μ_2				
		F-measure	Rand	Jaccard
t	b			
0.5	4	**0.72**	**0.74**	**0.27**

the best results were obtained by using only content-based distance, i.e., with $\alpha = 1$. However, our proposed conditional function μ_2 revealed a significative improvement with respect to all measures.
Finally, it is worth noticing that QC-SCAN behaved exactly as QFG, except for the Jaccard where QC-SCAN lost $\approx 53\%$.

Table 4: QC-SCAN: μ_1 vs. μ_2.

QC-SCAN μ_1				
		F-measure	Rand	Jaccard
α	$(1-\alpha)$			
1	0	0.77	0.71	0.17
0.5	0.5	0.74	0.68	0.06
0	1	0.75	0.68	0.07

QC-SCAN μ_2				
		F-measure	Rand	Jaccard
t	b			
0.5	4	**0.77**	**0.71**	**0.19**

The third algorithm we considered is QC-WCC. Tab. 5 shows the results we found using this algorithm either with distance function μ_1 and μ_2 and by varying the pruning threshold η. In particular, concerning μ_1 we only considered the best convex combination when $\alpha = 0.5$.
The best results with μ_1 were obtained when $\eta = 0.2$, while even better results were found with μ_2 when $\eta = 0.3$. In this last case, the overall evaluation was significantly higher than the baseline TS-26 but also than the state-of-the-art approach QFG. Concerning TS-26, the best QC-WCC gained $\approx 20\%$, $\approx 56\%$, and $\approx 23\%$ in terms of F-measure, Rand, and Jaccard, respectively. Moreover, QC-WCC improved also the results of QFG, gaining $\approx 5\%$ for F-measure, $\approx 9\%$ for Rand, and $\approx 10\%$ for Jaccard.
QC-HTC is the last algorithm we introduced and represents one of the novel contribution of our work. The results we got using this approach with both distance functions μ_1 and μ_2 and by varying the pruning threshold η are showed in Tab. 6. As for QC-WCC, regarding μ_1 we only considered the best convex combination when $\alpha = 0.5$. Again, the best results with μ_1 were obtained when $\eta = 0.2$, while the global best results were found with μ_2 when $\eta = 0.3$. As the table shows, the overall results are very close to the ones obtained with QC-WCC. In particular, QC-HTC improved TS-26 by gaining $\approx 19\%$, $\approx 56\%$, and $\approx 21\%$ in terms of F-measure, Rand, and Jaccard, respectively. Therefore, QC-HTC provided better results than QFG and gained $\approx 4\%$ for F-measure, $\approx 9\%$ for Rand, and $\approx 8\%$ for Jaccard.
Globally, Tab. 7 gives an overview and compares the best results found with each examined approach.
Finally, Tab. 8 clearly points out the benefit of exploiting collaborative knowledge like Wikipedia. QC-HTC was able

Table 5: QC-WCC: μ_1 vs. μ_2 varying the threshold η.

QC-WCC μ_1 ($\alpha = 0.5$)			
η	F-measure	Rand	Jaccard
0.1	0.78	0.71	0.42
0.2	**0.81**	**0.78**	**0.43**
0.3	0.79	0.77	0.37
0.4	0.75	0.73	0.27
0.5	0.72	0.71	0.20
0.6	0.75	0.70	0.14
0.7	0.74	0.69	0.11
0.8	0.74	0.68	0.07
0.9	0.72	0.67	0.04

QC-WCC μ_2 ($t = 0.5, b = 4$)			
η	F-measure	Rand	Jaccard
0.1	0.67	0.45	0.33
0.2	0.78	0.71	0.42
0.3	**0.81**	**0.78**	**0.44**
0.4	0.81	0.78	0.41
0.5	0.80	0.77	0.37
0.6	0.78	0.75	0.32
0.7	0.75	0.73	0.23
0.8	0.71	0.70	0.15
0.9	0.69	0.68	0.08

Table 6: QC-HTC: μ_1 vs. μ_2 varying the threshold η.

QC-HTC μ_1 ($\alpha = 0.5$)			
η	F-measure	Rand	Jaccard
0.1	0.78	0.72	0.41
0.2	**0.80**	**0.78**	**0.41**
0.3	0.78	0.76	0.35
0.4	0.75	0.73	0.25
0.5	0.73	0.70	0.18
0.6	0.75	0.70	0.13
0.7	0.74	0.69	0.10
0.8	0.74	0.68	0.06
0.9	0.72	0.67	0.03

QC-HTC μ_2 ($t = 0.5, b = 4$)			
η	F-measure	Rand	Jaccard
0.1	0.68	0.56	0.32
0.2	0.78	0.73	0.41
0.3	**0.80**	**0.78**	**0.43**
0.4	0.80	0.77	0.38
0.5	0.78	0.76	0.34
0.6	0.77	0.74	0.30
0.7	0.74	0.72	0.21
0.8	0.71	0.70	0.14
0.9	0.68	0.67	0.07

Table 7: Best results obtained with each method.

	F-measure	Rand	Jaccard
TS-26 (*baseline*)	0.65	0.34	0.34
QFG $_{best}$ (*state of the art*)	0.77	0.71	0.40
QC-MEANS $_{best}$	0.72	0.74	0.27
QC-SCAN $_{best}$	0.77	0.71	0.19
QC-WCC $_{best}$	**0.81**	**0.78**	**0.44**
QC-HTC $_{best}$	0.80	0.78	0.43

to capture and group together two queries that are completely different from a content-based perspective, but that are strictly semantically-related, using the function μ_2. Indeed, "Cancun" is one of the region where the "Hurricane Wilma" impacted during the 2005 season (see the cross reference inside the corresponding Wikipedia article[4]). Moreover, "Los Cabos" and "Cancun" are both in Mexico despite they are far away from each other. It might be the case,

[4] http://en.wikipedia.org/wiki/Cancun

of course with no absolute certainty, the user was looking for the relative position of Los Cabos from Cancun just to understand if Los Cabos was struck by the hurricane as well.

Table 8: The impact of Wikipedia: μ_1 vs. μ_2

QC-HTC μ_1 ($\alpha = 1$)		QC-HTC μ_2 (0.5, 4)	
Query ID	Query String	Query ID	Query String
		63	los cabos
		64	cancun
65	hurricane wilma	65	hurricane wilma
68	hurricane wilma	68	hurricane wilma

7. CONCLUSION AND FUTURE WORK

We have discussed a technique for splitting into meaningful user sessions a very large, long-term log of queries submitted to a Web Search Engine (WSE). We have formally introduced the *Task-based Session Discovery Problem* as the problem of extracting from a stream of user's queries several subsequences of queries which are all related to the same search goal, i.e., a *Web-mediated task*. We have also proposed a clustering-based solution, leveraging distance measures based on query content and semantics, while query timestamps were used for a first pre-processing breaking phase. In particular, we exploited both Wikipedia and Wiktionary to infer the semantics of a query. Our novel graph-based heuristic, QC-HTC, which is a simplification of the weighted connected components QC-WCC, significantly outperforms other heuristics in terms of F-measure, Rand and Jaccard index. As future work, we plan to learn a model that describes how users compose together several tasks for enacting more complex *Web-mediated processes*. Finally, we aim at investigating how such processes or parts of them can be recommended, devising a novel recommender system that goes beyond the simple query suggestion of modern WSEs.

8. ACKNOWLEDGMENTS

We acknowledge the partial support of S-CUBE (EU-FP7-215483), ASSETS (CIP-ICT-PSP-250527), and VISITO Tuscany (POR-FESR-63748) projects. Also, we acknowledge the authors of [1] and the Yahoo! Research Lab of Barcelona for providing us the possibility of using their Query Flow Graph implementation for evaluation purposes. Authors also thank Franco Maria Nardini for the help provided in adapting the original Query Flow Graph implementation.

9. REFERENCES

[1] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM '08*, pages 609–618. ACM, 2008.

[2] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[3] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD '96*, pages 226–231. ACM, 1996.

[4] E. Gabrilovich and S. Markovitch. Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis. In *IJCAI*, pages 6–12, 2007.

[5] N. S. Glance. Community search assistant. In *IUI '01*, pages 91–96. ACM, 2001.

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

[7] D. He and D. J. Harper. Combining evidence for automatic web session identification. *IPM*, 38(5):727–742, 2002.

[8] B. J. Jansen and A. Spink. How are we searching the world wide web?: a comparison of nine search engine transaction logs. *IPM*, 42(1):248–263, 2006.

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

[10] A. Järvelin, A. Järvelin, and K. Järvelin. s-grams: Defining generalized n-grams for information retrieval. *IPM*, 43(4):1005 – 1019, 2007.

[11] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM '08*, pages 699–708. ACM, 2008.

[12] T. Lau and E. Horvitz. Patterns of search: Analyzing and modeling web query refinement. In *UM '99*, pages 119–128. Springer Wien, 1999.

[13] C. Leacock and M. Chodorow. *Combining Local Context and WordNet Similarity for Word Sense Identification*, chapter 11, pages 265–283. The MIT Press, May 1998.

[14] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *WWW '05*, pages 391–400. ACM, 2005.

[15] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC '86*, pages 24–26. ACM, 1986.

[16] K. W.-T. Leung, W. Ng, and D. L. Lee. Personalized concept-based clustering of search engine queries. *IEEE TKDE*, 20(11):1505–1518, 2008.

[17] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. UC Press, 1967.

[18] D. Milne and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In *AAAI '08*, pages 25–30, 2008.

[19] S. Orlando and F. Silvestri. Mining query logs. In *ECIR*, volume 5478 of *LNCS*, pages 814–817. Springer, 2009.

[20] H. C. Ozmutlu and F. Çavdur. Application of automatic topic identification on excite web search engine data logs. *IPM*, 41(5):1243–1262, 2005.

[21] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[22] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE TSMC*, 19(1):17–30, 1989.

[23] F. Radlinski and T. Joachims. Query chains: Learning to rank from implicit feedback. In *KDD '05*, pages 239–248. ACM, 2005

[24] V. V. Raghavan and H. Sever. On the reuse of past optimal queries. In *SIGIR '95*, pages 344–350. ACM, 1995

[25] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453, 1995.

[26] D. E. Rose and D. Levinson. Understanding user goals in web search. In *WWW '04*, pages 13–19. ACM, 2004.

[27] G. Salton and M. J. Mcgill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[28] X. Shen, B. Tan, and C. Zhai. Implicit user modeling for personalized search. In *CIKM '05*, pages 824–831. ACM, 2005.

[29] C. Silverstein, H. Marais, M. Henzinger, and M. Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

[30] F. Silvestri. Mining query logs: Turning search usage data into knowledge. *Foundations and Trends in Information Retrieval*, 1(1-2):1–174, 2010.

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

[32] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, May 2005.

[33] J. R. Wen, J. Y. Nie, and H. Zhang. Query clustering using user logs. *ACM TOIS*, 20(1):59–81, 2002.

[34] R. W. White, M. Bilenko, and S. Cucerzan. Leveraging popular destinations to enhance web search interaction. *ACM TWEB*, 2(3):1–30, 2008.