Modeling and Predicting the Task-by-Task Behavior of Search Engine Users

Claudio Lucchese
ISTI-CNR, Pisa, Italy
claudio.lucchese@isti.cnr.it

Salvatore Orlando
Università Ca' Foscari Venezia, Italy
orlando@unive.it

Raffaele Perego
ISTI-CNR, Pisa, Italy
raffaele.perego@isti.cnr.it

Fabrizio Silvestri
ISTI-CNR, Pisa, Italy
fabrizio.silvestri@isti.cnr.it

Gabriele Tolomei
Università Ca' Foscari Venezia, Italy
gabriele.tolomei@unive.it

ABSTRACT

Web search engines answer user needs on a *query-by-query* fashion, namely they retrieve the set of the most relevant results to each issued query, independently. However, users often submit queries to perform multiple, related *tasks*. In this paper, we first discuss a methodology to discover from query logs the latent tasks performed by users. Furthermore, we introduce the *Task Relation Graph* (TRG) as a representation of users' search behaviors on a *task-by-task* perspective. The task-by-task behavior is captured by weighting the edges of TRG with a *relatedness* score computed between pairs of tasks, as mined from the query log. We validate our approach on a concrete application, namely a *task recommender system*, which suggests related tasks to users on the basis of the task predictions derived from the TRG. Finally, we show that the task recommendations generated by our solution are beyond the reach of existing query suggestion schemes, and that our method recommends tasks that user will likely perform in the near future.

Categories and Subject Descriptors

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, Task discovery, Task recommendation

1. INTRODUCTION

People heavily trust in Web search engines to satisfy their daily information needs and running activities.

A key factor for the popularity of today's search engines is their user-friendly interfaces [6], which allow users to phrase their needs by means of *queries* as simple lists of keywords. Users exploit this simple query-based interface to retrieve Web information and resources with the aim of performing one or more specific Web-mediated *tasks* [16], e.g., "*find a recipe*", "*book a flight*", etc. Moreover, often users operate in parallel on multiple, related tasks, which in turn are part of more complex goals, usually referred to as *missions* [12].

To clarify this better, let us consider the user Alice issuing the queries "`new york hotel`" and "`waldorf astoria`", while Bob typing the query "`cheap new york hotels`" and "`holiday inn ny`". Both users are clearly trying to achieve the *same task*, that is *"reserving a hotel room in New York"*, but with different pairs of queries.

Then, suppose Alice issues other queries related to the task intent *"booking a ticket to the MoMA"*, and Bob submits queries associated with the task *"dining in a pizzeria in New York"*. Again, both Alice and Bob are possibly involved in the *same mission* of *"organizing a touristic travel to New York"*, but such mission is enacted by means of different tasks. So, even the same mission can be performed by distinct users in terms of different tasks and queries.

In this paper, we are interested in studying how people search the Web from a novel *task-by-task* perspective, in contrast with the common *query-by-query* fashion. Furthermore, we explore how such high-level view of users' search behaviors help realize a new application, namely a *task recommender system*.

Since users may express the same task intent with a variety of queries, we focus on the following challenges:

(i) mining search engine *query logs* to detect *tasks*, i.e., groups of queries issued by many users yet related to the same latent need;

(ii) extracting a *graph-based* representation of the *task-by-task* behavior of a large number of search engine users;

(iii) exploiting the graph to *predict* tasks performed by users; predictions are in turn used as *task recommendations*.

We call *Task Relation Graph* (TRG) the task-by-task search representation extracted from a query log. In a nutshell, TRG is a directed graph, whose nodes are the tasks performed by users, and the weights on the edges model the likelihood that, given a task performed by a user, she will also perform another task directly connected. We show how TRG may be exploited to generate novel *task recommendations*, which can help user achieve her long-term missions. To the best of our knowledge, our work is the first attempt towards modeling the task-by-task user search behavior, and providing task recommendations.

OAIR'13, May 22-24, 2013, Lisbon, Portugal.

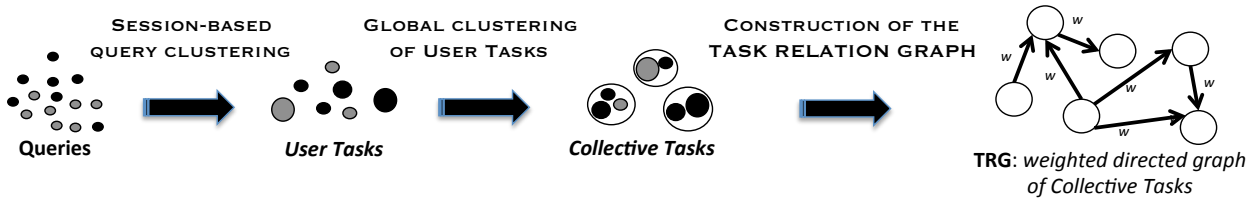

Figure 1: The proposed framework to represent the *task-by-task* search behaviors.

Of course, TRG may have many other possible applications, e.g., discovering periodic or frequently co-occurring tasks, understanding and anticipating user tasks for advertisements purposes, analyzing how users compose tasks to accomplish missions, etc.

TRG is built by first mapping queries to specific tasks. We propose a two-phase algorithm for this query mapping process. First, we extract sets of queries, named *user tasks*, where each set includes queries issued by a single user to achieve a given task within one of her interactions with the search engine. Then, we discover all the queries in the query log, related to a specific task, by grouping together all the similar user tasks, also performed by distinct users. We refer to this agglomeration of similar user tasks as a *collective task*. Finally, we learn the task relatedness, thus adding the weighted edges of TRG, by measuring the (temporal) task co-occurrence strength in the various user sessions used to build our model. Fig. 1 exemplifies the overall process of building TRG from the queries issued by two users identified by black and grey colors.

Furthermore, we firstly evaluate the precision of task predictions, which in turn lead to task recommendations. Secondly, we discuss some examples of possible task recommendations. This last evaluation reveals that the generated task recommendations can often be regarded as "*surprising*" and thus useful hints. To remark that task recommendation is different from query recommendation, we measure the ability of a well-known query suggestion scheme based on the *query-flow graph* (QFG) [7, 8], in recommending queries out of the current task boundary. As expected, QFG mostly recommends queries only related to the *same* task.

The rest of the paper is organized as follows. Section 2 describes the two-step task discovery process. In Section 3 we introduce TRG and we discuss several edge weighting schemes. More importantly, there we propose our novel task recommender system. An exhaustive experimental evaluation of this task recommender system is reported in Section 4. Section 5 discusses related work. Finally, Section 6 draws conclusion, and points out future research directions.

2. TASK DISCOVERY

This section describes how to discover tasks from historical usage data recorded in a search engine query log. We start by defining some notation and background concepts. Then, we introduce our task detection technique as composed of two main steps: *(i) discovering user tasks* and *(ii) discovering collective tasks*.

The rationale for this two-step task discovery strategy is that user tasks in each single user session can be discovered by exploiting the lexical and semantic content similarity of queries issued by individuals within their specific *search contexts*. On the other hand, the same approach would not be able to discover collective tasks if applied directly to users' queries. This is because queries that are issued by two users and that are similar, either from a lexical or a semantic perspective, might in fact refer to different latent needs.

2.1 Notation and Definitions

Let $\mathcal{QL}$ be a query log, which records the queries – along with userIDs, timestamps, clicked URLs, etc. – issued by a set of distinct users $\mathcal{U}$, with $|\mathcal{U}| = N$.

We denote by $\mathcal{S}_u = \langle q_1, q_2, \ldots, q_{|\mathcal{S}_u|} \rangle$ the chronologically ordered sequence of *all* the queries in $\mathcal{QL}$ issued by a user $u \in \mathcal{U}$. The sequence $\mathcal{S}_u$ of the queries submitted by u is the result of multiple, long-term interactions with the search engine. Therefore, we partition each sequence $\mathcal{S}_u$ into a set of *time-based sessions* according to a temporal query gap criterium. Basically, each time-based session contains the set of contiguous queries submitted by u such that each pair of query has been submitted within less than a maximum time-gap threshold Δ.

However, a time-based session s may include queries related to different needs due to *multitasking* [16, 3]. We further identify a partitioning of each time-based session into search related queries, and we refer to it as a *user task*.

Definition 2.1 (User Task [3]). *Given a time-based session $s \subseteq \mathcal{S}_u$ relative to user $u \in \mathcal{U}$, a* user task *θ, $\theta \subseteq s$, is the maximal sub-sequence of possibly non consecutive queries in s referring to the same latent need. The set of all* user tasks *in s gives a partitioning of s.*

According to the notation above, a user task θ is the set of queries that a user $u \in \mathcal{U}$ adopted to perform a given task. Also, the user may perform several tasks at the same time, thus generating a multitasking time-based session $s \in \mathcal{S}_u$. The sequence of tasks performed by a user $u \in \mathcal{U}$ is denoted by $\Theta_u = \bigcup_{s \subseteq \mathcal{S}_u} \bigcup_{\theta \subseteq s} \theta$, while the set of all the user tasks performed by all the N users is denoted by $\Theta = \bigcup_{u \in \mathcal{U}} \Theta_u$.

In general, we have to take into account the presence of multitasking within each time-based session $s \subseteq \mathcal{S}_u$, so we chronologically order the user tasks by looking at the timestamps of their first queries. In the following we use θ_u^i as a reference to the i-th user task performed by user u. This allows us to distinguish among the user tasks performed by each user, and to represent each Θ_u as an ordered set, namely a *sequence*, of user tasks, i.e., $\Theta_u = \langle \theta_u^1, \theta_u^2, \ldots, \theta_u^{|\Theta_u|} \rangle$.

Fig. 2 illustrates the process pursued to obtain the user tasks for a generic user, and also summarizes the notation we have just introduced.

The second step of our methodology concerns the agglomeration of user tasks to generate *collective tasks*, namely we aim at recognizing those *similar* user tasks across several users, and aggregating them into a single collective task.

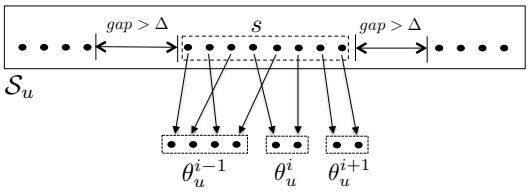

Figure 2: A generic time-based session composed of a set of *interleaved* user tasks θ_u^{i-1}, θ_u^i, and θ_u^{i+1}.

DEFINITION 2.2 (COLLECTIVE TASK). *Given the set Θ of all the sequences of user tasks, a* collective task *T is a maximal subset of Θ containing user tasks related to the same search intent. The set of all* collective tasks *in $\mathcal{QL}$, denoted by $\mathcal{T} = \{T_1, T_2, \ldots, T_{|\mathcal{T}|}\}$, is a partitioning of $\mathcal{QL}$.*

It turns out that this partitioning provides an implicit mapping from user to collective tasks, i.e., an *onto* function $f : \Theta \longmapsto \mathcal{T}$. Therefore, each original sequence of user tasks $\Theta_u = \langle \theta_u^1, \theta_u^2, \ldots, \theta_u^{|\Theta_u|} \rangle$ can be associated with a sequence of collective tasks $\widehat{\Theta}_u$, as follows:

$$\widehat{\Theta}_u = \langle f(\theta_u^1), f(\theta_u^2), \ldots, f(\theta_u^{|\Theta_u|}) \rangle.$$

Finally, we define $\overline{\Theta}_u$ as the *set* of the corresponding collective tasks, disregarding task ordering and duplicates:

$$\overline{\Theta}_u = \{f(\theta_u^i) \mid \theta_u^i \in \Theta_u\}.$$

2.2 Discovering User Tasks

The first step of our technique aims at identifying *user tasks*. To this end, we exploit the QC-HTC algorithm (Query Clustering based on Head-Tail Components) introduced in [3]. QC-HTC is a graph-based query clustering solution, which has proven to outperform other techniques addressing the problem of *session boundary detection* [7, 12].

In a nutshell, QC-HTC exploits a *content-based similarity* that combines two different kinds of query features, i.e., *lexical* and *semantic*. Moreover, since it is applied to individual time-based sessions, QC-HTC implicitly captures also the temporal feature of queries. The rationale for this choice is that each of those features affects the probability of any two queries being part of the same user task.

In more detail, QC-HTC operates as follows: first, it splits the long-term stream of queries $\mathcal{S}_u$ of a user $u \in \mathcal{U}$ into shorter time-based sessions. Then, for each time-based session $s \in \mathcal{S}_u$, QC-HTC builds a graph whose nodes are the queries in s, and whose edges link together any pair of consecutive queries. Each edge $(q_i,\ q_{i+1})$ is thus weighted by the probability of the two queries being task-related on the basis of their lexical and semantic similarity mentioned above.

The algorithm starts finding clusters of consecutive queries, i.e., *query chains*, whose edges are labeled by a similarity above a predefined threshold. Furthermore, QC-HTC augments the size of the clusters by hierarchically merging the chains. The merge is carried out by checking the similarity between *heads* (i.e., the first queries) and *tails* (i.e., the last queries) of the discovered chains, assuming that heads and tails might be a good representative for a user need. If the head (tail) of a chain is similar to the head (tail) of another chain, those clusters are eventually merged. At the end of this step, QC-HTC returns a set of user tasks θ_u^i for each time-based session $s \in \mathcal{S}_u$. This merging phase is crucial to handle the multitasking behavior of users.

2.3 Discovering Collective Tasks

The second step of our technique completes the overall task discovery process, and aims at capturing similar latent needs under the form of user tasks, i.e., different set of queries, submitted by several users.

We adopt a *bag-of-words* representation of each user task $\theta_u^i \in \Theta$. More formally, we denote with $\overline{q}$ the bag-of-words representation of a query $q \in \mathcal{QL}$, thus disregarding the order of the *word tokens* in q. Therefore, a user task is represented by $\overline{\theta_u^i} = \biguplus_{q \in \theta_u^i} \overline{q}$, where $\biguplus$ is the bag union operator. Of course, more complex solutions might be devised to enrich this simple representation by considering, for instance, the content of Web pages retrieved for the included queries and clicked by users.

Anyway, according to our chosen representation each query is viewed as a sentence, and a user task as a concatenation of many sentences, i.e., a *text document*. Thus, the problem of finding collective tasks can be reduced to the problem of clustering similar text documents, which is a well-investigated research topic [17, 18]. Note that, since we aggregate user tasks composed of several queries (indeed, their bag-of-words representations), this overcome the issue of the intrinsic ambiguity of single queries.

Eventually, clustering Θ results in a set of K collective tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$.

3. TASK RELATION GRAPH

To date, there exists several ways of representing the interactions between users and search engines on a *query-by-query* basis, like the *query-flow graph* (QFG) [7, 8]. We consider a different take on representing user-to-search engine interactions, and we define the *Task Relation Graph* (TRG). This is a new model capturing the *task-by-task*, as opposite to query-by-query, behavior of search engine users.

DEFINITION 3.1 (TASK RELATION GRAPH). *Let TRG $= (\mathcal{T}, E, w, \eta)$ be a weighted directed graph, called* Task Relation Graph*, such that:*
– $\mathcal{T} = \{T_1, T_2, \ldots, T_K\}$, the set of collective tasks, are the nodes *of the graph;*
– $E \subseteq \mathcal{T} \times \mathcal{T}$ includes the graph edges*, representing task relatedness;*
– $w : (\mathcal{T} \times \mathcal{T}) \longmapsto [0, 1]$ is a weighting function*;*
– $\eta > 0$ is a weight threshold*,*
where $E = \{(T_i, T_j) \in \mathcal{T} \times \mathcal{T} \mid w(T_i, T_j) \geq \eta\}$.

Intuitively, the TRG links any pair of collective tasks $T_i, T_j \in \mathcal{T}$ with an edge weighted by a function w, which measures the pairwise *task relatedness*. The threshold η is exploited to prune TRG, by dropping *weak edges*, i.e., edges connecting collective tasks with low relatedness.

This new graph-based modeling of the users' search behaviors can inspire innovative research directions and interesting applications. In fact, in this work we leverage the prediction capability of TRG for building a novel *task recommender system*.

3.1 Task Relatedness

In this section, we discuss how to compute the relatedness between pairs of collective tasks in our TRG. Intuitively, those pairs of tasks that highly frequently occur together in many user sessions are also likely to be related to each other.

Sequence Pairs. We first discuss how to exploit a classical algorithm for mining *frequent sequential patterns* [5] in order to compute task relatedness. To this end, we start from the input sequences $\widehat{\Theta}_u = \langle f(\theta_u^1), f(\theta_u^2), \ldots, f(\theta_u^{|\Theta_u|})\rangle$, one for each $u \in \mathcal{U}$. A *sequence* $\alpha = (T_i \rightarrow T_j)$, $T_i \neq T_j$, is supported by $\widehat{\Theta}_u$, denoted by $\alpha \sqsubseteq \widehat{\Theta}_u$, if there exists at least a pair of user tasks θ_u^h and θ_u^l, $h < l$, such that $f(\theta_u^h) = T_i$ and $f(\theta_u^l) = T_j$. The *support count* of α, denoted by $\sigma_1(\alpha)$, is the number of distinct input sequences $\widehat{\Theta}_u$ such that $\alpha \sqsubseteq \widehat{\Theta}_u$. Finally, let *seq-supp*$(\alpha) \in [0,1]$ the *support* of sequence α, defined as:

$$seq\text{-}supp(\alpha) = \frac{\sigma_1(\alpha)}{|\mathcal{U}|}.$$

This leads to the following task relatedness score, which is associated with the corresponding threshold $\eta = min\_supp$ (minimum support of the sequence):

$$w(T_i, T_j) = w_{seq\text{-}supp}(T_i, T_j) \;=\; seq\text{-}supp(T_i \rightarrow T_j). \quad (1)$$

Association Rules. In order to define task relatedness, we can also exploit classical *association rules* [4]. To this end, we consider the various sessions $\overline{\Theta}_u = \{f(\theta_u^i) \mid \theta_u^i \in \Theta_u\}$ as our input *transactions*, whose *items* are the collective tasks in $\mathcal{T}$. For our purposes, an *association rule* is defined as an implication of the form $T_i \Rightarrow T_j$, where $T_i, T_j \in \mathcal{T}$ and $T_i \neq T_j$, whose significance and interest are computed in terms of the classical *support* and *confidence* measures. Let $\sigma_2(X)$ be the *support count* of a set of collective tasks X, $X \subseteq \mathcal{T}$, defined as the number of distinct "transactions" $\overline{\Theta}_u$ such that $X \subseteq \overline{\Theta}_u$. Then, the support $supp(\cdot) \in [0,1]$ of an association rule is defined as follows:

$$supp(T_i \Rightarrow T_j) \;=\; \frac{\sigma_2(\{T_i, T_j\})}{|\mathcal{U}|}.$$

Instead, the confidence $conf(\cdot) \in [0,1]$ of the same rule is:

$$conf(T_i \Rightarrow T_j) \;=\; \frac{\sigma_2(\{T_i, T_j\})}{\sigma_2(\{T_i\})},$$

where the confidence of a rule can be interpreted as the conditional probability $P(T_j|T_i)$, namely the probability of finding collective task T_j in those transactions where users already performed T_i.

Finally, we derive the two following edge weighting scores:

$$w(T_i, T_j) = w_{ar\text{-}supp}(T_i, T_j) \;=\; supp(T_i \Rightarrow T_j), \quad (2)$$

$$w(T_i, T_j) = w_{ar\text{-}conf}(T_i, T_j) \;=\; conf(T_i \Rightarrow T_j), \quad (3)$$

which are associated with the following thresholds: $\eta = min\_supp$ (minimum support of the rule) for Eq. (2), and $\eta = min\_conf$ (minimum confidence of the rule) for Eq. (3).

3.2 Task Recommendation

The ultimate goal of this work is to prove the capability of TRG to predict, given a task performed by a user, which are the tasks most likely to be performed in the near future. We exploit this capability to build a novel *task recommender system* that considers as "related" exactly those tasks that TRG predicts to be executed next.

The main idea is to retrieve the set $\mathcal{R}_m(T_j)$ of collective tasks that are the most related to collective task T_j, which is currently performed by a user. In turn, from $\mathcal{R}_m(T_j)$ we select the most relevant tasks to be returned as recommendations to the current user.

Definition 3.2 (Task Recommendations). *Given a task relation graph* $TRG = (\mathcal{T}, E, w)$*, and a set of collective tasks* $\mathcal{T}^* \subseteq \mathcal{T}$*, compute the set* $\mathcal{R}_m(\mathcal{T}^*) \subseteq (\mathcal{T} \setminus \mathcal{T}^*)$*, such that:*
– $|\mathcal{R}_m(\mathcal{T}^*)| = m$*;*
– $\mathcal{R}_m(\mathcal{T}^*) \cap \mathcal{T}^* = \emptyset$*;*
– if $T_i \in \mathcal{T}^*$ and $T_j \in \mathcal{R}_m(\mathcal{T}^*)$ then $w(T_i, T_j) \geq w(T_h, T_k)$ for all $T_h \in \mathcal{T}^*$ and $T_k \in \mathcal{T} \setminus \mathcal{R}_m(\mathcal{T}^*)$.

In other words, the top-m recommended tasks $\mathcal{R}_m(\mathcal{T}^*)$ are those reachable from $\mathcal{T}^*$ on the TRG by following the edges with the maximum weighting scores.

Finally, we remark that we are not proposing a user interface for task recommendation, which involves other issues, such as labeling the set of tasks $\mathcal{R}_m(\mathcal{T}^*)$, and that are beyond the scope of this work.

4. EXPERIMENTS

In this section, we present the experimental setup used to generate our TRG model. Moreover, we assess the validity of TRG on a concrete application by evaluating the quality of the task recommendations it is able to provide.

4.1 Experimental Setup

All the experiments were conducted on the 2006 AOL query log $\mathcal{QL}$, which is a very large, long-term collection, consisting of about 20 million Web queries issued by approximately 657,000 users[1]. For our tests, we referred to a smaller dataset $\mathcal{QL}_{top-600} \subset \mathcal{QL}$, which contains the 600 user sessions with the highest total number of queries yet limited to the first week of logging. $\mathcal{QL}_{top-600}$ consists of 58,037 queries, meaning an average of about 97 queries per user over a week, and about 14 queries per user every day.

$\mathcal{QL}_{top-600}$ was first preprocessed via query *stopwords removal* and *stemming* [15], thereby it was randomly partitioned in two disjoint subsets $\mathcal{A}$ and $\mathcal{B}$ of user sessions. The first subset $\mathcal{A}$ had 48,257 queries issued by $\mathcal{U}_\mathcal{A} = 500$ users, and it has been used as the *training set* for building TRG. Instead, $\mathcal{B}$ had 9,780 queries issued by $\mathcal{U}_\mathcal{B} = 100$ users, and it has been used as the *test set* to evaluate the task recommendations. Both $\mathcal{A}$ and $\mathcal{B}$ are available to download[2, 3].

4.2 Building and Exploiting TRG

In order to build TRG from $\mathcal{A}$, we followed the two-steps methodology described in Section 2. After the first clustering step, we discovered a set of user tasks $\Theta_\mathcal{A}$, $|\Theta_\mathcal{A}| = 8,301$, also available to download[4].

Then we clustered the user tasks in $\Theta_\mathcal{A}$, thereby obtaining a new set of collective tasks $\mathcal{T}$. To this end, we evaluated two clustering approaches (i.e., *partitional* vs. *agglomerative*) included in the CLUTO[5] toolkit. In addition, both approaches needed three input parameters: *(i)* the similarity measure, *(ii)* the objective function, and *(iii)* the final number K of clusters. For the first choice, we used both *cosine similarity* and the *Pearson's correlation coefficient*. For the second one, we maximized the *intra-cluster similarity*. Lastly, in order to devise a reasonable value K of clusters,

[1] http://sifaka.cs.uiuc.edu/xshen/aol_querylog.html
[2] http://bit.ly/Tv6WkX
[3] http://bit.ly/WndztU
[4] http://bit.ly/UHRtfP
[5] http://glaros.dtc.umn.edu/gkhome/views/cluto

we exploited a common empirical heuristic, also known as the "*elbow method*". Roughly, we select a number $K = \bar{K}$ of clusters, such that for any $K > \bar{K}$ the slope of our objective function appears to increase lesser than what it does for $K < \bar{K}$. The motivation behind this method is that it suggests a number of clusters so that adding another cluster does not give much better modeling of the data.

Several solutions have been tested by combining all the above parameters. Eventually, we chose the partitional clustering in conjunction with the cosine similarity. This algorithm computes the final solution by performing a sequence of $K-1$ repeated bisections, starting from an initial cluster containing all the user tasks of the collection. This method has proven to outperform other tested solutions when compared to a ground-truth of collective tasks, which was manually built in advance by human annotators. Finally, we came up with a set of $K = 1,024$ collective tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_{1024}\}$.

On the basis of the edge weighting functions in Eq. (1), (2), and (3), respectively, we built three versions of TRG, then exploited to generate task recommendations for the users of the test set $\mathcal{B}$. Note that using different thresholds η, we can prune these graphs, thus obtaining different models. For the first two TRGs, we varied $\eta = min\_supp$ from 0.4% up to 100%. For the last TRG, we varied $\eta = min\_conf$ from 5% up to 100%.

In order to evaluate these TRGs on the test set $\mathcal{B}$, we mined from $\mathcal{B}$ a set of user tasks $\Theta_{\mathcal{B}}$[6], $|\Theta_{\mathcal{B}}| = 1,762$, as we did for the training set $\mathcal{A}$.

However, since no direct mapping exists from these user tasks to the collective tasks $\mathcal{T}$ identified from $\mathcal{A}$, namely the nodes of TRGs, we needed to "guess", for each user task $\theta_u \in \Theta_{\mathcal{B}}$, the most suitable collective task $T' \in \mathcal{T}$ to include θ_u, where $u \in \mathcal{U}_{\mathcal{B}}$. To this end, we defined the mapping function $g : \Theta_{\mathcal{B}} \longmapsto \mathcal{T}$, as follows:

$$g(\theta_u) = T' = \operatorname*{argmax}_{T \in \mathcal{T}} \{sim(\theta_u, T)\}, \tag{4}$$

where $sim(\theta_u, T)$ was the *cosine similarity* between the vector space representations of the user task θ_u and the collective task $T \in \mathcal{T}$, respectively.

Finally, task recommendations were provided according to the scheme described in Def. 3.2. In particular, given a user task θ_u, and the associated collective task $T' = g(\theta_u)$, we looked at TRG and we suggested the top-m neighboring tasks $\mathcal{R}_m(T_c)$, i.e., the m collective tasks, directly reachable from T', whose corresponding edges have the highest weighting values.

4.3 Precision of Task Recommendations

Evaluating the *quality* of provided suggestions is a well-known issue in traditional recommender systems. Indeed, the primary aim of recommender systems is to enhance user's *satisfaction*. Unfortunately, no standard measure of user's satisfaction exists, and most approaches evaluate the accuracy of provided suggestions using offline assessments. Therefore, we start measuring the *precision* of suggestions generated by our task recommender system. This in turn refers to the precision of TRG in *predicting*, given a currently performing task, which tasks a user might likely execute next.

In order to measure the precision of provided recommendations, each sequence of user tasks $\Theta_u \in \Theta_{\mathcal{B}}$, $u \in \mathcal{U}_{\mathcal{B}}$, was first subdivided into two subsequences, the former Θ_{u1} containing 1/3 of user tasks, and the latter Θ_{u2} the remaining 2/3. By using Eq. (4), we got the two sets of collective tasks:

$$\overline{\Theta}_{u1} = \{g(\theta_u) \mid \theta_u \in \Theta_{u1}\}, \qquad \overline{\Theta}_{u2} = \{g(\theta_u) \mid \theta_u \in \Theta_{u2}\}.$$

The first set of collective tasks $\overline{\Theta}_{u1}$ was used to generate task recommendations, which in turn were evaluated on the second one, i.e., $\overline{\Theta}_{u2}$.

According to Def. 3.2, the set of generated suggestions for the collective tasks in $\overline{\Theta}_{u1}$ was $\mathcal{R}_m(\overline{\Theta}_{u1})$, where no collective task of $\overline{\Theta}_{u1}$ was suggested, i.e., $\mathcal{R}_m(\overline{\Theta}_{u1}) \subseteq (\mathcal{T} \setminus \overline{\Theta}_{u1})$. Thus, for each user session Θ_u, $u \in \mathcal{U}_{\mathcal{B}}$, we computed the *precision* $p(\Theta_u)$ as follows:

$$p(\Theta_u) = \frac{|\mathcal{R}_m(\overline{\Theta}_{u1}) \cap \overline{\Theta}_{u2}|}{|\mathcal{R}_m(\overline{\Theta}_{u1})|}. \tag{5}$$

The precision of *all* the user sessions in the test set $\mathcal{B}$ was then computed as follows:

$$p(\Theta_{\mathcal{B}}) = \frac{\sum_{\Theta_u \in \Theta_{\mathcal{B}}} p(\Theta_u)}{|\Theta_{\mathcal{B}}|}. \tag{6}$$

Furthermore, we had to compare the precision of the task recommendations obtained from the three different TRGs, built as specified in Section 4.2. In addition, each TRG may have many instances, since it can be gradually pruned by increasing its associated threshold η. For each TRG, we argued that there existed a value of η, which corresponded to a particular TRG instance maximizing the precision of task recommendations.

In order to contrast a TRG instance with another, we checked if they were comparable, by implicitly looking at their *graph connectivity*. To this end, we measured the proportion of collective tasks $T \in \overline{\Theta}_{u1}$ that were able to provide *at least one* task to recommend. More formally, we defined the *coverage* $c(\Theta_u)$, as follows:

$$c(\Theta_u) = \frac{|\{T \in \overline{\Theta}_{u1} \mid \mathcal{R}_m(T) > 0\}|}{|\overline{\Theta}_{u1}|}. \tag{7}$$

We derived the value of coverage for *all* the user sessions in $\mathcal{B}$, as follows:

$$c(\Theta_{\mathcal{B}}) = \frac{\sum_{\Theta_u \in \Theta_{\mathcal{B}}} c(\Theta_u)}{|\Theta_{\mathcal{B}}|}. \tag{8}$$

Obviously, the coverage of a TRG is inversely proportional to the threshold η. When η increases, edges in the graph are pruned and, consequently, the coverage decreases.

Fig. 3 shows the precision of the top-k recommendations, provided by leveraging the various TRG models for both *medium-* and *large-sized* sessions, as a function of the corresponding coverage. Specifically, the plots report the outcomes obtained by different model instances, each built by using a given parameter setting, e.g., a specific threshold η (minimum support or confidence).

The medium-sized test sessions contain between 9 and 19 user tasks, and the large-sized more than 19. Also, they are significant representatives of the whole test set $\mathcal{B}$, since they account for about 79% of the total number of test sessions.

We compared the above results with a *baseline* approach, which, for a given set of tasks, generated the set of recommended tasks by *uniformly* choosing them among the set of all possible tasks. In all these cases, the uniform approach provided the lowest values of precision, as expected.

[6] http://bit.ly/YgCDDg

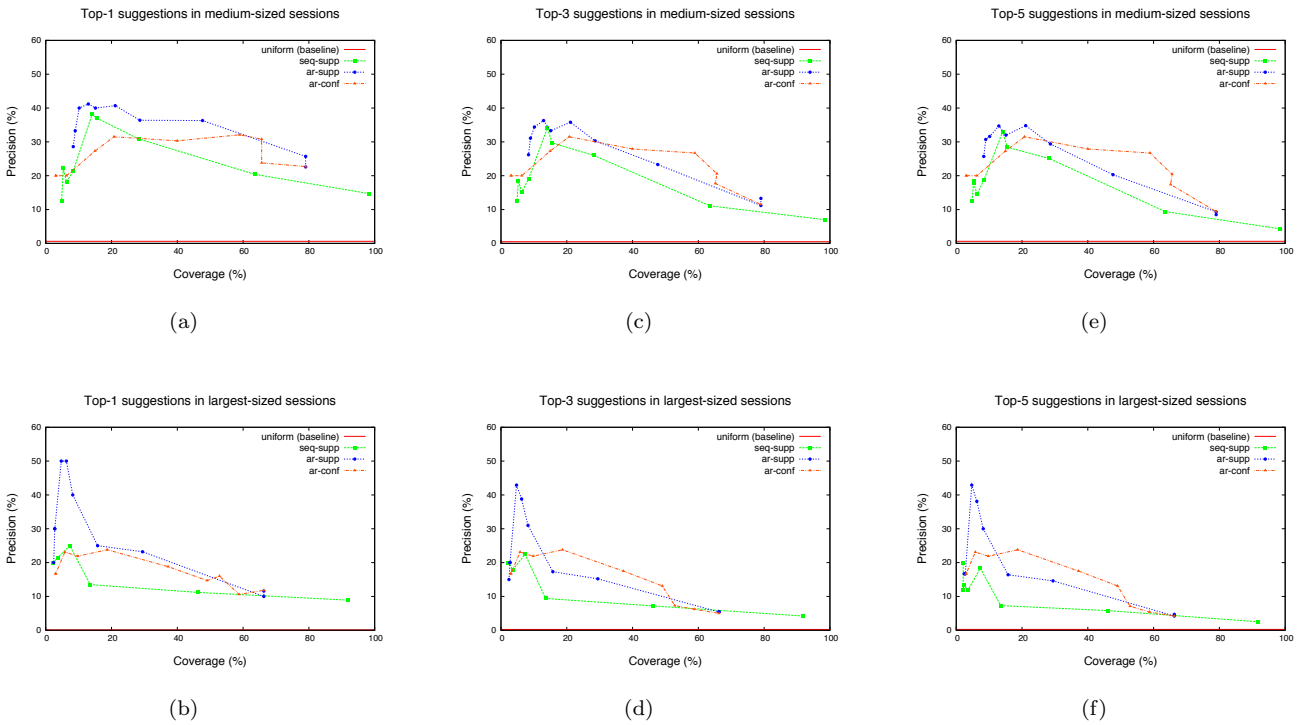

Figure 3: Precision vs. Coverage for top-1, top-3, and top-5 task recommendations.

From all the plots depicted in Fig. 3 we can draw general conclusion, which are valid independently of the number $k = \{1, 3, 5\}$ of provided recommendations.

First of all, the very best value of precision is obtained when the weights on the edges of TRG were computed with the association rule score of Eq. (2), namely when using the support of learned association rules. In particular, we obtain high values of precision both for medium- and large-sized sessions. For instance, if we limit to top-1 suggestions, we observe the precision evaluates to 41.2% and 50.0% (see Fig. 3(a) and 3(b), respectively).

The second observation concerns the different behaviors of medium- and large-sized sessions when coverage increases. In general, by increasing coverage we were able to produce more recommendations thus necessarily reducing the precision. However, this behavior appears significantly smoother in medium- than in larger-sized sessions. Indeed, precision in medium-sized sessions tends to remain "stable" or to gracefully degrade even for larger values of coverage. Again considering top-1 recommendations, we note that precision evaluates to 36.3% when coverage is approximately 47.7%. Thereby, precision is still close to its maximum (i.e., 41.2%), which instead is obtained with a smaller value of coverage (i.e., about 13.0%). In contrast, coverage seems to affect the precision of recommendations for large-sized sessions.

This depends on the fact that the first third of the sequence used to generate suggestions is *shorter* in medium- than in large-sized sessions. Intuitively, by increasing the coverage the likelihood that the first third of large-sized sessions already contain tasks that are provided as suggestions increases as well. The precision drops because these duplicate tasks are filtered out from the final set of suggestions.

4.4 Examples of Task Recommendations

It is worth noting that our recommendation mechanism lies at a higher level of abstraction than traditional query suggestion schemes. Indeed, we also show some examples derived from our test set, which clarify how and why our task recommender solution is substantially different from existing query suggestion.

Performed Tasks/Queries	Recommended Tasks
Home Furnitures	**Home Gardening**
	cottage garden
beach house	cottage garden roses
...	
...	
beach house vanity	
Kitchen Decor	**Kitchen Supplies**
dining room	stoves
...	
...	

Table 1: Recommended and performed tasks.

First, the ability of our technique in predicting tasks that users are going to execute in the near future is remarked in Tab. 1. The left column of this table shows tasks that a user planned to perform by issuing a set of queries, which appear below the *task label*[7]. In the right column is shown a suggested task that actually the same user executes during her next searches by issuing the underlined queries.

[7]Task labels have been manually generated in advance.

On the other hand, Tab. 2 shows task recommendations that are *not* performed by any user in our test set. Anyway, it is evident that these recommendations are very relevant to the task the user is performing because they might provide hints on "what task to do next". To clarify this better, we report some of the most representative queries contained in each collective task just below its label.

Performed Tasks/Queries	Recommended Tasks
Child Entertainment	**Child Games**
	baby shower games
fables	*baby horse*
...	**Child Dressing**
...	*baby gap*
baby fables	**Child Health**
	baby emotional disease
University	**University Sports**
university	*university sports*
...	*university basketball*
...	**University Information**
duke university	*university tuition*

Table 2: Recommended and *surprising* tasks.

4.5 Query vs. Task Recommendation

Up to now, we have evaluated our task recommender system in terms of its precision in predicting future tasks. In this section, we discuss the reasons why our proposed solution differs from existing query suggestion mechanisms, by comparing it to a state-of-the-art query suggestion approach, based on the *query-flow graph* (QFG) [7, 8]. To this end, we used the same experimental setup as described above in Section 4.3. First, we learned the QFG model on the same training set $\mathcal{A}$, illustrated in Section 4.1. Then, we used the test set $\mathcal{B}$ for evaluating QFG recommendations. Each sequence of user tasks $\Theta_u \in \Theta_{\mathcal{B}}$, $u \in \mathcal{U}_{\mathcal{B}}$, was divided into two disjoint portions: the first Θ_{u1} containing 1/3 of user tasks was used to ask QFG for suitable query suggestions, whereas the second Θ_{u2} included the remaining 2/3, and it is used for evaluation purposes. However, since the QFG was designed for recommending queries, we had to "roll back" each Θ_u into its original sequence of queries $\mathcal{S}_u$. Then, for each user task $\theta_u^i \in \Theta_{u1}$ and for each query $q \in \theta_u^i$, we retrieved the set of top-1, top-3, and top-5 recommended queries returned by the QFG. Thereby, we checked if the suggested queries q' were actually part of Θ_{u2}. If this was the case, we figured out which user task $\theta_u^j \in \Theta_{u2}$ were including q'. Eventually, the two queries q and q' belonged to two distinct collective tasks if $g(\theta_u^i) \neq g(\theta_u^j)$. In the end, we aimed at measuring the proportion of this recommended queries q' belonging to a collective task that was different from the one where the original query q lived.

From our results, the vast majority of the suggestions generated by QFG resides in the same collective task of the queries that originated them. In particular, about 99.7% of the top-1 QFG suggestions lives in the same task of their original queries. Of course, this percentage decreases as the number of generated suggestions increase, because the chance of finding recommended queries that belong to multiple tasks becomes higher. Thus, about 83.8% of top-3 and 72.3% of top-5 recommended queries reside in the same task of their original queries. Indeed, this is not an unexpected result, since QFG recommendations for a certain query q contain those queries that appear immediately next to q with high probability. From a *query-by-query* perspective, this is certainly reasonable, since users are assumed to keep interested in the same task from one query to the next. However, this approach becomes unsatisfactory when we go up to a *task-by-task* point of view, where users are instead supposed to perform multiple tasks for reaching complex missions.

Nevertheless, since a very small amount of suggested queries led users towards *new* tasks, we also evaluated the "*diversity*" of provided recommendations, by measuring the number of tasks covered by all the suggested queries. About 15.1% (23.8%) of top-3 (top-5) suggestions led to 2 distinct tasks, respectively, only 3.5% of top-5 suggestions cover 3 tasks, and a very small fraction of them (i.e., about 0.4%) refer to 4 or 5 tasks.

Globally, top-1 suggestions refer to only 1 task on average whereas for top-3 and top-5 suggestions the average diversity is about 1.2 and 1.3 tasks, respectively.

For the sake of completeness, we also measured the precision of QFG recommendations according to Eq.(6). To this end, we computed the proportion of collective tasks behind the suggested queries that were correctly mapped to the collective tasks in Θ_{u2}. Actually, no significant difference between top-1, top-3, and top-5 suggestions was evidenced, and precision we obtained was very low (i.e., about 2.1%).

Altogether, these results highlighted that a state-of-the-art query suggestion technique like QFG-based is suitable for driving Web search engine users inside the *same* task they are currently performing. Our experiments revealed that QFG-based query suggestion is not designed and cannot be adapted to supply "high-level" *task recommendations*, as instead allowed by our proposed method.

Concretely, we claim that modern Web search engines should integrate existing query suggestion tools with novel task suggestion techniques, thus giving users the chance of either staying in their current task or jumping to other tasks for achieving complex missions more quickly.

5. RELATED WORK

Search engine query logs record precious information about the *search activities* of users [2]. Extracting valuable knowledge from such data may help improve both the *effectiveness* and the *efficiency* of search engines [1].

Many works concerning query log mining aim to understand how people search the Web on a query-by-query perspective. Broder [9] conducted the first study attempting to classify Web queries with respect to their search intents, and introduced a widely used *query taxonomy*. Boldi *et al.* [7] proposed the *query-flow graph* (QFG) as a model for representing data collected in search engine query logs. Intuitively, in the QFG a directed edge from query q to query q' means that the two queries are likely to refer to the same search need. Any path over the QFG may be seen as a searching behavior, whose likelihood is given by the strength of the edges along the path. The authors proved the usefulness of the model in two concrete applications, i.e., *session boundary detection* and *query recommendation*.

Jones and Klinkner [12] presented the first *high-level*, task-by-task analysis of user search behavior, by introducing the concept of *hierarchical search*. They argued that within a user's query stream it is possible to recognize complex *search missions*, which are in turn composed of simpler *search goals*,

i.e., our *user task*, while a search mission is a set of topically-related information needs, resulting in one or more goals. The authors investigated how to learn a binary classifier, aimed to precisely detect whether two queries belong to the same goal or not. Mei *et al.* [14] presented a framework to analyze sequences of user search activities. This framework adopts the same hierarchical model introduced by Jones and Klinkner [12], and is able to capture sequences of user behavior at multiple levels of granularity. Donato *et al.* [10] developed *SearchPad*, a novel *Yahoo!* application. It is still built on the top of the hierarchical structure of search behavior proposed in [12], and is able to automatically identify search missions "on-the-fly", as the user interacts with the search engine. Guo *et al.* [11] introduced the concept of *intent-aware* query similarity, namely a novel approach for computing query pair similarity, which takes into account the potential search intents behind user queries, and then measures query similarity under different intents using intent-aware representations. The authors showed the usefulness of their approach by applying it to query recommendation, thereby suggesting diverse queries in a structured way to search users. More recently, Kotov *et al.* [13] proposed a classification framework for modeling and analyzing user search behavior spanning over multiple search sessions. The authors focused on two problems: *(i)* given a user query, identify all related queries from previous sessions that the user has issued, and *(ii)* given a multi-query task for a user, predict whether the user will return to *this* current task in the future. Unlike our approach, they did not tackle the problem of predicting *new* tasks, but they only aimed at guessing whether or not a user will ever perform again the same task she is currently executing. In fact, to the best of our knowledge, our solution is the first proposed technique to discover, model, predict, and finally recommend tasks to search engine users, thus providing an overall framework for the task-by-task Web search.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented the first proposal for a novel *task recommender system*, which generates *task recommendations* on the basis of the *task-by-task* (instead of the classical query-by-query) behavior of several search engine users.

This ambitious goal was achieved by firstly discovering the tasks from the stream of queries stored in a query log using of a two-step clustering technique. Furthermore, we introduced the *Task Relation Graph* (TRG) as a model for capturing any relationship between detected tasks. In TRG, each task is a node, and any edge connecting two tasks is weighted by the likelihood of those tasks being related to each other. We proposed three task relatedness scores learned from the query log, which lead to three distinct TRG models.

We conducted an intensive experimental phase on a real-world query log in order to measure the ability of each model in predicting and covering the actual tasks that users were going to perform next. Task predictions were in turn used to generate task recommendations whose precision and coverage are significantly higher than a baseline approach, which instead consisted in suggesting tasks uniformly. In addition, even if task recommendations were not part of future user requests, we have shown that they could be anyway "*surprising*". Then, we performed a set of additional experiments to show that our task recommender addressed a *new* problem, which existing query suggestion is not able to tackle.

We plan to investigate several research directions in the future. First, advanced task representations, as opposite to simple *bag-of-queries*, could allow to suggest more interesting "items" to search engine users at search-time. In addition, TRG might be exploited for many sound applications other than task recommendation, such as task-based advertisement, mission discovery, etc. Finally, automatic task labeling might help build a "*taxonomy of Web tasks*".

7. ACKNOWLEDGMENTS

The authors would like to acknowledge the partial support of projects MIDAS (FP7 EU Grant Agreement no. 318786), InGeoCloudS (CIP-PSP EU Grant Agreement no. 297300), and MIUR PRIN ARS TechnoMedia. Finally, we acknowledge the authors of [7] and the Yahoo! Research Labs in Barcelona, Spain, for providing us their *query-flow graph* implementation, and Dr. Franco Maria Nardini for adapting this implementation to our needs.

8. REFERENCES

[1] F. Silvestri, R. Baraglia, C. Lucchese, S. Orlando, and R. Perego. (Query) history teaches everything, including the future. In *LA-WEB 2008*, pp. 12–22. IEEE CS 2008.

[2] F. Silvestri. *Mining Query Logs: Turning Search Usage Data into Knowledge*. Foundations and Trends in Information Retrieval, 4(1-2):1–174, 2010.

[3] C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Identifying task-based sessions in search engine query logs. In *WSDM '11*, pp. 277–286. ACM 2011.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94*, pp. 487–499, 1994.

[5] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE '95*, pp. 3–14, 1995.

[6] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley 1999.

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

[8] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *WSCD '09*, pp. 56–63. ACM 2009.

[9] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36:3–10, September 2002.

[10] D. Donato, F. Bonchi, T. Chi, and Y. Maarek. Do you want to take notes?: identifying research missions in yahoo! search pad. In *WWW '10*, pp. 321–330. ACM 2010.

[11] J. Guo, X. Cheng, G. Xu, and X. Zhu. Intent-aware query similarity. In *CIKM '11*, pp. 259–268. ACM 2011.

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

[13] A. Kotov, P. N. Bennett, R. W. White, S. T. Dumais, and J. Teevan. Modeling and analysis of cross-session search tasks. In *SIGIR '11*, pp. 5–14. ACM 2011.

[14] Q. Mei, K. Klinkner, R. Kumar, and A. Tomkins. An analysis framework for search sequences. In *CIKM '09*, pp. 1991–1994. ACM 2009.

[15] M. F. Porter. *An Algorithm for Suffix Stripping*, pages 313–316. Morgan Kaufmann Publishers 1997.

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

[17] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *CIKM '02*, pp. 515–524. ACM 2002.

[18] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55(3):311–331, June 2004.