Behavior-driven Clustering of Queries into Topics

Luca Maria Aiello
Università di Torino
Torino, Italy
aiello@di.unito.it

Debora Donato
Yahoo! Labs
Sunnyvale, CA, USA
debora@yahoo-inc.com

Umut Ozertem
Yahoo! Labs
Sunnyvale, CA, USA
umut@yahoo-inc.com

Filippo Menczer
Indiana University
Bloomington, IN, USA
fil@indiana.edu

ABSTRACT

Categorization of web-search queries in semantically coherent topics is a crucial task to understand the interest trends of search engine users and, therefore, to provide more intelligent personalization services. Query clustering usually relies on lexical and clickthrough data, while the information originating from the user actions in submitting their queries is currently neglected. In particular, the *intent* that drives users to submit their requests is an important element for meaningful aggregation of queries. We propose a new intent-centric notion of topical query clusters and we define a query clustering technique that differs from existing algorithms in both methodology and nature of the resulting clusters. Our method extracts topics from the query log by merging *missions*, i.e., activity fragments that express a coherent user intent, on the basis of their topical affinity. Our approach works in a bottom-up way, without any a-priori knowledge of topical categorization, and produces good quality topics compared to state-of-the-art clustering techniques. It can also summarize topically-coherent missions that occur far away from each other, thus enabling a more compact user profiling on a topical basis. Furthermore, such a topical user profiling discriminates the stream of activity of a particular user from the activity of others, with a potential to predict future user search activity.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Clustering; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Algorithms, Human factors

Keywords

Topic, Mission, User intent, User profiling, Query log

1. INTRODUCTION

Methods and algorithms for improving web search have been extensively studied in the last two decades. In the vast majority of the cases, such methodologies are query-centric, i.e., they exploit only the query itself to understand a user's intent and to provide relevant results. Only recently user search activity has been studied by looking at the entire set of actions she performs in order to satisfy a need. Such a line of research is based on the observation that users searching the web often perform a sequence, or chain, of queries with a similar information need [27]. Empirical studies have indeed shown that most of the needs are actually too complex to be satisfied by just one query [13]. Hence users are inclined to organize their search activity in so-called *missions*.

By way of example, let's consider the activity of a user who wants to purchase a vacuum cleaner. She will probably organize her queries in a number of sequential steps. Initially she starts acquiring information about available brands and models. In a second phase she will look for reviews and comparisons between different models with similar features. Finally, she will search for sellers who offer the chosen model at an advantageous price or with an extended guarantee. Each single step is a different granular task and the totality of tasks are meant to fulfill the "vacuum-cleaner purchase" need. In a similar scenario, an unsatisfied user who wants to return a vacuum cleaner is going to submit queries such as "returning shark navigator" or "dyson customer service." The two needs in this example concern particular aspects related to the same general topic "vacuum cleaner." We use the term *topic* in its acceptation of *mental object* or *cognitive content*, i.e., *the sum of what can be perceived, discovered or learned* about any real or abstract entity. In such a sense, topics naturally emerge from user search activity, since queries are issued to discover or learn facets (model, name, characteristics, pros and cons, price, seller's location, return policy) of a cognitive content (vacuum cleaner).

In this paper we present a methodology to extract topics from query logs. Our objective is inherently different from previous attempts to classify queries according to a predefined set of categories, because our definition of topic encompasses and outstrips the definition of category. Categories like shopping, sport, news, and finance can indeed be seen as the perspective or focus of a search mission in the context of a particular topic.

We propose a mission-based clustering technique to aggregate missions that are topically-related. As a first step we train a classifier to predict if two different missions have a similar topical connotation. The learned function takes as input two sets of queries and computes the probability that they are topically-related. Such a function is used as a similarity metric by an agglomerative algorithm to merge missions into large topical clusters.

The resulting topics would be useful in a number of different scenarios where great advantage can be derived by performing query categorization in a more natural, intent-driven fashion, without any constraint imposed by artificial categories. As an application of the methodology, we show how to build user profiles over topics and we use such profiles to predict user behavior.

The paper is organized as follows. In Section 2, we will give an overview of related work. Section 3 introduces terminology and presents the problem we aim to solve. Section 4 presents a semi-supervised classifier able to detect if two different missions belong to the same topic. A greedy agglomerative algorithm for topic extraction is illustrated in Section 5. The experimental framework and the results are respectively described in Sections 6 and 7. Section 8 presents an application of our methodology for user profiling.

2. RELATED WORK

Query chains, missions, goals. Previous work on query log mining has introduced general and widely used terms to define different structural features of the query log and behavioral aspects of the search engine users. Typically, sequences of query reformulations aimed to achieve the same atomic search need are referred as *query chains* [27]. More relaxed notions of coherence within the search session are represented by the concept of *logical session* [2] and *search mission* [20], namely a set of queries that express a complex search need, possibly articulated in smaller goals. Our work uses missions as atomic elements to build behavioral-driven query clusters.

Query clustering. Even if closely related to the document *topic extraction* task [29] or to multi-document summarization [4], where items from a textual corpus are summarized in one or more, possibly hierarchical [17] categories, our work gives a contribution to the *query clustering* area. Currently, query clustering is one of the hot topics in query log mining [6].

Grouping together queries with strong semantic relations is a task that is intrinsically harder than classic topic extraction or web document clustering [37], because of the short textual information contained into queries. Many approaches to query clustering rely on the computation of some notion of similarity between query pairs. When dealing with *query classification*, where the semantic categories are defined a-priori, it may be sufficient to compute the similarity based only on textual features to obtain good classification results [7]. Even classification based only on the so-called *clickthrough* data such as the information inside the result pages associated to the queries, can lead to good results [18]. However, if predefined concept categorizations or taxonomies are not available, lexical and content-based information taken separately are not sufficient to obtain good clusters. In this regard, an attempt to cluster queries from the Encarta user logs [33] showed that query-to-query similarity metrics that linearly combine textual features with click-through data can be used much more profitably in query clustering than single-attribute similarities. Similarly, hierarchical agglomeration of queries based on the similarity of their search result snippets (that mix words from the query and text from the page results) has also been profitably used [12].

Approaches focused only on the activity of single users, instead of the whole query log, are also interesting; in fact, it is clear that detecting the topics that can well shape the interests of the user is useful for personalization and recommendation. For instance, Song et al. [31] proposed a topic model able to extract most relevant themes from the user activity through probabilistic Latent Semantic Indexing [19]; once identified, such themes are used as a topical summarization of the user search history.

More recently, content-agnostic approaches based only on the relations between queries and clicked URLs have been explored. The idea is that similarity graphs between queries obtained by proper projections on the multiple dimensions of the query log accurately model semantic relations between queries [3, 9]. Many approaches of this kind represent the query-URL relation as a bipartite graph [6] and pick the densely connected components, like bicliques [36], as representative sets of similar queries or pages. *Query graphs* where queries are connected by some lexical or semantic relation are also commonly used. In order to overcome the problem of query ambiguity in topic detection, query graphs that introduce a lightweight query contextualization using the pairs $(sessionId, query)$ as nodes, instead of single queries, have also been proposed [32].

Graph clustering. In this work we use a graph-based model as baseline for our technique. This model relies on a network *community detection* algorithm to partition a graph of queries into topics. Community detection is an important branch of complex network analysis that received much attention in the recent past. One of the most known result in this field is the definition of the *modularity* as a global metric of goodness of clusters [26]. Even if finding maximum modularity partitions is an NP-complete problem [10], numerous algorithms based on suboptimal modularity maximization has been proposed, until the intrinsic resolution limit of modularity in cluster detection was discovered [16]. Lately, different fitness measures to define cluster quality have been introduced (e.g., [24]) and many recently proposed algorithms use local metrics to detect communities [15]. To our purpose, we will use a graph clustering algorithm that has been proved more effective than some of the main state of the art competitors [23].

3. DEFINITIONS

Query log. In a search engine context, search activity is typically recorded in a query log $\mathscr{L}$ defined as a set of tuples $\tau = \langle q, u, t, V, C \rangle$ containing the submitted query $q \in \mathscr{Q}$, an anonymized user identifier $u \in \mathscr{U}$, the time t when the action took place, the set of documents V returned by the search engine, and the set of clicked documents $C \subseteq V$ [8].

Missions. According to the original definition of Jones at al. [21], a *search mission is a related set of information needs, resulting in one or more goals*. In the example presented in the Introduction, "Purchasing a vacuum cleaner" is a mission that represents an intent that a user wants to satisfy. The three steps ("looking for models", "models comparison," and "sellers comparison") are the three sub-tasks (or goals) contained in the mission. All the queries in a missions have a strong topical coherence, which means that all

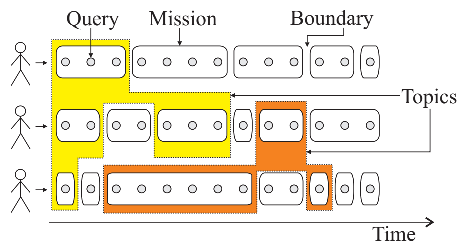

Figure 1: The activity of three search engine users partitioned at different levels. Every stream of user queries can be articulated in different missions that are aimed to satisfy a particular user intent. At a coarser level, topics can include missions from different users and also portions of activity originated by the same user at different times.

of them are issued with a main common objective. It has been observed [13] that search activities that take place in complex domains like "travel" or "health" often require several queries before complex user intents are fully satisfied.

Topics. Missions are characterized by a main objective and one or more sub-tasks related to the objective itself. For example, a mission devoted to organize a trip, has the travel itself as main objective and a number of functional sub-tasks (like booking the flight, reserving the hotel, finding a guided tour). Travel missions generated by different users are all characterized by the same main objective regardless the destination, the temporal order in which the sub-tasks are issued or even the recreational activities booked. In such a sense all the missions devoted to organize a travel can be seen as part of a same topic or *cognitive content*.

Missions within the same cognitive content are meant to fulfill one or more intents related to such a content. Here on, we use the term *topic* to define a *cognitive content that includes the sum of what can be perceived, discovered or learned about any real or abstract entity*. From an operational point of view, this means that a topic can be seen as the aggregation of all missions with the same cognitive content generated over time across different users. A sketch depicting the relation between queries, missions and topics can be found in Figure 1.

Our goal. Our primary objective is to define a methodology to aggregate different missions within the same cognitive content. This technique should be effective at different scales. In a individual perspective, it should be able to aggregate different but related missions of the same user, while in a wider context it can be used as a tool to cluster together all the missions that are topically coherent, across users. The method must not rely on predefined categorizations or taxonomies of user intent or topics.

4. MISSION SIMILARITY CLASSIFIER

According to the definitions given in Section 3, the concepts of mission and topic are strictly related to each other. In fact, sequences of queries that express coherently a single and well defined user intent must have a high degree of topical coherence. This strong connection allows us to use missions as fundamental building blocks for topics: distinct missions can be merged together if their semantic connotation is very similar. In the following, we first summarize the state of the art technique that we use to detect missions from the query log; then we describe a method that is able to effectively combine together pairs of topically-related missions.

4.1 Detection of search missions

To partition the user activity into missions we use the machine learning approach proposed by Donato et al. [13]. This method is able to detect the boundaries of a mission by analyzing the live stream of actions performed by the user on the search engine. This approach relies on a module based on a gradient boosted decision tree classifier [34], called the *mission detector*, that works at the level of query pairs. Given a set of features extracted from a pair of consecutive query log tuples τ_1, τ_2 generated by the same user, the mission detector indicates whether τ_2 is coherent with τ_1, from a topical perspective. When two queries are found to be incoherent, then a mission boundary is placed, so that the query log $\mathscr{L}$ is partitioned into missions containing one or more tuples.

The features used for the classifications come from three different domains: the *textual features*, that include different flavors of lexical similarity between the two queries, the *session features*, that measure several aspects of the click activity of the user in the time between the two queries and in the overall session, and the *time-related features* that take into account the inter-event time distance for some representative user actions. Using all of these feature together, the mission detector is able to reach a 95% accuracy in detecting boundaries on real user datastreams [13].

It has to be noted that missions identified by this method are semantically much narrower than topics, because queries in the same missions are not only constrained to be submitted by the same user, but they are also consecutive in time. Indeed, while a mission in theory can be fragmented in time, the mission detector by definition can only aggregate consecutive queries and, in practice, generates short-lived missions. Thus, the topical coherence constraints imposed on missions are much stronger than those that we require to be applied to topics.

4.2 Merging missions

Given the state of the art of mission boundary detection, it is possible to segment the user activity of every query log into missions. Furthermore, the strong topical coherence of queries inside the same mission can be exploited to generalize the approach used for mission boundary to a topic boundary detection. The idea is to use a new classifier, the *topic detector*, trained in semi-supervised fashion based on the data generated by the missions detector, to decide whether two query sets belong to the same topic. Its scheme is sketched in Figure 2.

Specifically, positive examples are automatically built by splitting missions in two consecutive query sequences and considering such two sequences as separate (sub)missions belonging to the same topic. Conversely, negative examples are formed by sets of queries belonging to consecutive missions of the same user, since we know they are topically unrelated because they are separated by the boundary placed by the mission detector. The topic detector is implemented with a Stochastic Gradient Boosted Decision Tree (GBDT) [35]. GBDT outputs the probability that the given sample is from

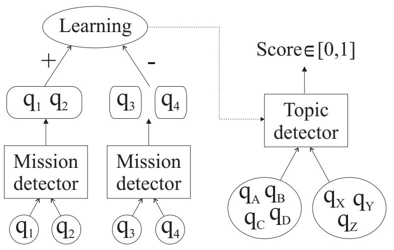

Figure 2: Training and application of the topic detector. Positive and negative examples for the learning phase are respectively missions and pairs of consecutive missions that are detected by the mission detector. Once trained, the topic detector can take in input any pair of query sets and compute a confidence score that can be interpreted as a topical similarity between the two sets.

the positive class; applied to our case, this is the probability that the two missions in input belong to the same topic. We can interpret this probability as a similarity score in $[0, 1]$ measuring the topical relatedness of the two sets and, consequently, the classifier can be seen as a *topical similarity function* $\mathscr{S}$.

The features given in input to the classifier are aggregated values over features computed from all the query pairs across two missions. Namely, given a (positive or negative) pair of missions m_1, m_2, all query pairs $q_1, q_2 | q_1 \in m_1 \wedge q_2 \in m_2$ are taken into account. Then, all the values of each feature are aggregated over all the pairs yielding four scores representing the average, standard deviation, minimum, and maximum values for that feature. For each query pair, features from three different categories are extracted[1]:

- **Lexical features**. Very often, similarity between the text of different queries denotes a strong semantic relation (e.g., "paris cheap travel" and "travelling to paris"). For this reason, we train the classifier using several lexical features such as length of common prefix and suffix, size of the intersection, edit distance, several similarity measures computed at word and character 3-grams level, and many others.

- **Behavioral features**. The behavior of users during the search activity gives much implicit information on the semantic relatedness of queries. For instance, if a user submits two queries in close succession, it is likely that the two queries are very related to each other, based on the assumption that the user activity is *bursty* [5] and events happening in the same burst are meant to accomplish the same task. However, since user behavior is very heterogeneous, it is necessary to aggregate behavioral information from several user sessions. We compute the average values of the behavioral features over a year of query log for each query pair q_1, q_2 such that q_2 has been observed at least once right after q_1 in the log.[2] The average time and the average number of clicks between two queries are examples of behavioral features.

- **Search result features**. Intuitively, the page result sets returned for a pair of topically-related queries will be topically related as well, to some extent. Thus, we consider a bunch of result-related features such as the intersection between result sets and the similarity between the vectors of the K most frequent words from a given content dictionary [1] appearing in the N top results.

[1]We do not report the complete list of features since they are commonly used in session analysis [20, 8].

[2]Behavioral features are defined only for successive query pairs observed at least 2 times. Else, a default value is used.

Algorithm 1 Iterative topic extraction

```
Require: Initial set of seed topics $\mathscr{T}_0$; similarity threshold $\theta \in (0,1)$; termination threshold $\alpha \in (0,1)$; topic similarity function $\mathscr{S} : \mathscr{T} \times \mathscr{T} \to [0,1]$
 1: $\mathscr{T}_{i+1} = \mathscr{T}_0$
 2: repeat
 3:     $\mathscr{T}_i = \mathscr{T}_{i+1} - T_0$ ; $\mathscr{T}_{i+1} = T_0$
 4:     for $T_1 \in \mathscr{T}_i$ do
 5:         $T_x = T_2 \in \mathscr{T}_{i+1} | \mathscr{S}(T_1, T_2) \geq \theta \wedge \mathscr{S}(T_1, T_2) \geq \mathscr{S}(T_1, T), \forall T \in \mathscr{T}_{i+1}$
 6:         if a valid $T_x$ has been found then
 7:             $\mathscr{T}_{i+1} = \mathscr{T}_{i+1} - T_x + (T_x \cup T_1)$
 8:         else
 9:             $\mathscr{T}_{i+1} = \mathscr{T}_{i+1} + T_1$
10:         end if
11:     end for
12: until $\frac{|\mathscr{T}_i+1|}{|\mathscr{T}_i|} \leq \alpha$
```

We trained our topic detector using a balanced sample of 500K mission pairs extracted from random user sessions over the 2010 query log of the Yahoo! search engine. Results of 10-fold cross validation give an AUC value of 0.95, thus confirming that the topic detector is able to accurately discriminate between sets of queries that come from the same topic and those that do not.

One may think that the mission detector could have been used as-is to extract topics by means of detecting pairs of topically coherent queries among all the possible query pairs and iteratively clustering them. However, this approach has two main flaws. First, the computational effort to classify every possible query pair in $\mathscr{Q}$ is prohibitive given the dimension of real search query logs. Second, the mission detector is trained given query pairs that are adjacent realizations of the same user activity stream, while in topic extraction we are interested in comparing queries of different users, possibly in different times. On the other hand, classifying pairs of query sets allows us to leverage the mission data as an already available source. In any case, even if it were possible to classify every query pair, the clustering step would still require similarity computation for query sets.

5. GREEDY AGGLOMERATIVE TOPIC EXTRACTION

The topic similarity function $\mathscr{S}$ can be used iteratively to extract topics from the mission data. Consider a set $\mathscr{T}_0$ of *seed topics* where each topic contains only the set of queries submitted during a single user mission. This set is a very strict partition of the query corpus, where each partition is not only a coherent topic but it is the expression of an intent of a single user.

Our topic extraction algorithm is in the spirit of classic hierarchical agglomerative clustering [34]. As shown in the

Algorithm 1 pseudo-code, at any iteration i, the two sets $\mathscr{T}_i$ and $\mathscr{T}_{i+1}$ are considered. Initially, $\mathscr{T}_{i+1} = \{T_0\}$ and $\mathscr{T}_i = \{\mathscr{T}_0 - T_0\}$.

Each topic $T \in \mathscr{T}_i$ is compared with all topics $T_x \in \mathscr{T}_{i+1}$ through the function $\mathscr{S}(T, T_x)$; if topic similarity is below a certain threshold θ for all pairs, then T is moved from $\mathscr{T}_i$ to $\mathscr{T}_{i+1}$. Otherwise, the pair (T, T_x) with the highest score is greedily selected, T is removed from $\mathscr{T}_i$ and its elements are added to T_x, creating a broader topic. The algorithm is iterated as long as the relative decrease in the number of topics is large, or until $\frac{|\mathscr{T}_i+1|}{|\mathscr{T}_i|} > \alpha \in (0, 1)$.

The complexity of a single iteration, in terms of number of computations of the $\mathscr{S}$ function, is $O(|\mathscr{T}_i|^2)$ and $\Omega(|\mathscr{T}_i|)$, since $\frac{|\mathscr{T}_i|^2 - |\mathscr{T}_i|}{2}$ computations of the topic similarity are needed if no topics are merged, while just $|\mathscr{T}_i|$ comparisons take place if every topic merges into a single supertopic. The number of iterations needed depends on the stop condition α, but it is always bounded by $O(log(|\mathscr{T}_0|))$, thus leading to an overall algorithm complexity of $O(|\mathscr{T}_0|^2 \cdot log(|\mathscr{T}_0|))$.

Even if the number of iterations required is considerably smaller than the theoretical upper bound, in practice the quadratic complexity of computing $\mathscr{S}$ for every topic pair is still too costly for large query logs. However, the efficiency can be improved through a heuristic approach based on the observation that a very small portion of all possible topic pairs are actually merged in each iteration. To reduce the number of comparisons, the topic set $\mathscr{T}_i$ is partitioned into several smaller sets that are given in input to independent instances of Algorithm 1, so that the $\mathscr{S}$ function is applied only between elements inside the same partition and not across partitions. In addition to reducing the number of topic pairs considered, this approach enables a parallel implementation of the clustering algorithm, thus dramatically decreasing the actual computation time, even though the theoretical complexity remains the same. For brevity, in the following we will refer to our Greedy Agglomerative Topic Extraction algorithm using the acronym *GATE*.

The choice of a good *partitioning criterion* is crucial for the outcome of GATE. To maximize the number of topics merged at each iteration, partitions should contain topics that are more likely to be combined than randomly selected topics. This can be done by putting in the same partition topics that share some of the features given in input to the classifier used to compute topic similarity; for instance, topics can be partitioned on the most common character-level 3-gram that appears in their query sets, given that topics with some lexical similarity are more likely to be merged than random topics. The partition criterion can also possibly change at each iteration.

Aggregation on the basis of user identity is one of the most relevant to our study. If the first iteration of the algorithm is run keeping the missions of different users in different partitions, then the resulting agglomeration produces a minimal group of topically-coherent mission sets, called *supermissions*. These supermissions allow to define more compact profile of user activity on a topical basis.

6. EXPERIMENTAL SETUP

We extracted the total activity of $40K$ users from 3 months of the anonymized Yahoo! query log. Statistics for the data set are reported in Table 1. The queries in the log were sequentially grouped into 3,005,724 missions using the mission detector described in Section 4.1. The number of missions per user and number of unique queries per mission are broadly distributed with average values as in Table 1. The first iteration of GATE is performed with a user-based aggregation criterion, therefore the topics produced in the first iteration are supermissions. The average number of supermissions per user is 42, against the 57 missions per user, meaning that GATE can be used to compress the user description by nearly 30% on average.

Table 1: Dataset statistics

Unique queries (uq)	2,198,815
Missions (m)	3,005,724
Unique missions (um)	1,606,733
Avg uq per mission	1.72
Avg m per user	75
Avg um per user	57

In the rest of this section, we present OSLOM [23], a representative network-based clustering algorithm used for comparison with GATE. We then introduce the metrics used to compare such methodologies. The two methodologies are profoundly different, therefore a fair comparison is difficult. Nevertheless we show that our approach has a number of advantages when compared with OSLOM.

6.1 Baseline: Topic extraction through network clustering

We compare GATE with a content-agnostic query clustering algorithm based on query graphs. As mentioned in Section 2, graph-based clustering considers the queries as nodes and model relation between them with edges. Depending on the relation used, the query graph can assume different topologies and semantics [2]. The choice of comparing our method with a baseline from a different paradigm is motivated by the significant body of recent work and promising results using graph based approaches for query clustering. A thorough comparison between agglomerative clustering and graph based approaches is still missing in the context of query log mining.

The input to our baseline is a query graph based on the *click co-occurrence* relation, also known as *URL cover graph* [2]. Such graph models the topical relatedness of queries from the perspective of the common results to which they lead users: two queries are connected if users click on the same results. Edges are weighted depending on how many distinct clicked URLs are shared by the queries. In principle, several different query-query relations can be mapped onto a single query graph; for instance, it is quite common to mix lexical similarity and click co-occurrence (or other content-related similarity measures). In the present evaluation we consider a simple URL cover graph as input to our baseline.

To detect topically coherent clusters from the URL cover graph, we use a network *community detection* algorithm. Intuitively, the basic idea of community detection is to spot groups of nodes that have many connections with each other and few to the rest of the network, thus forming a dense cluster. Accordingly, a cluster on the URL cover graph would include a set of queries that have many more clicked URLs in common compared to all the other queries in the corpus.

Among the many community detection algorithms in the literature [15], we adopt the recently proposed OSLOM (`www.oslom.org`). This choice is motivated by its good performance over other state of the art algorithms on both synthetic benchmarks [22] and real network datasets [23]. Furthermore, unlike the vast majority of other community detection techniques, OSLOM automatically detects overlapping communities and hierarchies of clusters. This allows a more direct comparison with our greedy algorithm, which also outputs overlapping topics through a hierarchical agglomeration process.

OSLOM performs clustering based on the optimization of a local fitness function that measures the statistical significance of the detected cluster compared to a global randomized null model known as configuration model [25]. Clusters are detected by selecting several random seed nodes in the graph and finding the locally optimal clusters that include the seed nodes. Similar clusters found over different realizations of random seeds are then merged together, thus originating a minimal set of clusters that may overlap. The procedure is iterated over higher hierarchical levels by collapsing clusters into nodes. Iterations stop when no higher level clusters are found. For further details about the algorithm we refer to the original paper [23].

6.2 Metrics

In the following, we present the criteria used for a quantitative and qualitative comparison between the two methodologies. Although prior work has relied on human editors, manual evaluation for a huge corpus of topics is unfeasible, therefore here we focus on automatic evaluation metrics.

Clustering measures

The first group of metrics are meant to give a quantitative comparison of the two methods. We define three measures:

query set coverage: fraction of queries that the methodology is considering in the clustering phase;

singleton ratio: fraction of queries that remains isolated in singleton at the end of the iterative procedures;

aggregation ability: percent of topics that are aggregated in two consecutive iterations or in two consecutive hierarchical levels.

Cluster purity

The quality of a topic depends on its *purity*, or semantic coherence. When no ground truth about the correct composition of a topic is available, an automatic way to assess purity is to consider the results for all pairs of queries in a topic. We compare the result sets of two queries to assess how *related* they are, and average the relatedness over all pairs.

Perhaps the simplest way to establish the relatedness of two queries is to compare the result sets returned for each query and use the intersection to derive a measure of similarity [28, 14]. However, a more fine-grained criterion that uses the information inside the clickthrough result provides a more accurate evaluation basis. We construct a bag-of-words vector for each query, consisting of the *concepts* in the documents returned for this query. Concepts are selected using a predefined dictionary [1]. For a given query q and a concept dictionary D, Algorithm 2 computes the vector of concepts with their scores. The parameters are set to $k = 10$ and $K = 20$. The score of a term $t \in D$ depends on the number of documents in which the term t appears,

Algorithm 2 Algorithm to compute the concept vector

```
Require: Concept dictionary $D$, query $q$, parameters $k$, $K$
 1: Retrieve set $R$ of top-$k$ results for $q$
 2: $T$ = Terms from $D$ contained in $R$
 3: Eliminate from $T$ terms in $q$
 4: for term $t \in T$ do
 5:     $d(t)$ = number of results in which $t$ appears
 6:     $r(t)$ = sum of ranks of the results in which $t$ appears
 7:     $R(t) = \frac{((k+1)\ d(t)) - r(t)}{d(t)\ k}$
 8:     $S(t) = \frac{d(t)\ R(t)}{k}$
 9: end for
10: Return the $K$ terms with highest score $S(t)$
```

Table 2: Examples of the concept vector

Query	Terms in aboutness vector (ordered)
iphone	iphone, store, 4, camera, phone, apps, inc, facetime, mode, recording, software, 3gs, resolution, battery, shop, network, ios, 3g, broadband, ipod
iphone 4 reviews	phone, iphone, store, 4, camera, apple, reviews, apps, review, 3g, recording, model, inc, screen, resolution, ipad, coverage, network, photo, 3gs
toyota prius	hybrid, car, prius, toyota, mpg, photo, price, sales, specs, vehicle, yaris, cars, review, reviews, msrp, specification, milage, model, economy, research
toyota yaris	yaris, hatchback, car, price, hybrid, toyota, models, spec, mpg, liftback, dealer, model, vehicles, review, photo, reviews, city, prius, transmission,vehicle

and on the sum of ranks of those documents. Table 2 shows examples of two pairs of topical related queries, and their top concept terms ordered by descending score. Given the concept vectors of the queries that it contains, the concept vector of a topic is obtained by marginalizing all concept vectors and keeping the top concept terms.

Let us define the purity of a topic by considering all pairs of terms in the corresponding concept vector and measure how much they are related to each other on average. One could achieve this using the well-known pointwise mutual information (PMI) given by:

$$PMI(t_1, t_2) = \frac{f(t_1, t_2)}{f(t_1)f(t_2)}$$

where $f(t_1, t_2)$ is the number of queries that have both terms t_1 and t_2 in their concept vectors, and $f(t_1)$, $f(t_2)$ are the numbers of queries that have t_1 and t_2 in their concept vectors, respectively. One weakness of PMI is that it may become very unstable for a pair of rare terms. For example, if both $f(t_1)$ and $f(t_2)$ are small, a few coincidental co-occurrences may lead to a superficially high PMI value. To take this into account, we use the log-likelihood ratio, which is the expected value of the PMI:

$$\begin{aligned} LLR(t_1, t_2) &= p(t_1, t_2)PMI(t_1, t_2) + p(t_1, \overline{t_2})PMI(t_1, \overline{t_2}) \\ &+ p(\overline{t_1}, t_2)PMI(\overline{t_1}, t_2) + p(\overline{t_1}, \overline{t_1})PMI(\overline{t_1}, \overline{t_1}) \end{aligned}$$

where $\overline{t}$ denotes the set of all terms except t. LLR fixes the unstability problem of PMI. Note that when the marginal query frequencies $f(t_1)$ and $f(t_2)$ are small, the other terms in the LLR equation will start to dominate. Averaging LLR across all the pairs of terms in the topic concept vector, we obtain the log-likelihood ratio for the topic.

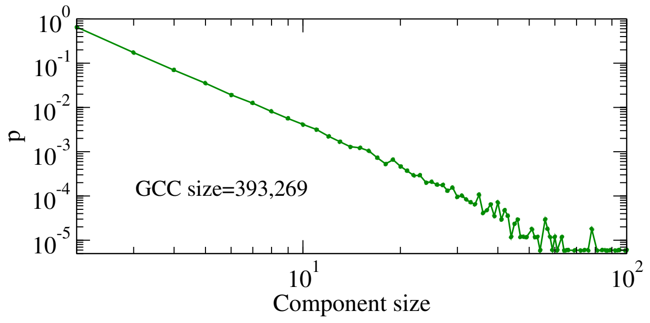

Figure 3: Distribution of connected components size in the query graph, except the giant connected component (GCC).

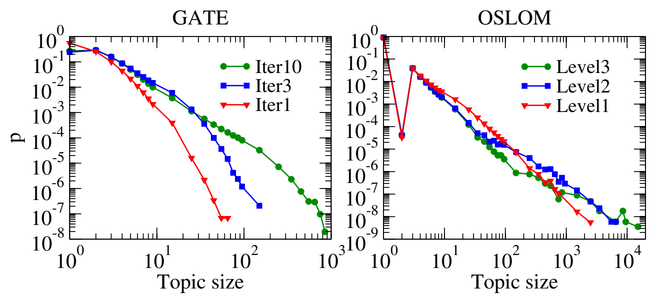

Figure 4: Distribution of topic size for GATE and

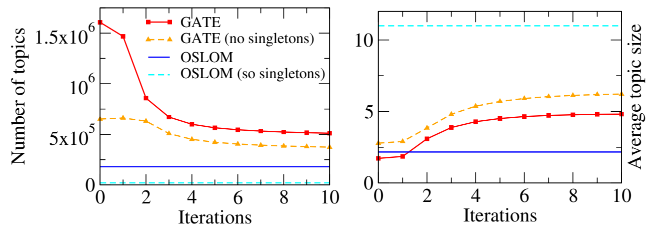

Figure 5: Number of topics and average topic size vs. GATE iterations. The values from the lowest hierarchical level of OSLOM are reported as a reference.

URL coverage

The topic LLR only focuses on the purity of the topic, which by itself is not too meaningful for the evaluation. A trivial solution of considering each query as a topic would yield very high topic LLR values. Similarly, one can easily cluster only synonymous queries, such as the query "facebook" and its misspellings, again leading to a high purity measure. Similar to singletons, such topics consisting of queries with almost identical results are not useful. To generate meaningful abstractions of the query space, it is desirable to aggregate related queries with different result sets. We therefore need a measure of *coverage* to complement purity, as suggested by previous work on clustering [30].

We measure the coverage by the number of unique URLs in the result sets for the queries in the topic. Given the 2010 query logs of the Yahoo! search engine, we extract all distinct URLs clicked by users for each query. To remove the tail of the clickthrough distribution, URLs that received less than 0.01 clickthrough rate are discarded. We define the coverage of a topic as the aggregate number of distinct URLs across the queries in the topic. Note that the trivial solutions of singleton topics or synonymous queries have very low coverage. Overall, our goal is to extract topics with both high purity and high coverage.

7. EXPERIMENTAL RESULTS

In this section we compare the two clustering algorithms according to the metrics presented in Section 6.2.

The GATE algorithm has a full query coverage on the dataset, because, by definition, every query can be found in at least one mission. Conversely, when dealing with graph-based clustering algorithms, the sparsity of the graph can lead to the emergence of isolated components that directly affect query coverage. In fact, since the vast majority of queries share very few clicked results, it turns out that the URL-cover graph is composed by a galaxy of tiny disconnected components. These little islands cannot be merged with other components due to lack of connections and, since they are mainly singletons, they do not represent any meaningful cluster just by themselves. The size distribution of the components in the URL-cover graph generated by our dataset is shown in Figure 3. We note that less than 400K queries are in the Giant Connected Component (GCC), thus leading to query set coverage below 0.2.

Figure 4 shows the distribution of the topic size with increasing number of iterations in GATE and in the three different hierarchical levels detected by OSLOM. The first thing to observe is that the singleton ratio in GATE decreases from 0.55 in the first iteration to 0.27 in the last one, while in OSLOM it remains stable around 0.88. Second, while in GATE the number of both medium-size (10-100) and big size (>100) topics grows with the iterations, in OSLOM the medium-size topics that are detected in the first hierarchical level tend to be merged in very large comprehensive and heterogeneous topics, with up to some tens of thousands queries. Note that the low frequency of size-2 topics in OSLOM is probably due to the tendency of the algorithm to merge dyads in larger clusters to optimize the partition fitness function.

Lastly, the aggregation ability of the two algorithms is shown in Figure 5. In GATE the number of topics decays quickly in the early iterations and then stabilizes, until the stop condition is reached. The final number of topics is around 500K, against the 180K found by OSLOM; recall that OSLOM only deals with the fewer queries in the GCC. The number of topics in the different hierarchical levels in OSLOM varies very slightly. Furthermore, the average size of OSLOM topics is more than double the size of the topics generated by GATE, mainly because of the presence of very large topics that skew the mean value.

Results on topic purity are shown in Figure 6. To check how the purity of the topic decreases with its size we computed the correlation between the topic size and LLR by averaging the LLR values of the topics with the same size:

$$\langle LLR_s \rangle = \frac{1}{|T \in \mathscr{T} : size(T) = s|} \sum_{T \in \mathscr{T} : size(T) = s} LLR(T).$$

As expected, the larger the topic, the more heterogeneous

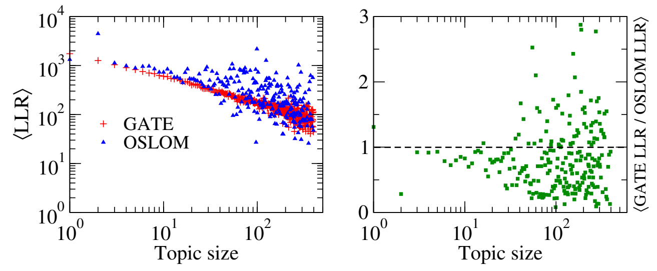

Figure 6: Left: Correlations between topics size and

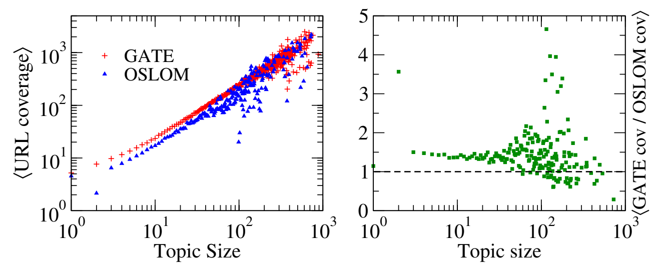

Figure 7: Left: Correlation between topic size and URL coverage. Right: Ratio between the two quantities.

the queries included, so we observe a negative correlation. OSLOM's topics have on average higher LLR than topics from GATE; this emerges clearly when computing the ratio between the average scores obtained by the two approaches for each topic size. This is mainly due to the fact that OSLOM generates many topics that include just concept *reformulations*, i.e., queries that are different from the lexical perspective but express exactly the same concept (e.g. "duran duran", "duan duran", "duran", etc.). As we remarked, such queries surely belong to the same cluster, but still they do not express a complex cognitive content. For this reason, it is necessary to complement the purity measure with the URL coverage. In Figure 7, the same analysis made for the URL coverage shows that the greedy algorithm has much higher coverage than OSLOM. If considering purity and coverage metrics separately is useful to learn the peculiarities of the two approaches, a joint measure that captures the tradeoff between the two scores is useful to compare the overall quality of the query clusters. Such unified measure can be obtained by multiplying the purity by the coverage for all the size classes of the topics and then comparing the resulting curves. Results in Figure 8 show that GATE outperforms OSLOM for all the medium-small topic sizes, that represent the great majority of topics. For clusters containing about 100 queries the two approaches are comparable, and only for some bigger, more rare topics OSLOM achieves the best performance.

To summarize, if we consider all the quality measures together, we can conclude that GATE leads to a better topical query clusters because it is able to process all the queries in the corpus, and most of the topics it generates have a better tradeoff between purity and coverage compared to OSLOM.

As a final remark we note that a fair comparison of the computational time needed by the two techniques is hard,

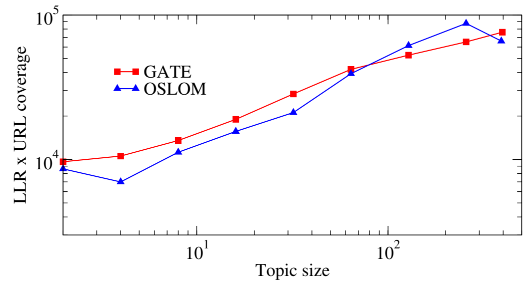

Figure 8: Trade-off between purity and URL coverage

first because OSLOM's theoretical complexity is difficult to estimate [23] and last because GATE can run in parallel while the current OSLOM implementation does not enable parallelism. However, we underline that parallelization is a strong advantage of GATE, because when dealing with real-life data sizes, it is hard to regard non-parallel algorithms as practical solutions for clustering web search queries.

8. USER PROFILING

A practical way to use the topics extracted from the query log is to profile users on a topical basis: each user can be described by the set of topics that match her queries. Since OSLOM has a small query coverage and allows only for an exact match between the user queries and the queries inside the clusters, it is not practical to employ it in profiling. Therefore, here we focus on user profiling based on our approach.

To build the profile of a user, we apply the topical similarity function $\mathscr{S}$ between the user missions and every topic that contain at least one query from that mission, then selecting the best match. Formally, let us define the topic that best matches mission m as:

$$T_m = \arg\max_{T \in \mathscr{T}} \mathscr{S}(m, T).$$

Given the best match scores, let us define the *topical profile* of a user u as a weighted vector over the topics matching her missions:

$$\mathscr{P}_u = \left\{ \left(T_m, \frac{\mathscr{S}(m, T_m)}{\sum_{m' \in \mathscr{M}_u} \mathscr{S}(m', T_{m'})} \right), \forall m \in \mathscr{M}_u \right\},$$

where $\mathscr{M}_u$ is the set of missions of user u. For a more compact user representation, supermissions can be used instead.

The topical profile can be used not only to detect the topics relevant to the user, but also to predict her future search goals. To check such a prediction potential, we perform an experiment to examine whether a user profile matches her future missions better than random missions from other users.

The match between a mission and a profile is performed by computing the $\mathscr{S}$ function between the mission and every topic in the profile, and scaling the resulting scores by the weights of the corresponding topics in the profile. This yields a vector of match scores over the profile topics. The match vector can be generalized to sequences of missions by averaging the elements of the vectors across the missions.

We produce the topical profile for the 40K users in the dataset starting from the 3 months of query log. For each

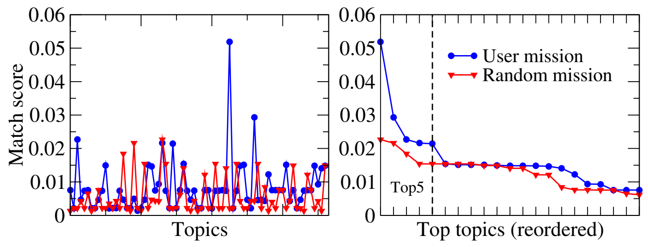

Figure 9: Example of match of missions sequences on a topical user profile. On the x axis are the topics in the profile; on the y axis are the match scores of the mission sequences for each topic. The sequence originated by the same user has more spikes compared to the sequence of the random user. When the values of the each curve are sorted, the best matching sequence is evident by looking at the top-N scores.

user we select two sequences of missions from the 30 days right after the 3 months considered. One sequence comprises of all the missions generated by the same user in the 30-day period. The other is a sequence of equal length generated by another user chosen at random among all other users.

Intuitively, a mission is likely to match at most a few of the several, possibly different topics in a user profile. Given this intuition, to decide which of the two sequences best matches the profile, we focus on the top-N ($N = 5$) elements of the match vectors between topic profile and mission sequences. We then apply a simple majority rule, i.e., which sequence has the most elements with higher match scores. This idea is exemplified by Figure 9.

Using this technique we are able to detect the user's own sequence in 65% of the cases. We stress that the mission sequences of randomly selected users are strongly biased toward high-frequency queries such as "facebook," "amazon," and so on. Since these are shared by a large number of users, any user profile is likely to match them, leading to a decay in detection performance. For this reason we divided the random sequences into three sets according to the average frequency of their queries. The accuracy rises to 72% when considering the sequences with lower frequency queries and drops to around 55% when considering the sequences with higher frequency queries. 65% success in prediction can be considered a good result. Even if the interest of a given user would presumably be quite stationary *within a particular domain*, in web search, where a much wider range of suitable topics is available, the user focus can be often inconstant in time, independent by past search sessions and made even more variable by bursty search activity triggered by external events (e.g., "Michael Jackson death"), hardly predictable by looking only at the user history. This issues make this prediction task much more difficult compared to domain-specific predicition or recomendation.

9. CONCLUSIONS

The *behavior* of the users in submitting queries to a search engine, including the implicit and explicit information that their actions reveal about their search intent, is a crucial element to determine what is the *topic* of a query or of a sequence of search actions. We introduce a novel definition of topic in the context of query log analysis and propose a topic extraction algorithm based on agglomerative clustering of sequences of queries that exhibit a coherent user intent. Our algorithm relies on a semi-supervised classifier that can tell if two query sets are topically coherent with excellent accuracy (AUC 0.95). We compare our method with a graph-based clustering baseline, showing its advantages on query coverage and on the trade-off between purity and resource coverage of the clusters. Finally, we define the *topical profile* of a user in terms of a topic vector that best defines the user search history. With our classifier we are able to discriminate a query sequence submitted by the profiled user from a random query sequence 72% of the time in the best case scenario.

One could consider more sophisticated baselines, for instance combining click co-occurrence with lexical similarity features, or even using query clustering algorithms based on alternative paradigms [7, 18, 11]. A more direct evaluation could be achieved with the golden standard of human judgments on the quality of the topics. A preliminary effort in this direction has pointed to the difficulty of human inspection of topic quality, as well as the challenge of identifying a suitable tradeoff between topic specificity and coverage.

This work opens several research directions. In particular, we are working on the formulation of a user-to-user similarity metric based on topics that can overcome the sparsity problem of similarity metrics based on exact query matches. Finally, we want to explore in greater depth the potential of the topical profiling technique to predict future search activity and provide novel search recommendation services.

10. REFERENCES

[1] P. Anick. Using terminological feedback for web search refinement: a log-based study. In *SIGIR '03*, 2003.

[2] R. Baeza-Yates. Graphs from search engine queries. In *SOFSEM'07: 33rd conference on Current Trends in Theory and Practice of Computer Science*, pages 1–8, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *KDD'07: 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 76–85, New York, NY, USA, 2007. ACM.

[4] N. Balasubramanian and S. Cucerzan. Automatic generation of topic pages using query-based aspect models. In *CIKM'09: 18th ACM conference on Information and knowledge management*, pages 2049–2052, New York, NY, USA, 2009. ACM.

[5] A. L. Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005.

[6] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *KDD'00: 6th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM.

[7] S. M. Beitzel, E. C. Jensen, D. D. Lewis, A. Chowdhury, and O. Frieder. Automatic classification of web queries using very large unlabeled query logs. *ACM Transactions on Information Systems*, 25, April 2007.

[8] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and

applications. In *CIKM'08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 609–618, New York, NY, USA, 2008. ACM.

[9] I. Bordino, C. Castillo, D. Donato, and A. Gionis. Query similarity by projecting the query-flow graph. In *SIGIR'10*, pages 515–522. ACM, 2010.

[10] U. Brandes, D. Delling, M. Höfer, M. Gaertler, R. Görke, Z. Nikoloski, and D. Wagner. On Finding Graph Clusterings with Maximum Modularity. In *WG'07: Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, 2007.

[11] H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD'08: 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883, New York, NY, USA, 2008. ACM.

[12] S.-L. Chuang and L.-F. Chien. A practical web-based approach to generating topic hierarchy for text segments. In *CIKM'04: 13th ACM international conference on Information and knowledge management*, pages 127–136, New York, NY, USA, 2004. ACM.

[13] 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: 19th international conference on World wide web*, pages 321–330, New York, NY, USA, 2010. ACM.

[14] L. Fitzpatrick and M. Dent. Automatic feedback using past queries: social searching? In *SIGIR '97*, 1997.

[15] S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[16] S. Fortunato and M. Barthèlemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, Jan 2007.

[17] S. Gollapudi and R. Panigrahy. Exploiting asymmetry in hierarchical topic extraction. In *CIKM'06: 15th ACM international conference on Information and knowledge management*, pages 475–482, New York, NY, USA, 2006. ACM.

[18] X. He and P. Jhala. Regularized query classification using search click information. *Pattern Recognition*, 41:2283–2288, July 2008.

[19] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR'99*, pages 50–57, 1999.

[20] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM'08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 699–708, New York, NY, USA, 2008. ACM.

[21] R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *CIKM '08: 17th ACM conference on Information and knowledge mining*, pages 699–708, New York, NY, USA, 2008. ACM.

[22] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 78(4), 2008.

[23] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato. Finding statistically significant communities in networks. *PLoS ONE*, 6(4), 04 2011.

[24] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW'08: Proceedings of the 17th international conference on World Wide Web*, pages 695–704, New York, NY, USA, 2008. ACM.

[25] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6:161–179, March 1995.

[26] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69, 2004.

[27] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *KDD'05: 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 239–248, New York, NY, USA, 2005. ACM.

[28] V. V. Raghavan and H. Sever. On the reuse of past optimal queries. In *SIGIR '95*, 1995.

[29] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[30] L. Sarmento, A. Kehlenbeck, E. Oliveira, and L. Ungar. Efficient clustering of web-derived data sets. In *MLDM '09: 6th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 398–412, Berlin, Heidelberg, 2009. Springer-Verlag.

[31] W. Song, Y. Zhang, T. Liu, and S. Li. Bridging topic modeling and personalized search. In *COLING'10: 23rd International Conference on Computational Linguistics: Posters*, pages 1167–1175, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[32] A. Veilumuthu and P. Ramachandran. Intent based clustering of search engine query log. In *CASE'09: 5th IEEE international conference on Automation science and engineering*, pages 647–652, Piscataway, NJ, USA, 2009. IEEE.

[33] J.-R. Wen, J.-Y. Nie, and Z. Hong-Jiang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20:59–81, January 2002.

[34] I. H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, 1999.

[35] J. Ye, J.-H. Chow, J. Chen, and Z. Zheng. Stochastic gradient boosted distributed decision trees. In *CIKM'09: 18th ACM conference on Information and knowledge management*, pages 2061–2064, New York, NY, USA, 2009. ACM.

[36] J. Yi and F. Maghoul. Query clustering using click-through graph. In *WWW'09: 18th international conference on World wide web*, pages 1055–1056, New York, NY, USA, 2009. ACM.

[37] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to cluster web search results. In *SIGIR'04: 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217, New York, NY, USA, 2004. ACM.