Clique Analysis of Query Log Graphs

Alexandre P. Francisco[1]*, Ricardo Baeza-Yates[2], and Arlindo L. Oliveira[1]

[1] INESC-ID/IST, Technical University of Lisbon, Portugal
[2] Yahoo! Research Barcelona, Spain & Santiago, Chile

Abstract. In this paper we propose a method for the analysis of very large graphs obtained from query logs, using query coverage inspection. The goal is to extract semantic relations between queries and their terms. We take a new approach to successfully and efficiently cluster these large graphs by analyzing clique overlap and *a priori* induced cliques. The clustering quality is evaluated with an extension of the modularity score. Results obtained with real data show that the identified clusters can be used to infer properties of the queries and interesting semantic relations between them and their terms. The quality of the semantic relations is evaluated both using a tf-idf based score and data from the Open Directory Project. The proposed approach is also able to identify and filter out multitopical URLs, a feature that is interesting in itself.

1 Introduction

Knowledge discovery is one of the main problems in data mining and information retrieval. Human interaction through the Web generated implicit knowledge -or the wisdom of crowds [1]- represents an important path towards improving the discovery of interesting knowledge. 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 important and interesting challenge is then to extract relevant relations from query logs, namely semantic relations between queries and their terms.

Graphs are a natural way to view relations between queries and URLs. As Baeza-Yates and Tiberi [2], we start with the bipartite graph of queries and URLs, where a query q and a URL u are connected if a user clicked in the URL u that was an answer for the query q. Then, we generate a query graph by analyzing common URLs among queries. A more frequent approach is to define a similarity measure among queries. However it is more difficult to understand why queries are similar and it can add noise to data already noisy.

This paper proposes two different contributions. First, we propose a method to efficiently cluster very large graphs using clique percolation [3] and *a priori*

* Work done while visiting Yahoo! Research Barcelona.

induced cliques. The quality of the clustering is evaluated by computing an extension of the modularity score [4, 5] for overlapping clusters. Second, we analyze the obtained clusters and extract semantic relations, inside and among clusters, obtaining new information about the nature of the queries and the URLs. To evaluate our method we use a part of the query log of the Yahoo! search engine, with 2.8 million queries with at least one clicked URL and 4.9 million different URLs. For each query, the data includes information on which URLs were clicked and with which frequency. The quality of the inferred semantic relations is evaluated both with a tf-idf (term frequency-inverse document frequency) derived score and against data from Open Directory Project (ODP).

The rest of the paper is organized as follow. In Section 2 we discuss previous work on query similarity and knowledge extraction from queries. In Section 3 we describe the cover graph and its properties. In Section 4 we describe the clustering method and we present the modularity score. In Section 5 we analyze a large graph and we evaluate our approach. We end with some final remarks and future work.

2 Previous Work

Most of the work on query similarity is related to query expansion or query clustering. Here we mention only the main related papers.

Wen *et al* [6] proposed to cluster similar queries to recommend URLs to frequently asked queries of a search engine. They used 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 URL's; and (4) based on the distance of the clicked documents in some pre-defined hierarchy. As the average number of words in queries is small (about two) and the number of clicks in the answer pages is also small [7], notions (1) and (2) generate distance matrices that are very sparse. Notion (4) needs a concept taxonomy and requires the clicked documents to be classified into the taxonomy as well, something that cannot be done in a large scale. Although (3) is also sparse, this sparsity can be diminished by using large query logs. Befferman and Berger [8] also proposed a query clustering technique based on (3) and Zaiane and Strilets [9] used variants of (1) and (3) as well as other simpler features.

Baeza-Yates *et al.* [10, 11] 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 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.

In the work by Chuang *et al.* [12–15] they use 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 [16], but this is not the same as building a taxonomy of the queries, which is what we would call a query taxonomy. Later, Dupret and Mendoza [17] 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. Recently, Baeza-Yates and Tiberi [2] used (3) and a very large query log to define semantic relations such as equivalency 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. Our work can be viewed as a followup to this paper.

3 The Cover Graph

In this section we define the cover graph G that arises naturally from the bipartite graph of queries and URLs, based on the notion of common clicked URLs [18, 19]. 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 *cover graph* $G = (V, E)$ is an undirected and unweighted graph with queries as vertices 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 $\mu(q_1) \cap \mu(q_2) \neq \emptyset$.

For the part of the query log of the Yahoo! search engine analyzed in this paper, the full cover graph is very large with more than 359 million edges (first row of Table 1). Since many URLs are clicked with a very low frequency, this graph is also very noisy. Hence, we will filter the edges in order to remove the noise and reduce the graph size.

Let $\mathcal{W} : \mathcal{Q} \times \mathcal{U} \to [0, 1]$ be a function such that $\mathcal{W}(q, u)$ is the ratio with which the URL u was clicked for the query q. Thus, given a ratio threshold w, the filtered cover graph $G = (V, E)$ is such that $V = \mathcal{Q}$ and $(q_1, q_2) \in E$ if and only if $\{u \in \mu(q_1) \mid \mathcal{W}(q_1, u) \geq w\} \cap \{u \in \mu(q_2) \mid \mathcal{W}(q_2, u) \geq w\} \neq \emptyset$. Note that this type of filtering is different from previous methods used to filter edges [2], where each query has a frequency weight vector associated and edges are weighted according to the cosine similarity. Both approaches are related since high frequency URLs increase the cosine similarity and also increase the confidence we have in the fact that queries joined by a given edge are truly related. However, with our approach, we can easily find cliques in G as we describe in the next section.

In Table 1 we give details about the cover graph for different values of w. For $w = 0.5$ we successfully reduce the size of the cover graph, since the filtered graph has only 36% of the edges in the full graph. Higher values of w reduce even more the size of the graph and the size of the giant connected component. The degree distributions of studied cover graphs follow a power law behavior,

Table 1. Details of the studied cover graphs, where CC is the set of connected components, S is the set of singleton vertices, gC is the giant component and $|V| = 2,822,337$.

w	$\lvert E\rvert$	$2\lvert E\rvert/\lvert V\rvert$	$\lvert E\rvert/\lvert V\rvert \log \lvert V\rvert$	$\lvert CC\rvert/\lvert V\rvert$	$\lvert S\rvert/\lvert V\rvert$	$\lvert gC\rvert/\lvert V\rvert$
0.0	359,881,327	255.023	8.584	0.556	0.499	0.348
0.5	129,915,749	92.062	3.099	0.697	0.620	0.145
0.7	70,487,699	49.949	1.681	0.785	0.718	0.063
0.8	35,324,706	25.032	0.842	0.859	0.806	0.002
1.0	30,695,828	21.752	0.732	0.890	0.847	0.002

as exemplified later in Figure 1 for the case $w = 0.5$ the power law has an exponent of 1.51. We note also that an abrupt change occurs in the size of the giant connected component, as is clear in Table 1 when w changes from 0.7 and 0.8. Another interesting fact is that for $w \geq 0.8$, even for $w = 1.0$, the number of edges does not decrease as much as we may expect. That happens because there are many navigational queries, i.e., queries with just one clicked URL, and many of them refer to the same URL. In the next section we will study the cover graph for $w = 0.5$ in detail and analyze the existing semantic relations among queries.

Given $0 \leq w \leq 1$, the cover graph can be computed efficiently. Let $\bar{N}$ and $\hat{N}$ be the average and the maximum number of URLs covered per query, respectively. Let also $\bar{M}$ and $\hat{M}$ be the average and the maximum number of queries that cover an URL, respectively. First, we sort the URLs for each query in $O(|\mathcal{Q}|\bar{N} \log \hat{N})$ time. Filtering the URLs for each query takes linear time with respect to the number of URLs per query, i.e., takes $O(|\mathcal{Q}|\bar{N})$ time. We also need to compute the list of queries for each URL. This can be done in $O(|\mathcal{Q}|\bar{N})$ time by transposing the list of URLs for each query, a procedure that can be performed while filtering. Finally, we compute the adjacency list for each query by merging the list of queries for each URL in the URL list of the current query. Given the list of URLs for some query and given that the lists of queries and URLs are sorted, we can do the merge using a priority queue where we store the head of each URL query list. Thus, the merging takes $O(\bar{N}\hat{M} \log \hat{N})$ time per query. Given these complexities, the cover graph can be computed in $O(|\mathcal{Q}|\bar{N}\hat{M} \log \hat{N})$ where $\bar{N}$, $\hat{N}$ and $\hat{M}$ are small compared to $|\mathcal{Q}|$ and $|\mathcal{U}|$. For the data we study in this paper and for $w = 0.5$, $\bar{N}$ is 1.370, $\hat{N}$ is 4 and $\hat{M}$ is 7,974.

4 Clique Analysis and Clustering

We are interested in studying overlapping clusters of G in order to identify query relationships and extract semantic information from them. In this paper we specifically study the cliques of G and how they overlap. Previous work has been done on the identification of overlapping clusters from overlapping cliques [20, 3]. In our approach, we use a clustering method similar to clique percolation as introduced by Palla *et al.* [3], where clusters are formed by joining cliques that overlap above a given threshold k.

Since computing maximal cliques is computationally hard [21], we will use *a priori* induced cliques. Let $\mu^{-1} : \mathcal{U} \to 2^{\mathcal{Q}}$ be the "inverse" function of μ that maps each u to its coverable set of queries $\mu^{-1}(u) \subseteq \mathcal{Q}$. Clearly, since every query q in the set $\mu^{-1}(u)$ share at least the URL u, the set $\mu^{-1}(u)$ induces a clique in the graph G. If we are dealing with a filtered version of G, the cliques are induced by the sets $\{q \in \mu^{-1}(u) \mid \mathcal{W}(q,u) \geq w\}$, where w is the frequency threshold.

Given the induced cliques and a threshold k, the clustering method works by merging every clique which overlaps in more than k vertices. Note that each URL has a (filtered) list of queries that are the vertices of the induced clique. Therefore we must intersect each URL list with all other URL lists. The running time is $O(|\mathcal{U}|^2\bar{M})$, where $\bar{M}$ is the average number of queries per URL and, for the studied graph with $w = 0.5$, $\bar{M}$ is 0.784. Given that we are filtering by k, the number of URLs is usually much smaller than $|\mathcal{U}|$.

We compute an extension of the modularity measure for overlapping clusters to evaluate the quality of the clustering. Given a non-overlapping clustering $\mathcal{C} = \{C_1, ..., C_n\}$ of G, the *modularity* Q is defined [4, 5] as

$$Q = \frac{1}{2|E|} \sum_{p,q \in V} \left[A_{pq} - \frac{d_p d_q}{2|E|} \right] \delta(c_p, c_q), \tag{1}$$

where $1 \leq c_i \leq n$ denotes the cluster where vertex i belongs, A is the adjacency matrix of G, d_i is the degree of vertex i and the δ-function is such that $\delta(i,j) = 1$ if $i = j$ and $\delta(i,j) = 0$ otherwise. Note that the above sum runs over all possible pairs of vertices. Therefore each edge is summed twice. If we split the sum in two terms,

$$\frac{1}{2|E|} \sum_{p,q \in V} A_{pq} \delta(c_p, c_q) \qquad \text{and} \qquad \frac{1}{2|E|} \sum_{p,q \in V} \frac{d_p d_q}{2|E|} \delta(c_p, c_q), \tag{2}$$

the first term is the fraction of edges that fall within the clusters and the second term is the expected fraction of edges within the clusters, if the edges were randomly distributed while respecting the vertices degrees. In particular, if the edges were randomly placed as mentioned, $d_p d_q / |E|$ is the probability of the existence of an edge between vertices p and q.

Thus, modularity measures the fraction of edges that connect vertices in the same component minus the expected value of the same quantity in a graph with the same components but random connections between the vertices [4]. Values near 1, the maximum value of Q, indicate strong community structure. Typically, values for graphs underlying common networks with known community structure are in the range from 0.3 to 0.7.

However, if $\mathcal{C}$ has overlapping clusters, the measure needs refinement because it was designed for non-overlapping clusters. In this work, we extend the definition of modularity by weighting edge contributions with the cluster overlap centrality of the vertices. The *cluster overlap centrality* ν of a vertex $q \in V$ is the number of clusters that contain q. It is a generalization of the clique overlap centrality proposed by Everett and Borgatti [20]. Therefore, we extend the

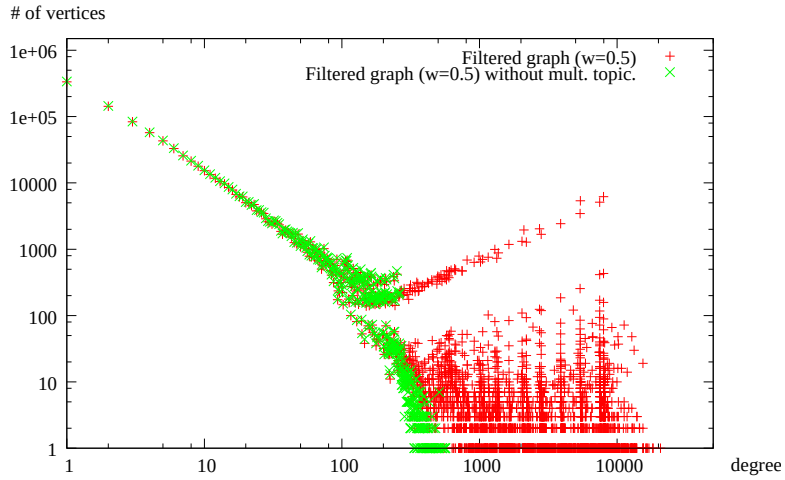

Fig. 1. Cover graph degree distribution before and after multitopical URL removal.

modularity definition as

$$Q = \frac{1}{2|E|} \sum_{p,q \in V} \left[A_{pq} - \frac{d_p d_q}{2|E|} \right] \frac{\delta(c_p, c_q)}{\nu_p \nu_q}. \tag{3}$$

This definition of modularity is a particular case of a more general extension proposed recently by Nicosia *et al.* [22]. Notice that this definition is equivalent to equation 1 when the clustering does not contain overlapping clusters.

5 Experimental Evaluation

We studied several cover graphs for different values of w. In this section we present the experiments with $w = 0.5$, as they exemplify our approach and provide interesting insights with respect to semantic relations among queries. We built the filtered cover graph as described in Section 3. Then, we applied the clustering method described in Section 4 for different values of k and we computed the modularity score to evaluate the quality of clustering. The clustering obtained with $k = 266$ (Figure 2) has the highest modularity score, $Q = 0.667$, a value that indicates the existence of well defined clusters. We found that these clusters are induced by 67 URLs with a high number of queries, the biggest one having $7,974$ queries. It is interesting to note that these 67 URLs are all multitopical web pages.

Since multitopical web pages usually introduce noise as they relate very different queries, we generate the graph without the above 67 URLs. We note that a previous method was proposed by Baeza-Yates and Tiberi [2] to remove multitopical URLs. It is interesting that, although different, both approaches reduce the number of edges to similar values. The filtered graph has less than 14

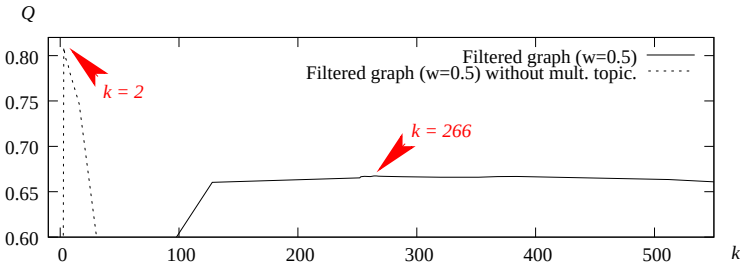

Fig. 2. Modularity score for different values of k.

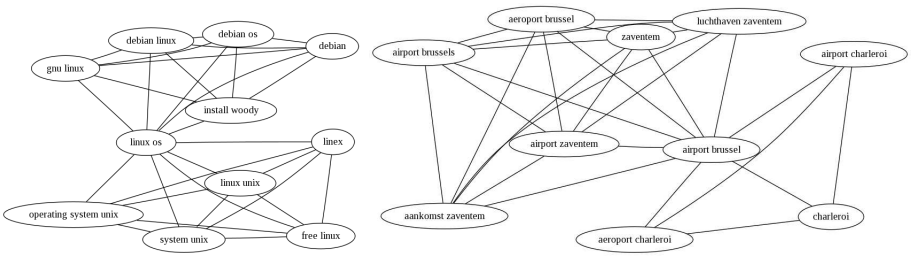

Fig. 3. Cluster intersections for queries "linux os" and "airport brussel".

million edges and Figure 1 depicts the degree distribution before and after the URL removal. Both distributions follow power laws with coefficients of 1.51 and 1.70, respectively. We have applied the clustering method to this new graph and the results are rather interesting. The clustering with highest score, $Q = 0.809$, was obtained with $k = 2$ (Figure 2). This implies that the original graph structure was almost defined by a few large cliques and, after the multitopical URL removal, the result is a graph with a structure defined by cliques of size 3 (percolation of size 2 results from the intersection of cliques with more than 3 vertices). This clustering has 116,044 clusters and covers 25% of the original queries. Note that 62% of the original queries are isolated vertices and, therefore, the clustering covers 66% of non singleton queries. More interesting, this clustering covers 79% of the queries in the giant component for the filtered graph with $w = 0.5$.

After obtaining this clustering, we examined the relations within and among clusters. Figure 3 contains two examples of the observed clusters and cluster intersections. In the first example the query "linux os" appears in two contexts: one related to the Debian distribution and the other to Unix operating systems. In the second example we see that the technique can find related queries even in different languages (English and Flemish). By analyzing the clusters we were able to find different contexts for given queries. We were also able to find equivalences between queries, e.g. "ge" and "general electrics". However, it would be interesting to have some method to classify queries and terms in the clusters and extract relevant semantic information as in [2].

Table 2. Cluster tf-idf average, maximum and terms with tf-idf score ≥ 0.5.

C.id	$\overline{\text{tf-idf}}$	Max	$\lvert C\rvert$	Terms
1	5.658	5.831	3	synaptics, touchpad
2	2.374	2.569	4	erricson, ericcson, ericson, ericsson
3	2.336	4.998	4	charleroi, airport, brussel, aeroport
4	1.612	5.282	6	velux, skylite, www.velux.com, skylight, windows
5	1.423	2.881	6	debian, linux, os, gnu, woody, install
6	1.369	3.588	7	zaventem, brussel, airport, luchthaven, aankomst
7	1.329	2.359	3	linux, os, xp, servers
8	1.327	2.991	3	hfs, whfs, 99.1, whfstival, 105.7, dc
9	1.099	2.691	5	slackware, linux, kernel, slackware.com, 2.6, 9.0
10	1.024	2.143	6	unix, linux, linex, system, operating, os
11	1.019	2.778	5	longhorn, windows, screenshots
12	0.899	2.359	3	linux, wine, windows
13	0.848	2.105	4	cooking, wine, recipes, good, food
14	0.757	1.616	9	longhorn, steakhouse, horn, long, steak
15	0.662	1.262	5	spirits, liquor, pa, wine, pawineandspirits.com
16	0.591	1.727	7	baseball, longhorn, texas
17	0.585	1.054	7	union, credit, federal, teachers, teacher, tfcu
18	0.464	1.994	8	scorpios, scorpio, meaning, sign
19	0.091	0.970	51	windows, xp
20	0.072	0.838	76	delta, flights
21	0.069	0.594	104	yahooligans, games, kids

For this, we have processed the clusters as follows. First we enumerated the terms for each cluster and then we computed the tf-idf score for each term. We ranked the clusters by tf-idf average and, for each cluster, we also ranked the terms by tf-idf score. In Table 2 we provide tf-idf values for some clusters.

The analysis of the cluster ranking provides interesting information with respect to the nature of the queries. Clusters with navigational queries appear at the top of the ranking. Usually these queries have few terms, although very informative. The top ten clusters in Table 2 are examples of clusters with navigational queries, e.g., "synaptics touchpad" that is one of the 3 queries in the first cluster or "www.velux.com" that appears as term and as query in the second cluster. But we can generalize this analysis and verify that when a user knows what to search the queries appear in a higher ranked cluster. An example is the query "install woody" in Figure 3 and also in the fifth cluster of Table 2. Although this query does not refer Debian or Linux, it clearly refers to the installation of the Debian Linux distribution named Woody. Note that navigational queries are also an example of queries where the user knows what he wants.

The importance of a given term within a cluster can be inferred from the ranking of terms for each cluster. In Table 2 the terms are sorted from left to right, with the maximum tf-idf being the score of the first term. Thus, those terms are the most relevant in each cluster and can be seen as a description of the cluster.

Table 3. Distribution of queries with respect to ODP score.

σ	≥ 0.25	≥ 0.33	≥ 0.50	≥ 0.66	≥ 0.75	$= 1.00$
%	90.5	85.6	69.7	56.5	49.6	34.7

By studying the overlap of terms among clusters, we can improve the semantic information obtained from cluster overlapping and we are able to identify context and polysemy. In Table 2 we can identify clusters where a given term appears in the same context and others where the context is different. For example, the term "windows" appears in cluster three with the usual meaning and it appears in the other clusters as the Microsoft operating system. The term "wine" is an example of polysemy. It appears in cluster eleven as the Windows api for Unix-like operating systems and in the other clusters as a beverage.

In spite of the above interesting results, evaluating the quality of semantic relations is difficult. We would like to know if, from the user perspective, the queries in each cluster are truly related and contain valuable semantic information. Thus, we systematically evaluated a sample of clusters against data from ODP[3]. The Open Directory is a comprehensive human-edited directory of the Web and is constructed and maintained by a community of volunteer editors.

We randomly selected $1,000$ clusters among the $116,044$ clusters obtained. Then, we submitted each query in these clusters to the ODP and we obtained a set of categories matches in form of paths between directories. Note that these categories are ordered by relevance. For instance, the query "airport brussel" would provide the category "Regional: Europe: Belgium: Transportation" as the most relevant. To measure the similarity between queries, we measure the similarity between categories [2]. Thus, given two queries, we select the two most similar categories d_1 and d_2 as provided by ODP. The similarity score σ is defined as follows

$$\sigma(d_1, d_2) = \frac{|\pi(d_1, d_2)|}{\max\{|d_1|, |d_2|\}}, \tag{4}$$

where $\pi(d_1, d_2)$ is the longest common prefix and $|\cdot|$ is the directory path length. The ODP similarity for two queries is the value of equation 4 for the most similar categories between them, i.e., the maximum among all possible pairs of categories for those queries. For each cluster we computed the ODP score as average of ODP similarity over all pairs of queries for which ODP provides at least one category. Note that, for pairs with queries with 0 categories, equation 4 is undefined.

Although the similarity score has values from near 0 to 1, we get an average of 0.7 among the $1,000$ clusters and 35% of the clusters have a score of 1.0. We provide the ODP score distribution in Table 3. Thus, we can infer that the clusters found reflect relevant semantic relations. We also verify that small clusters usually have higher values, which is to be expected, given the focus of their queries as discussed above. But, as depicted in Figure 4, the cluster ranking

[3] http://www.dmoz.org/

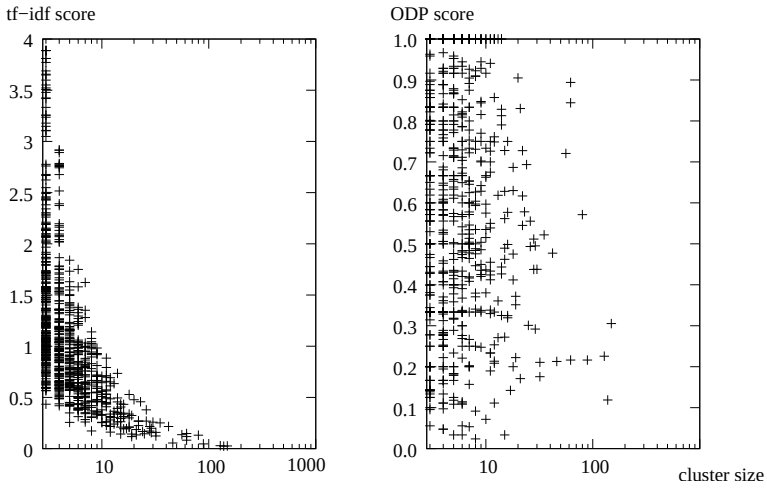

Fig. 4. Tf-idf and ODP scores for different cluster sizes.

is much different from the tf-idf ranking and the score decreases much slower. It is also important to note that there are small clusters with very low tf-idf and ODP score. This happens because these clusters usually have few queries with many terms and usually these are more specific queries.

It is also interesting that, with the ODP score we were able to evaluate large clusters and distinguish which ones have more relevant information, as shown in Figure 4. Note that the score is not correlated to the cluster size and is better than tf-idf similarity. With the tf-idf score we were unable to perform this discrimination since larger clusters have smaller scores. In fact, if we also had considered the best ranked terms by tf-idf in each cluster, we would have improved the ranking of some larger clusters (although not the ODP score).

In Table 4 we provide the obtained scores for the clusters given in Table 2. We also provide the ODP category obtained with the two or three most significant terms for each cluster given in Table 2. Note that this is done by combining the tf-idf score and the ODP score, thus providing an interesting categorization of clusters.

6 Final Remarks

The efficient graph mining techniques proposed in this paper can be applied to very large graphs obtained from query logs, and the results show that these techniques are effective at obtaining semantic relations between queries. In the concrete case studied, the semantic relations discovered are useful and have provided interesting insights about implicit knowledge contained in queries submitted to a search engine.

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.

Table 4. Cluster ODP scores and relevant categories.

C.id	Score	$\|C\|$	ODP category (most significant)
1	1.000	3	Computers: Software: Operating Systems: Linux: Hardware ...
3	1.000	4	Regional: Europe: Belgium: Transportation
11	1.000	5	Computers: Software: Operating Systems: Microsoft Windows: ...
9	1.000	5	Computers: Software: Operating Systems: Linux: Distributions: ...
4	1.000	6	Business: Construction and Maintenance: Materials and Supplies: ...
17	0.976	7	Business: Financial Services: Banking Services: Credit Unions
5	0.889	6	Computers: Software: Operating Systems: Linux: Projects: ...
13	0.722	4	Home: Cooking
16	0.722	7	Sports: Baseball: College and University: NJCAA
10	0.629	6	Computers: Software: Operating Systems: Linux
19	0.621	51	Computers: Software: Operating Systems: Microsoft Windows: ...
12	0.600	3	Computers: Emulators: Intel x86 Architecture: DOS and Windows
15	0.535	5	Recreation: Food: Drink: Liquor
21	0.477	104	Kids and Teens: Games: Online: Collections
7	0.467	3	Computers: Software: Shareware: Networking
14	0.443	9	Business: Hospitality: Restaurant Chains: Steakhouses
6	0.400	7	Regional: Europe: Belgium: Regions: Brussels: Travel ...
20	0.312	76	Recreation: Aviation: Experience Flights
8	0.056	3	Arts: Radio: Regional: North America: United States: Maryland
2	0.048	4	Regional: ...
18	0.024	8	Society: Religion and Spirituality: Divination: Astrology: ...

Further analysis of the structure of the graph will also be important to unveil more relations. One possibility is to extend the similarity analysis to all the clusters and use edges with $w < 0.5$ to find weaker relations among the clusters to infer a possible taxonomy. The efficiency of the proposed methods makes them applicable to much larger graphs, thus making them useful for the extraction of semantic relations from less frequent queries.

References

1. Surowiecki, J.: The Wisdom of Crowds: Why the Many are Smarter than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations. Little and Brown (2004)
2. Baeza-Yates, R.A., Tiberi, A.: Extracting semantic relations from query logs. In: Proc. of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2007) 76–85
3. Palla, G., Dernyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435** (2005)
4. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review Letters **69** (2004)
5. Newman, M.E.J.: Modularity and community structure in networks. In: Proc. of the National Academy of Sciences. Volume 103. (2006)

6. Wen, J., Mie, J., Zhang, H.: Clustering user queries of a search engine. In: Proc. of the 10th International World Wide Web Conference, W3C (2001)
7. Baeza-Yates, R.: Applications of web query mining. In: European Conference on Information Retrieval (ECIR'05), D. Losada, J. Fernández-Luna (editors). Volume 3408 of Lecture Notes in Computer Science., Springer (2005) 7–22
8. Beeferman, D., Berger, A.: Agglomerative clustering of a search engine query log. In: Proc. of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (1999)
9. Zaiane, O.R., Strilets, A.: Finding similar queries to satisfy searches based on query traces. In: Proc. of the International Workshop on Efficient Web-Based Information Systems (EWIS). (2002)
10. Baeza-Yates, R., Hurtado, C., Mendoza, M.: Query clustering for boosting web page ranking. In: Advances in Web Intelligence, AWIC 2004. Volume 3034 of Lecture Notes in Computer Science., Springer (2004) 164–175
11. Baeza-Yates, R., Hurtado, C., Mendoza, M.: Query recommendation using query logs in a search engine. In: EDBT Workshops. Volume 3268 of Lecture Notes in Computer Science., Springer (2004) 588–596
12. Chuang, S.L., Chien., L.F.: Automatic query taxonomy generation for information retrieval applications. Online Information Review **27**(5) (2003)
13. Chuang, S.L., Chien, L.F.: Enriching web taxonomies through subject categorization of query terms from search engine logs. Decision Support System **30**(1) (2003)
14. Chuang, S.L., Chien, L.F.: Towards automatic generation of query taxonomy: A hierarchical query clustering approach. In: Proc. of the 2002 IEEE International Conference on Data Mining, IEEE (2002)
15. Pu, H.T., Chuang, S.L., Yang, C.: Subject categorization of query terms for exploring web users' search interests. Journal of the American Society for Information Science and Technology (JASIST) **53**(8) (2002)
16. Cheng, P.J., Tsai, C.H., Hung, C.M., Chien, L.F.: Query taxonomy generation for web search (poster). In: CIKM. (2006)
17. Dupret, G., Mendoza, M.: Automatic query recommendation using click-through data. In: Symposium on Profesional Practice in Artificial Intelligence, 19th IFIP World Computer Congress, WCC 2006, Springer (2006)
18. Wen, J., Mie, J., Zhang, H.: Clustering user queries of a search engine. In: Proc. of the 10th international conference on World Wide Web, ACM (2001) 162–168
19. Baeza-Yates, R.A.: Graphs from search engine queries. In: SOFSEM. Volume 4362 of Lecture Notes in Computer Science., Springer (2007) 1–8
20. Everett, M.G., Borgatti, S.P.: Analyzing clique overlap. Connections **21** (1998)
21. Karp, R.: Reducibility among combinatorial problems. In: Proc. of a Symposium on the Complexity of Computer Computations, Plenum Press (1972)
22. Nicosia, V., Mangioni, G., Carchiolo, V., Malgeri, M.: Extending modularity definition for directed graphs with overlapping communities (2008) arXiv.org:0801.1647.