CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2011; **0**:1–15 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Mining query log graphs towards a query folksonomy

A. P. Francisco[1,†,∗], R. Baeza-Yates[2] and A. L. Oliveira[1]

[1] *INESC-ID / CSE Dept, IST, Tech Univ of Lisbon, Portugal*
[2] *Yahoo! Research Barcelona, Spain & Santiago, Chile*

SUMMARY

The human interaction through the web generates both implicit and explicit knowledge. An example of an implicit contribution is searching, as people contribute with their knowledge by clicking on retrieved documents. When this information is available, an important and interesting challenge is to extract relations from query logs, and, in particular, semantic relations between queries and their terms. In this paper we present and discuss results on query contextualization through the association of tags to queries, *i.e.*, query folksonomies. Note that tags may not even occur within the query. Our results rely on the analysis of large query log induced graphs, namely click induced graphs. Results obtained with real data show that the inferred query folksonomy provide interesting insights both on semantic relations among queries and on web users intent.

KEY WORDS: query folksonomies; query log analysis; graph mining; knowledge discovery

INTRODUCTION

Nowadays the Web is the biggest representation of human knowledge, where people contribute with content either explicitly or implicitly. An example of an implicit contribution is searching, as people contribute with their knowledge by clicking on retrieved documents. Thus, queries submitted to search engines carry implicit knowledge and they can be seen as equivalent to tags associated to clicked documents. An interesting challenge is then to extract relevant semantic relations from query logs, which have several interesting applications. For instance, ranking algorithms, query recommendation systems and advertisement systems integrate such semantic information to improve their results.

In this paper we discuss query classification, tagging and meaning. Queries have usually less than three words, which may have several different meanings. The main problem is then how to identify and distinguish the different meanings of a given query, which we address here through a query folksonomy. A folksonomy is usually taken as content classification within a given domain through collaborative tag

∗Correspondence to: INESC-ID, Rua Alves Redol 9, 1000-029 Lisboa, PORTUGAL / E-mail: aplf@ist.utl.pt
†Work done while visiting Yahoo! Research Barcelona.

annotation. Folksonomies become widespread in recent years as part of many social Web applications, where users can collectively classify and annotate resources. It turns out that users implicitly provide tags while searching, namely URL tags that lead to an implicit URL folksonomy. In this context we take a step further and obtain a query folksonomy, *i.e.*, we associate tags to queries based on common clicked URLs. Note that in our work, although we use common clicked URLs, we only use terms from queries as tags. Note that this is rather different from URL tagging or URL folksonomies. We use click-data to infer relationships and similarities among queries. Then, by finding closely related queries and relevant terms, we are able to define a hierarchical query folksonomy by associating tags to queries. As we should see, this approach may associate a tag to a query even if that tag does not make part of the query, leading to query contextualization, an important feature for query recommendation systems. Moreover, by providing a hierarchical query folksonomy, we have a fine-grained query categorization, being able to distinguish queries at different category levels and to identify query specializations.

Our approach relies on graphs to represent relations both among queries and between queries and URLs. We start with the bipartite graph of queries and URLs, where a query q and an URL u are connected if a user clicked in the URL u that was an answer for the query q. We also know how many times a given URL u was clicked for each query and, thus, we weight each edge accordingly to click frequency. A second graph has queries as nodes and we add an edge between two queries whenever they share at least a common clicked URL. Each edge is also weighted by computing a similarity score between queries, such as a vector representation of the queries in the high dimensional space of all unique URLs. A more frequent approach is to define a similarity measure among queries ignoring the common clicked URLs. However it is more difficult to understand why queries are similar and it can add noise to data already noisy.

Within this line of work, graph mining techniques are crucial to uncover relations in query graphs. According to SearchEngineWatch.com, the number of queries of large search engines per day is of the order of hundreds of millions. By considering just a one day query log, the query graph would have tens of billion edges. Thus, analyzing such huge graphs is a hard task, that becomes even harder if we take into account similarity weights on the edges. On the other hand, the number of potential relations and their applications is huge.

Our study follows recent works on the analysis of query graphs [1, 2], which introduce the notion of click induced graph and present several results concerning semantic relations among queries. Here we propose three main contributions. First, given the existence of noisy relations among queries mostly caused by multi-topical URLs, we start by discussing how to detect such URLs, proposing a new heuristic. Second, we study how recent results on graph clustering can improve the extraction of semantic relations from query graphs and contribute to query classification. We tackle the problem of clustering click induced graphs, namely we discuss an efficient hierarchical clustering method for these large weighted graphs. We use a well known local optimization approach applied to seed sets, that may however fail if we choose the wrong seeds. Thus, we propose a suitable core enumeration procedure to select seed sets. Third, given a hierarchical clustering, we discuss the inferred semantic relations among queries and how the clustering can induce a query folksonomy. Note that although folksonomies are not usually hierarchical, in our case label specialization allows the creation of a hierarchical folksonomy. To evaluate our approach we use a sample of a query log of Yahoo! search engine and we compare our results with a query classification obtained by mapping queries over the Open Directory Project (ODP) categories. The idea is to analyze how much of the knowledge expressed in queries is different from traditional topic classification.

This is an extended and revised version of a preliminary work already published [3].

RELATED WORK

Query logs record all the interactions of users with a search engine and, thus, they constitute an invaluable resource of information about users behaviour and wisdom. In the recent years there has been an increasing amount of literature on studying properties, models, and algorithms for query-log data analysis. In this context, query similarity analysis has been shown to be extremely effective for unveiling user querying patterns and interests, with several applications such as query recommendation systems and other real time applications. Most of the work on query similarity is related to query expansion or query clustering. Here we mention only the most closely related papers to our work.

Clustering similar queries is a common task in many applications such as query recommendation systems. Wen *et al* [4] proposed to cluster similar queries using four notions of query distance: (1) based on keywords or phrases of the query; (2) based on string matching of keywords; (3) based on common clicked URLs; and (4) based on the distance of the clicked documents in some predefined hierarchy. As the average number of words in queries is small and the number of clicks in the answer pages is also small [5], notions (1) and (2) generate distance matrices that are very sparse. For notion (4) we need a concept taxonomy and the clicked documents must be classified into that taxonomy as well, something that usually requires direct human intervention and that cannot be done in a large scale. Although notion (3) can generate also sparse distance matrices, the sparsity can be greatly reduced by using large query logs. Previous works have used notion (3), such as Befferman and Berger [6], or even variants combining (1) and (3) as well as other simpler features such as in Zaiane and Strilets [7].

Baeza-Yates *et al.* [8, 9] used the content of clicked Web pages to define a term-weight vector model for a query. They consider terms in the URLs clicked after a query. Each term is weighted according to the number of occurrences of the query and the number of clicks of the documents in which the term appears. Then, the similarity of two queries is equivalent to the similarity of their vector representations, using the cosine distance function. This notion of query similarity is based on common clicked URLs as (3) and has several advantages. First, it is simple and easy to compute. On the other hand, it makes it possible to relate queries that happen to be worded differently but stem from the same topic. Therefore, semantic relationships among queries are captured. More recently, Shen *et al.* [10] also used the notion (3) to cluster similar queries and build a query taxonomy. As Baeza-Yates *et al.*, they also consider the terms in the clicked documents instead of the terms in the queries. In this paper we represent each query in a high dimensional space, where each dimension corresponds to a unique URL, and the weights are defined accordingly to click frequency. This notion of similarity uses common clicked URLs and it was introduced by Baeza-Yates and Tiberi [1] to analyze a very large query log. They define semantic relations such as equivalence or specificity based on different set conditions among the set of clicked URLs. Using the ODP they found a precision up to 83% on the relations discovered and also that the ones not found were too specific to appear in ODP. More recently, we [2] have further studied the query graph generated by such similarity relations and we found that even a simple clustering approach can produce interesting results. In the present paper we further improve these results.

The work by Chuang *et al.* [11, 12, 13, 14] also uses query logs to build a query taxonomy to also cluster answers. However they do not use any user feedback, like user clicks. This idea of building a taxonomy based on queries is extended in [15], but this is not the same as building a taxonomy of

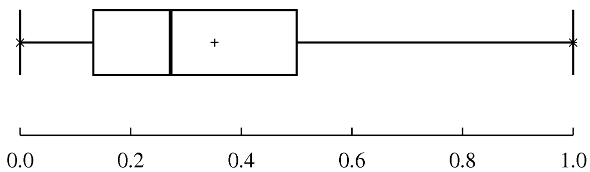

Figure 1. Edge weight statistics. The box plot details the minimum at 0.0, the lower quartile at 0.132, the median at 0.273, the upper quartile at 0.5, the maximum at 1.0 and the mean at 0.352.

the queries, which is what we call a query taxonomy or folksonomy. Later, Dupret and Mendoza [16] used the rank of clicked URLs to define relations among queries. They recommend better queries by generating query relations that can be associated to parts of a query taxonomy.

CLICK INDUCED GRAPH

Let $\mathcal{Q}$ be the set of queries and $\mathcal{U}$ be the set of URLs. Given a query $q \in \mathcal{Q}$, the cover of q is the set of URLs clicked by q. Let $\mu : \mathcal{Q} \to 2^{\mathcal{U}}$ be a function that maps each query q to its cover set $\mu(q) \subseteq \mathcal{U}$. The *click induced graph* $G = (V, E)$ is an undirected graph with queries as nodes and where exists an edge between two queries whenever they share at least one common clicked URL. Formally, $V = \mathcal{Q}$ and $E \subseteq V \times V$ is such that $(q_1, q_2) \in E$ if and only if $\mu(q_1) \cap \mu(q_2) \neq \emptyset$.

In what follows we will refer to the weighted click induced graph. Edges are weighted accordingly to the cosine similarity of the queries they connect. Thus, for $(q_1, q_2) \in E$, the weight is given by

$$\sigma(q_1, q_2) = \frac{\sum_{u \in \mu(q_1) \cap \mu(q_2)} \rho(q_1, u) \rho(q_2, u)}{\sqrt{\sum_{u \in \mu(q_1)} \rho(q_1, u)^2} \sqrt{\sum_{u \in \mu(q_2)} \rho(q_2, u)^2}},$$

where $\rho \quad \mathcal{Q} \times \mathcal{U} \to 0, 1$ is a function such that $\rho(q, u)$ is the frequency ratio with which the URL u was clicked for the query q.

Data Set

For experimental evaluation we considered a query log piece from the Yahoo! search engine. The data was collected in April 2005 and contains 2,822,337 queries with at least one clicked URL and 4,927,980 different URLs. From these, only 660,910 URLs were clicked for more than one query and these are the relevant ones since we are interested in common clicked URLs. On average, each query has 2.39 distinct clicks and each URL is clicked by 1.37 distinct queries. Both click distributions, per query and per URL, follow a power law, with exponents 3.50 and 2.59, respectively. Queries comprise 554,380 different terms.

The click induced graph for this data set has as many vertices as queries, *i.e.*, 2,822,337 vertices, and 359,881,327 edges. The degree distribution follows a power law with exponent 1.50 and the weights are distributed as depicted in Figure 1, showing many noisy edges. The graph has 1,568,617 connected components, the giant component contains 81,156 vertices, about 34.8% of the vertices, and 1,407,321 components are singleton vertices. The giant component is very dense, with the second smallest connected component having only 64 vertices.

Noisy Edges Detection and Removal

The main purpose of the click induced graph is to represent semantic relations between queries and to enable knowledge extraction. Semantic relations can however have low quality introducing noise. In what concerns the edge weights for the studied query log, we have that about 75% of edges are weighted with values below 0.5 and 50% with values below 0.273. Thus, there are many connections between queries which are not much similar. Most of this connections are due to URLs covering dubious topics, several topics or very general topics. These URLs are usually denoted as multi-topical, being examples many e-commerce and directory sites.

An approach to remove noise is to ignore contributions from multi-topical URLs. Baeza-Yates and Tiberi [1] suggested that multi-topical URLs are the ones that contribute more to edges with low weights. Then, we regenerate the click induced graph ignoring such URLs. Although this approach reduces the graph size removing the noise, we observed that URLs which contribute more to low weighted edges also may contribute more to high weighted edges. Moreover, we also observed a strong positive correlation between the number of queries covered by a URL and the number of contributions to edge weights. In Figure 2 we plot the geometric mean of the URLs weight contribution versus their size for our query log data. These results are due to the high number of queries for which a given multi-topical URL is the only clicked URL, generating many high weighted relations in the graph.

To solve this problem, we considered as documents the terms among the set of queries covered by each URL. Let $\mu^{-1}{:}\mathcal{U} \to 2^{\mathcal{Q}}$ be the "inverse" function μ that maps each URL u to its coverable set of queries $\mu^{-1}(u) \subseteq \mathcal{Q}$. The set of queries $\mu^{-1}(u)$ was taken as a document d_u associated to each URL u and, then, we computed the tf-idf score for each term and for each document as usual. We observed that multi-topical URLs have a low average tf-idf score. This is true even when we select the high related queries for which those URLs were clicked. Therefore, we propose to compute the maximum tf-idf among the bag of terms associated to each URL and select the URLs with lowest score as multi-topical candidates. In Figure 2 we depict the maximum tf-idf score against URL coverage size for the query log analyzed. As we discuss ahead, this approach effectively reduces the size of the graph keeping its properties, such as the size of the giant component and the weight distribution. We should note that this is consistent with previous results [1].

Next, we sorted the URLs by the maximum tf-idf score and we regenerate the click induced graph ignoring 0.05% of the URLs with lowest score, namely ignoring the 330 URLs with lowest tf-idf score. In Figure 2 we provide the distribution of tf-idf scores for the analyzed query log and, by selecting just 0.05% of the URLs we are filtering the click induced graph in a conservative way. Note that many of the selected URLs have a large coverage and, maybe unexpectedly, they are not spam URLs.

The resulting click induced graph has 23,177,430 edges, about 6.44% of the size of the full click induced graph. Since we continue having low weighted edges, we remove 10% of the edges with lowest score, all of them having a weight lower than 0.043. Thus, the filtered click induced graph has 20,974,257 edges and 1,648,649 connected components. The giant component contains 861,903 vertices and the second smallest connected component has only 64 vertices. There are now 1,474,249 singleton vertices. The degree distribution follows a power law with exponent 1.65. Therefore the

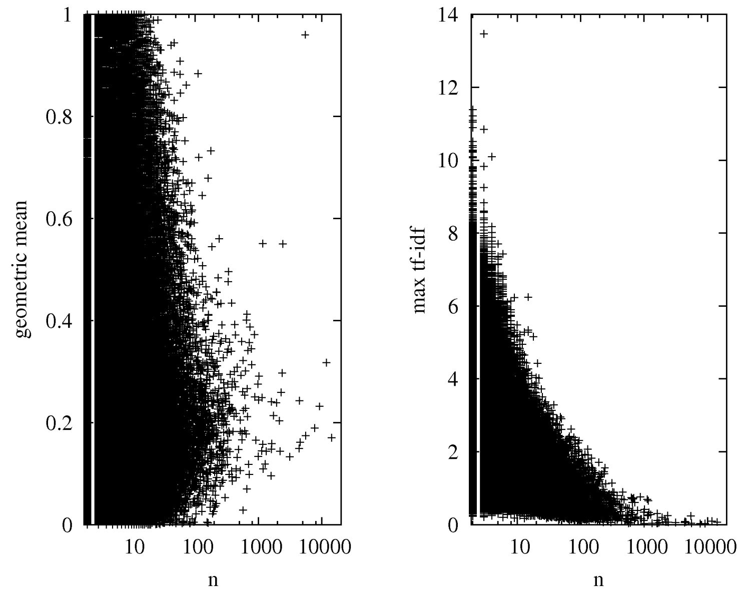

Figure 2. On left, URL weight contribution geometric mean versus URL coverage size n. On right, given the set of terms associated to the queries covered by each URL, URL maximum tf-idf score versus URL coverage size n.

approach to remove noise and multi-topical URLs dramatically reduces the size of the click induced graph, which becomes much more sparse while keeping its core structure almost unchanged. This is an important fact since we can effectively reduce the noise without losing much information [1]. Moreover, we can neglect the effect of this filtering technique on the final clustering since, as we discuss later, the hierarchical clustering only becomes interesting for thresholds on weights above 0.1.

HIERARCHICAL CLUSTERING IN GRAPHS

One of the hardest problems in graph mining is detecting graph community structure or graph clustering. The notion of community and the first formalizations of the concept have been proposed in the social sciences. Usually, communities are groups of vertices such that the number of edges within the groups is higher than the number of edges among different groups. The general aim of community finding and graph clustering methods is to detect meaningful divisions by inspecting the structure of the network. This problem has recently attracted a large interest and, for a deep review on this topic, we refer the reader to a review on complex networks by Boccaletti *et al.* [17] or a more recent survey on community finding by Fortunato [18].

In this paper we follow a two stage approach. We find a set of seed sets and, then, we apply a well known local optimization method. Several methods have been proposed based on the optimization of a given score [19, 20, 21, 22, 23], in particular to detect overlapping clusters based on global partition and local expansion [21, 22]. As pointed out by several authors, the main problem is how to choose seed sets. Usually, a well known spectral partitioning method is used to generate seed clusters, for instance multilevel bisection. Although the results are promising, such approaches inherit some of the drawbacks of standard multilevel methods when we are looking for overlapping communities. This problem becomes even harder when we have weighted graphs. To solve this problem, we propose a core enumeration method based on a vertex similarity score, where a core is a densely connected subgraph which usually occurs within communities or clusters and that, by local optimization, leads to the full cluster.

Finding Cores

We could define the initial seed sets by taking many different approaches. Since we are interested in forming clusters of similar queries, a simple approach could be thresholding the edge weights. Another approach could be find the nearest neighbors. But we know that an URL may induce a clique in the graph and, in particular, it can induce a clique with high weights. If we follow the simple approach we could join two cliques even if they are connected by a single edge, *i.e.*, chaining effects may occur. Thus we take a different approach where not only weights are considered, but also vertex structural similarity, *i.e.*, how many neighbors are shared among connected vertices. Note that by taking into account the connectivity around connected vertices instead of considering just edge weights, we try to overcome known drawbacks of single-linkage approaches, such as the sensibility to outliers and chaining effects. Let $G = (V, E)$ be a graph and $\sigma : E \longrightarrow \mathbb{R}_0^+$ the edge weight function. Given two connected vertices $(v_1, v_2) \in E$, their structural similarity takes values in 0, 1 and is given by

$$\eta(v_1, v_2) = \text{avg}(v_1, v_2)\cos(v_1, v_2),$$

where $\text{avg}(v_1, v_2)$ is the weight mean among common neighbors, *i.e.*,

$$\text{avg}(v_1, v_2) = \frac{\sum_{w \in N(v_1) \cap N(v_2)} \sigma(v_1, w) + \sigma(v_2, w)}{|N(v_1) \cap N(v_2)|}$$

and $\cos(v_1, v_2)$ is a cosine similarity based score given by

$$\cos(v_1, v_2) = \frac{2\,\sigma(v_1, v_2) + \sum_{w \in N(v_1) \cap N(v_2)} \sigma(v_1, w)\sigma(v_2, w)}{\sqrt{1 + \sum_{w \in N(v_1)} \sigma(v_1, w)^2}\sqrt{1 + \sum_{w \in N(v_2)} \sigma(v_2, w)^2}},$$

with $N(v)$ being the set of neighbors of v. The term $\cos(v_1, v_2)$ measures how similar are the two vertices with respect to common neighbors and respective weights. $\cos(v_1, v_2)$ takes value 1.0 whenever the vertices v_1 and v_2 share all neighbors, even if they are connected through edges with low weights. Thus, we introduced the term $\text{avg}(v_1, v_2)$ to distinguish common neighbors connected through low weighted edges from common neighbors connected through high weighted edges.

Given $\varepsilon > 0$, we say that $(v_1, v_2) \in E$ is a *core edge* if $\eta(v_1, v_2) \geq \varepsilon$. A set of vertices $C \subseteq V$ is a *core* in G if all $v \in C$ form a connected component composed only of core edges. By choosing different values for $\varepsilon > 0$, we can enumerate the set of cores in a graph at different resolutions. By considering the edges in decreasing order with respect to η, we obtain a hierarchy of cores. This method takes $O(|E| \max(\Delta, \log |V|))$ time, with Δ the maximum vertex degree: $O(|E|\Delta)$ time to compute the similarity for each edge; $O(|E|\log|V|)$ time to sort the edges in decreasing order; and $O(|E|)$ time to obtain the hierarchical clustering.

Local Optimization

Given a hierarchy of cores, we take each core as a seed set and we perform a local optimization step based on the local partition method proposed by Chung [24]. This method takes a seed set and uses a heat kernel [25] to expand it, which is a typical random walk and that was shown to provide good results [24]. Given a graph G, the transition probability matrix W of a typical random walk on G is defined as $W = D^{-1}A$, where A is the adjacency matrix of G (since G is undirected and weighted, A is symmetric and its entries are the edge weights), and D is a diagonal matrix with $D_{vv} = \sum_{w \in N(v)} \sigma(v, w)$. The heat kernel is then defined as $e^{-\alpha L}$, where $L = I - W$ with I the identity matrix. The parameter $\alpha > 0$ is known as the temperature and it plays an important role as the heat diffusion coefficient. We did several experiments and, within the scope of this paper, different values of alpha do not change much the results. Thus, in what follows, α is equal to 1.0. Given a preference vector p_0 obtained from a seed set, we use the following discrete approximation $p_\alpha = p_0 \left(I - \frac{\alpha}{k} L\right)^k$, where k is the number of iterations. Yang *et al.* [26] used this approximation in a different context and they proposed a heuristic to find the minimum number of iterations for a given approximation error threshold. In particular, if the graph is connected, then p_α converges to the stationary distribution. However, we are not interested in this limiting distribution but rather in the distributions obtained after a small number of steps. Given a seed set, we define p_0 as the uniform distribution over the seed set and we simulate several heat kernel steps, computing the probability distributions p_α. After each step, we sort the vertices in descending order according to the degree-normalized probabilities $r_\alpha(v) = p_\alpha(v)/d(v)$, where $d(v) = \sum_{w \in N(v)} \sigma(v, w)$. This ordering defines a collection of sets $\{C_i\}_{i=1}^{\ell}$, where $C_i = \{v_j \mid 1 \leq j \leq i\}$ and ℓ is the number of vertices v such that $r(v) \neq 0$. We select the set C_i that minimizes the conductance, also known Cheeger ratio,

$$\Phi(C) = \frac{\partial(C)}{\min(\mathrm{Vol}(C), \mathrm{Vol}(V \setminus C))},$$

where the volume $Vol(C)$ and the cut size $\partial(C)$ are given by

$$\mathrm{Vol}(C) = \sum_{v \in C} \sum_{w \in N(v)} \sigma(v, w) \qquad \text{and} \qquad \partial(C) = \sum_{v \in C} \sum_{w \in N(v) \setminus C} \sigma(v, w),$$

respectively. Note that conductance measures the faction of the weight of the edges incident on a cluster C that are connected to vertices outside of C, being trivially minimized if C is V. Although conductance may not be enough for evaluating a complete clustering (usually we need other measures such as the sum of inter-cluster weights [27]), it has been shown to work well for single cluster evaluation and local clustering optimization on scale-free networks [24, 28], as is the case with the click induced graph. Usually we are interested in a reasonable expansion of the seed set. In this paper we stop after finding the first local minimum. The cut sizes and the volumes for all sets C_i can be computed in $O(\mathrm{Vol}(C_i))$ time, by determining the change to C_i due to the addition of vertex v_{i+1}. This process is referred to as a *sweep* [28].

Figure 3. Tag cloud for the giant component of the studied click induced graph. The font size is proportional to the tf-idf score for each term. This image was created with the help of Wordle™.

INDUCED QUERY FOLKSONOMY

A folksonomy is usually taken as content classification through tag annotation. Given a hierarchical clustering on a graph, we show here how we obtain a folksonomy by associating to each node in the hierarchy the most relevant terms, *i.e.*, the tags. Given a node in the hierarchical clustering tree, such association takes into account the queries in the leafs of the underlying subtree. We identify the most relevant terms for each internal node in the clustering tree as follows. We identify the level of the tree for the given node and we compute the set of queries associated to each node at that level by grouping together all queries in the respective subtree. Moreover, we identify the set of terms for each node simply by inspecting the set of queries. Those sets of terms become our documents and we infer the most relevant terms for each node by computing the tf-idf score for each term in each document. Those terms become the tags associated to the queries on that node.

Let $\mathcal{Q}$ be the set of queries and $\{C_1, ..., C_k\}$, with $k \in \mathbb{N}$, a clustering of Q for a given cut of the hierarchical clustering. Formally, given a cluster C of closely related queries, we compute the tf-idf score for each term t within a cluster C, taking each cluster as a document. Then, for each $q \in C$, the most relevant terms become their tags. Note that in particular this approach provides a tag cloud for each cluster C. In Figure 3, we can see the tag cloud for the biggest cluster we obtained, *i.e.*, for the giant component of the click induced graph described before. We are able to distinguish the meaning of queries by identifying clusters at different levels, *i.e.*, cuts of the hierarchical clustering.

The folksonomy labels are selected by sorting the terms by decreasing relevance. We can either select the first terms or just the ones scored above a given threshold. The later approach may conduce to unlabeled nodes, which we may prefer instead of bad quality labels. In particular, since click induced graphs are scale-free and have a giant component, internal nodes corresponding to the giant component,

Table I. Clustering statistics for different values of ε, where $\mathcal{C}$ is the set of non-singleton clusters, $\mathcal{S}$ is the set of singleton clusters, C denotes non-singleton clusters, $\hat{\Phi}$ is the clustering average conductance, n_{core} is the number of core vertices, $n_{\text{non-sing}}$ is the number of queries in non-singleton clusters and n_{overlap} is the number of queries in more than one cluster. The percentages refer to the increasing in the number of non-singleton queries and to the number of vertices in more than one cluster after optimization, respectively. The hierarchical clustering contains 1,348,088 distinct queries.

ε	$\lvert\mathcal{C}\rvert$	avg $\lvert C\rvert$	max $\lvert C\rvert$	$\lvert\mathcal{S}\rvert$	$\hat{\Phi}$	n_{core}	$n_{\text{non-sing}}$	%	n_{overlap}	%
0.0	174,400	7.73	861,903	0	0.00	1,348,088	1,348,088	0.00	0	0.00
0.1	213,557	6.73	768,174	27,524	0.04	1,225,791	1,320,564	7.73	114,723	8.68
0.2	228,550	6.38	629,416	81,621	0.07	1,104,657	1,266,467	14.65	181,833	14.36
0.3	224,683	5,58	245,050	213,357	0.09	962,193	1,134,731	17.93	112,549	9.92
0.4	210,532	4.94	9,421	346,550	0.08	815,791	1,001,538	22.77	34,108	3.41
0.5	180,812	4.84	1,385	496,583	0.08	656,720	851,505	29.66	21,020	2.47
0.6	147,228	4.50	1,331	696,712	0.07	507,507	651,376	28.35	9,342	1.43
0.7	103,553	4.27	1,174	909,602	0.05	353,592	438,486	24.00	2,958	0.67
0.8	82,701	3.62	235	1,049,433	0.03	254,514	298,655	17.34	619	0.21
0.9	61,792	2.92	113	1,167,521	0.01	168,320	180,567	7.28	30	0.01
1.0	48,547	2.29	14	1,237,095	0.00	110,993	110,993	0.00	0	0.00

or even to part of it, have bad quality labels which do not bring relevant semantic information - see ahead.

EXPERIMENTAL EVALUATION

We applied the hierarchical clustering method described above to the filtered click induced graph. Given the results achieved on several experiments, as mentioned before, the most relevant parameter is the core threshold ε, which provides different cuts on the hierarchical clustering and that we discuss next. In Table I we provide several statistics for different snapshots of the hierarchical clustering. Since we removed the singleton vertices from the graph, we are considering 1,348,088 vertices. The degree distribution follows a degree power law and the graph contains a giant component, thus the giant cluster for $\varepsilon = 0$ was expected and it coincides with the giant component. In particular for $\varepsilon = 0$, the clusters are precisely the non-singleton connected components in the original graph. Moreover, we can see that the method effectively clusters the giant component. For instance, with $\varepsilon = 0.4$ the biggest cluster is much smaller, about 1.1% of the original giant component. Note also the low values for average conductance Φ, that core vertices are contained in non-singleton clusters and that the cores do not overlap initially.

Semantic Contextualization

Although the clustering is effective, we obtain many small clusters at each level. These correspond to loosely connected clusters that could appear connected if we consider larger query logs. Many are

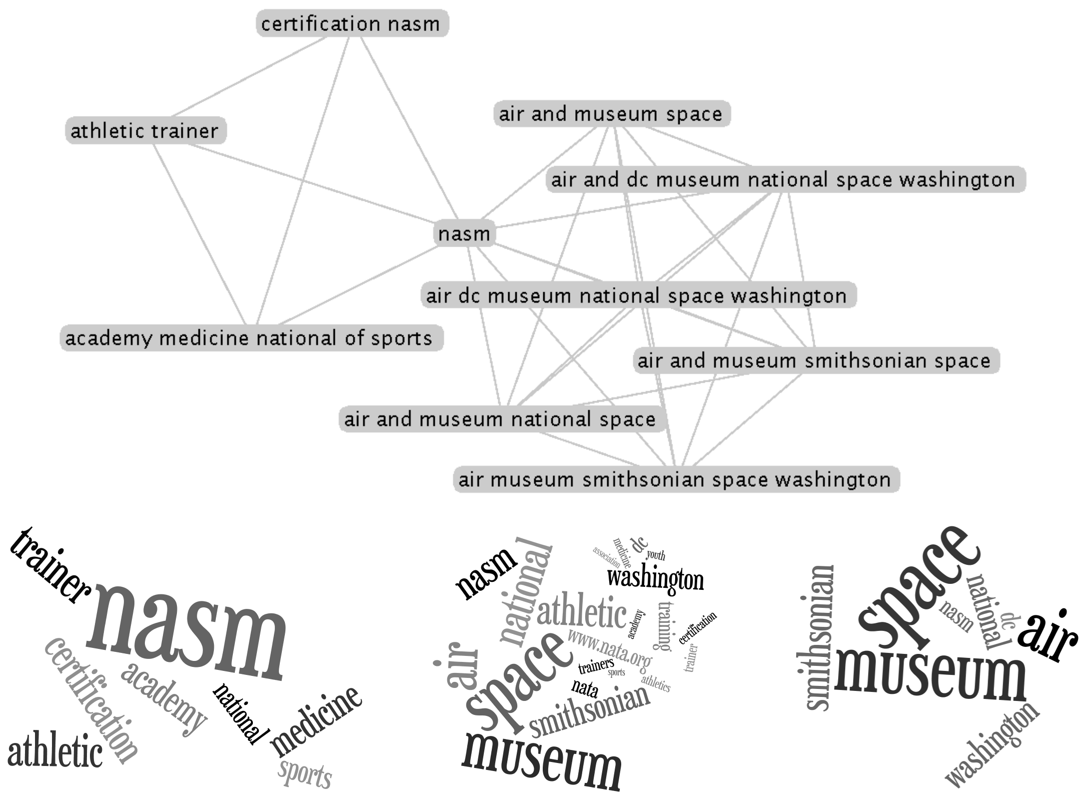

Figure 4. Two overlapping clusters in the hierarchical clustering after local optimization and in a snapshot for $\varepsilon = 0.5$. Below we have the tag clouds for the full cluster (center) and for each of the overlapping clusters.

highly specific queries, such as “53545 clinic in janesville riverview wisconsin”, for which the search engine returns a low number of results and where the user clearly knows what he wants. Navigational queries also fall in this category, examples being “java.com www” or “slashdot.org”.

From the overlaps we can infer relevant information about queries, namely about their ambiguity, context, topics and term polysemy. As an example, let us consider the overlap in Figure 4. We see that “nasm” appears in two different contexts, namely it is an acronym to both the National Air and Space Museum and the National Academy of Sports Medicine.

The obtained clusters provided also interesting insights with respect to web slang, namely term polysemy and semantic relations. By just looking at the cluster in Figure 4, we can infer that NASM has two possible meanings and that one of them should confer a kind of certification. By considering terms within clusters, we can detect that for example “windows”, “mouse” and “wine” are polysemic terms. For instance, the term “wine” appears in several clusters together with terms such as “napa”, “food”, “magazine”, and “noir” relating it to beverages. However, we find at least one cluster where

```
...
fights street video → iraq war pictures
fights street video → iraq helicopter civilian
fights street video → intelligence network homelandsecurity
fights street video → tape caught fights
...
free → virus airlines anti → slow computer virus
free → virus airlines anti → airlines airlines.com
free → virus airlines anti → virus anti free
...
tires goodyear bridgestone → firestone bridgestone tire
tires goodyear bridgestone → tires cooper dunlop
tires goodyear bridgestone → goodyear gemini car
tires goodyear bridgestone → goodyear tires tire
tires goodyear bridgestone → tires bridgestone truck
...
```

Figure 5. Examples of label specialization for clusters which provide also tag refinement for the query folksonomy.

it appears with terms such as "linux" and "windows", clearly relating it to the Wine translation layer for Unix like operating systems. An approach to identify term polysemy is to compare the bag of terms among overlapping clusters. Clearly, if a query is in two clusters but they share few terms, then the query shall be polysemic. Similarly, by analyzing similar words in the same cluster we can detect misspellings.

Given the hierarchical clustering described above, we build the induced query folksonomy. Note that the folksonomy we discuss here is entirely based on user interaction through a search engine. We do not add any other source of data in order to filter or improve it. As mentioned before, the click induced graph is scale-free and has a giant component. We computed the tf-idf score for all clusters at each level and the scores become meaningful only for $\varepsilon > 0.3$. For $\varepsilon \leq 0.3$, the tf-idf score for the giant component takes values between 0.05 and 0.07, and the most relevant term is "free". Thus, in our discussion we focus on the categories with more queries and at clusters for $\varepsilon > 0.3$, since they have higher tf-idf scores and are more informative.

The folksonomy is rather different from usual taxonomies, both because of the click induced graph structure and because of the type of categories found. Note also that in traditional taxonomies the topics are selected beforehand, while we do not have any topic pre-specified. We observed that category paths correspond most of the times to keywords meaningful for users, such as trademarks. It is interesting that, although we consider the undirected version of the click induced graph, we are able to detect query specialization through the hierarchical clustering (see Figure 5).

Nevertheless, some of the categories are somewhat strange. For instance, the second group of categories in Figure 5 joins anti-virus on computers with anti-virus on airlines. The term virus makes sense in both contexts, as it is usual to run such software on computers and biological virus are also a current trend within flights and travels. However, such categorization seems to be wrong and it may occur because of some URL badly clicked.

Table II. Query distribution over the ODP top categories. In this table we map each of the 2,822,337 queries to a single category, the category with highest score.

Category	Queries	%	Category	Queries	%
Adult	105,123	3.72	News	7,672	0.27
Arts	367,702	13.03	Recreation	99,224	3.52
Business	141,512	5.01	Reference	56,112	1.99
Computers	157,313	5.57	Regional	605,483	21.45
Games	69,830	2.47	Science	83,100	2.94
Health	74,261	2.63	Shopping	86,758	3.07
Home	70,152	2.49	Society	187,106	6.63
Kids and Teens	50,539	1.79	Sports	80,522	2.85
			World	282,110	10.00

Comparison with ODP

Evaluating the query classification is difficult since it is very different from traditional directories. Nevertheless, human curated URL directories such as ODP can provide good insights on query classification. Note that URLs in these directories have associated curated descriptions, based on user interests, and that allow us to contextualize queries with some confidence. By querying these descriptions, we are able to identify common categories for different queries, reflecting query similarity even when such queries do not share any term. Thus, in this section, we try to compare the query classification with ODP categories in order to understand how different are these two ways of expressing knowledge. We map all queries over the ODP categories, obtaining several category paths for each query. Then we compare the ODP paths with the induced folksonomy.

Since we had millions of queries, we had to perform such mapping offline. We downloaded the ODP data, available at http://www.dmoz.org, and we fed it to a local installation of the Lucene search engine, available at http://lucene.apache.org. The ODP data consisted of a large set of URL entries, each one with a category, a title and a description. The ODP data set we considered for this paper contains 4,595,111 URL entries and 763,535 distinct categories. We took each URL entry as a document and we indexed all fields, *i.e.*, URL, category, title and description snippet. Lucene was configured to search over all fields and, for each query, to return several categories ranked by relevance. We used the default Lucene scoring function, which combines the Vector Space Model and the Boolean Model to determine the relevance of documents [29]. Note that we did not obtain categories to all queries. By inspecting Table II, we see that 297,818 queries, 10.55%, were not mapped. If we compare with the click induced graph, we have that 67% of these queries are singleton queries. Moreover, 68% of the queries with a score lower than 1.0 as reported by Lucene are also singleton queries in the click induced graph. This is consistent with our observations about the singleton queries, that many of them are ambiguous and uninformative.

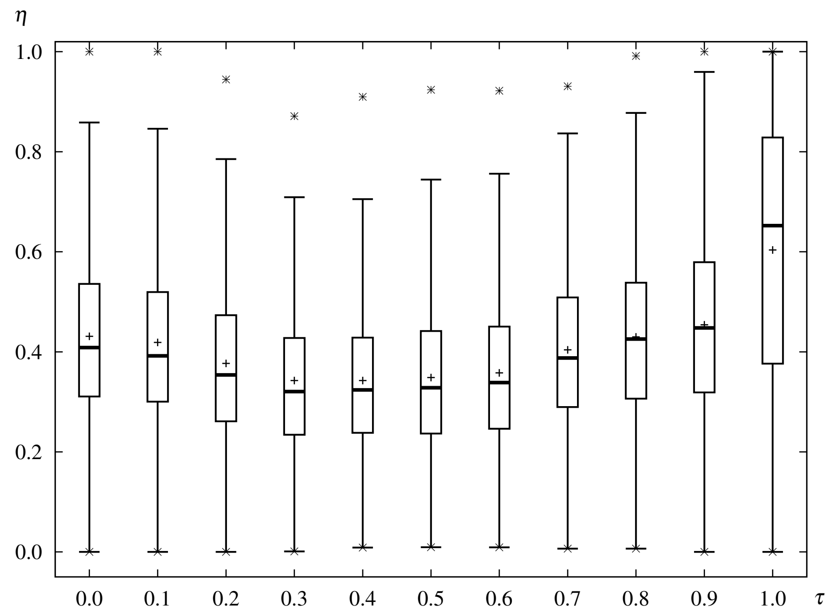

Figure 6. ODP score statistics for the 1000 biggest clusters at different snapshot levels, *i.e.*, different values of ε. For each snapshot, the box plot details the σ_{odp} minimum ($*$), the lower, median, and upper quartile, the maximum ($*$) and the mean ($+$).

The folksonomy labels are not comparable to the categories in the ODP mapping since they are not topic based. Thus, we evaluate the clusters by comparing the common ODP path prefix among the queries. Given two queries q_1 and q_2, we select the two most similar ODP category paths p_1 and p_2, i.e., the ones which share the longest common prefix $\pi(p_1, p_2)$. Then we compute the score

$$\sigma_{\text{odp}}(p_1, p_2) = \frac{|\pi(p_1, p_2)|}{\max\{|p_1|, |p_2|\}},$$

where $|\cdot|$ denotes the path length. The ODP score for a given cluster is the average of the score σ_{odp} for all pairs of queries in that cluster.

For all snapshots of the hierarchical clustering at different values of ε, more than 50% of the clusters have an ODP score higher than 0.5. Since we do not obtain ODP categories for all queries, many clusters have an ODP score of 0.0. In our experiments, depending on the value of ε, we have 16% to 30% of clusters with an ODP score equal to 0.0. We also have 30% to 39% of the clusters with an ODP score equal to 1.0. It is interesting that these clusters are small in the number of queries and that they appear independently of ε. Thus, we may infer that either these clusters are well defined or they are meaningless. It is interesting that this fact supports our previous observation that these clusters usually consist of either navigational queries or ambiguous and uninformative queries.

In order to analyze the remaining clusters, we select the 1000 biggest clusters at different depths, *i.e.*, for different values of ε. Note that, for $\varepsilon \leq 0.9$, the number of clusters with an ODP score of either 0.0 or 1.0 among the 1000 selected is 0%. Figure 6 depicts the ODP score values for the 1000 biggest clusters at different snapshot levels. Note also that the first snapshots have a higher average score, because for $\varepsilon \leq 0.3$ exists a giant component and the remaining clusters are rather small. As we mentioned before, the giant component vanishes for $\varepsilon > 0.3$. Thus, after we cluster the giant component, the score increases with the hierarchical clustering depth, revealing that clusters at higher depths have better quality. This is also supported by the tf-idf scores.

FINAL REMARKS

Queries submitted to search engines can be viewed as an expression of the knowledge of the users. In this paper we discuss how to infer a query folksonomy through the efficient analysis of large query logs and, in particular, of click induced graphs. First we discussed how to filter out noise caused, for instance, by multi-topical URLs, proposing a new method to detect such URLs based on the analysis of the queries for which URLs were clicked. Second we devised and applied a hierarchical clustering method for weighted graphs. This method was shown to be effective and the results revealed effective semantic relations between queries. By building an induced folksonomy, we were able to identify query contextualization and specialization. Another interesting result is the fact that our approach, based only on click-through data generated by the users, provides a query classification much different from the one expected by traditional directories. This points out how hard is query classification and how highly relevant is the implicit knowledge found in query logs.

Although it was not the main aim of this paper, the results described can be used to improve real-time query recommendation and query contextualization, two important practical problems that current search engines have to deal with. Using our approach, query recommendation on a search engine can be improved by providing better contextualization of recommendations, avoiding the bias to common terms and allowing the suggestion of relevant but less frequent terms. The hierarchical clustering takes here an important role since it enables the system to refine the clustering around a given query and, because of the chosen local optimization, it can provide unbalanced clusters revealing less frequent contexts. Given a query, we can apply the local optimization step starting with it (or with a suitable set of queries) and, thus, detect a local cluster. Then, we can compute a tag cloud as in Figure 3 or just the tags ranking and select the most relevant ones. Nevertheless, we can reach a situation as in Figure 4, where the query "nasm" belongs to two clusters and, clearly, a simple local optimization will be not able to detect them. Note that distinguishing different clusters in these cases is of great importance, since they provide a richer contextualization. As we discussed in this paper, this problem can be addressed through core enumeration, which combined with local optimization provides a solution. In this context, it is important to recall that the proposed hierarchical clustering method takes $O(|E| \max(\Delta, \log |V|))$ time, with Δ the maximum vertex degree, *i.e.*, $O(|E|\Delta)$ time to compute the structural similarity for all pairs of connected vertices, $O(|E| \log |V|)$ time to sort the edges in decreasing order, and $O(|E|)$ time to obtain the hierarchical clustering. The local optimization of each core C takes $O(\mathrm{Vol}(C))$ time. Thus, given that the click induced graph is scale-free, the average time for computing the structural similarity is much less than $O(|E|\Delta)$ and the method is usually much faster. With respect to space requirements, we just need to store the graph and the weights, that can be efficiently done using succinct data structures [30]. We should also note that, for the log piece analyzed, all tests were run in a common laptop in a few hours. Since the methods are easily parallelizable, we are able to analyze much larger logs on high-end systems.

The query log studied here is just a case study and the quality of the results can be improved by incorporating more data, *i.e.*, by using larger logs, since more data will consolidate the relations obtained. The efficiency of the proposed methods makes them applicable to much larger graphs, thus making them suitable for the analysis of larger logs and for the extraction of semantic relations from less frequent queries. Keeping applications on sight, it is important to note that both the edges and the queries have probabilities associated and provided by the hierarchical clustering algorithm. Thus, we have confidence measures that are crucial to rank the relations among queries and their cluster membership.

REFERENCES

1. Baeza-Yates RA, Tiberi A. Extracting semantic relations from query logs. *SIGKDD*, ACM, 2007; 76–85.
2. Francisco AP, Baeza-Yates R, Oliveira AL. Clique analysis of query log graphs. *SPIRE'08*, *LNCS*, vol. 5280, Springer, 2008; 188–199.
3. Francisco AP, Baeza-Yates R, Oliveira AL. Mining large query induced graphs towards a hierarchical query folksonomy. *String Processing and Information Retrieval*, *Lecture Notes in Computer Science*, vol. 6393, Springer, 2010; 238–243.
4. Wen J, Mie J, Zhang H. Clustering user queries of a search engine. *Proc. of the 10th International World Wide Web Conference*, W3C, 2001.
5. Baeza-Yates R. Applications of web query mining. *European Conference on Information Retrieval (ECIR'05)*, *LNCS*, vol. 3408, Springer, 2005; 7–22.
6. Beeferman D, Berger A. Agglomerative clustering of a search engine query log. *SIGKDD*, ACM, 1999.
7. Zaiane OR, Strilets A. Finding similar queries to satisfy searches based on query traces. *Efficient Web-Based Information Systems (EWIS)*, 2002.
8. Baeza-Yates R, Hurtado C, Mendoza M. Query clustering for boosting web page ranking. *Advances in Web Intelligence (AWIC'04)*, *LNCS*, vol. 3034, Springer, 2004; 164–175.
9. Baeza-Yates R, Hurtado C, Mendoza M. Query recommendation using query logs in a search engine. *EDBT Workshops*, *LNCS*, vol. 3268, Springer, 2004; 588–596.
10. Shen D, Qin M, Chen W, Yang Q, Chen Z. Mining Web Query Hierarchies from Clickthrough Data. *AAAI'07*, AAAI Press, 2007; 341–346.
11. Chuang SL, Chien LF. Towards automatic generation of query taxonomy: A hierarchical query clustering approach. *IEEE International Conference on Data Mining*, IEEE, 2002.
12. Chuang SL, Chien LF. Automatic query taxonomy generation for information retrieval applications. *Online Information Review* 2003; **27**(5).
13. Chuang SL, Chien LF. Enriching web taxonomies through subject categorization of query terms from search engine logs. *Decision Support System* 2003; **30**(1).
14. Pu HT, Chuang SL, Yang C. Subject categorization of query terms for exploring web users' search interests. *JASIST* 2002; **53**(8).
15. Cheng PJ, Tsai CH, Hung CM, Chien LF. Query taxonomy generation for web search (poster). *CIKM*, 2006.
16. Dupret G, Mendoza M. Automatic query recommendation using click-through data. *IFIP World Computer Congress (WCC'06)*, Springer, 2006.
17. Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU. Complex networks: structure and dynamics. *Physics Reports* 2006; **424**(4-5):175–308.
18. Fortunato S. Community detection in graphs. *Physics Reports* 2010; **486**:75–174.
19. Baumes J, Goldberg M, Magdon-Ismail M. Efficient identification of overlapping communities. *IEEE International Conference on Intelligence and Security Informatics (ISI)*, 2005; 27–36.
20. Lancichinetti A, Fortunato S, Kertész J. Detecting the overlapping and hierarchical community structure in complex networks. *New J. Phys.* 2009; **11**.
21. Leskovec J, Lang KJ, Dasgupta A, Mahoney MW. Community structure in large networks: Natural cluster sizes and the absence of large well-define clusters. *Internet Mathematics* 2009; **6**(1):29–123.

22. Wei F, Qian W, Wang C, Zhou A. Detecting Overlapping Community Structures in Networks. *World Wide Web* 2009; **12**(2):235–261.
23. Zhang S, Wang RS, Zhang XS. Identification of overlapping community structure in complex networks using fuzzy c-means clustering. *Physica A: Statistical Mechanics and its Applications* 2007; **374**(1):483–490.
24. Chung F. The heat kernel as the pagerank of a graph. *Proceedings of the National Academy of Sciences* 2007; **104**(50):19 735.
25. Chung FRK, Yau ST. Coverings, heat kernels and spanning trees. *Electr. J. Comb.* 1999; **6**.
26. Yang H, King I, Lyu MR. Diffusionrank: a possible penicillin for web spamming. *SIGIR'07*, ACM, 2007; 431–438.
27. Kannan R, Vempala S, Vetta A. On clusterings: Good, bad and spectral. *Journal of the ACM* 2004; **51**(3):497–515.
28. Andersen R, Lang KJ. Communities from seed sets. *World Wide Web (WWW'06)*, ACM, 2006; 223–232.
29. Gospodnetic O, Hatcher E, McCandless M. *Lucene in Action*. second edn., Manning Publications, 2009.
30. Boldi P, Vigna S. The webgraph framework I: compression techniques. *World Wide Web (WWW'04)*, ACM, 2004; 595–602.