The anatomy of a large query graph

IOP PUBLISHING JOURNAL OF PHYSICS A: MATHEMATICAL AND THEORETICAL

J. Phys. A: Math. Theor. **41** (2008) 224002 (13pp) doi:10.1088/1751-8113/41/22/224002

The anatomy of a large query graph

Ricardo Baeza-Yates[1,2] **and Alessandro Tiberi**[2]

[1] Yahoo! Research, Barcelona, Spain and Santiago, Chile
[2] University of Rome 'La Sapienza', Italy

E-mail: rbaeza@acm.org and tiberi@di.uniroma1.it

Received 13 November 2007, in final form 10 March 2008
Published 21 May 2008
Online at stacks.iop.org/JPhysA/41/224002

Abstract
In this paper, we analyze the structure of a graph derived from query logs. Previous query log analysis was mostly done with just the queries and not the actions that followed after them, while the studied graph is induced by the actions of the users. A few exceptions do consider the clicks after a query, but mainly in a small or filtered query log. In this paper, we analyze a graph produced by more than twenty million queries, showing that it is less sparse than previous results suggested. We show that our graphs strongly resemble scale-free networks and that almost all the measures of these graphs are well approximated by power laws.

PACS numbers: 89.75.−k, 89.75.Da, 89.75.Fb

1. Introduction

Queries submitted to search engines convey implicit knowledge if we assume that most of the time user actions are meaningful. Hence, the challenge is to extract interesting relations from very large query logs. One natural starting point is to infer a graph from the queries. Another possibility, most frequent in previous research, is to define a similarity (or distance) function between queries. This also implies a graph based on this function. One drawback of defining a function is that it is more difficult to understand why two queries are similar. Moreover we can possibly add artificial artifacts that add noise to data that is already noisy.

In this paper, we use a graph based on both approaches just described by mixing a natural representation for the graph [4] with a simple distance among queries for the edge weights [6] and we analyze large graphs that are generated by such a distance. The graphs are less sparse than in previous work, perhaps because of the previous lack of the availability of large data sets. We analyze several characteristics of these graphs, providing information not only about how people query but also how they behave after a query and the content distribution of what they look at. Our analysis shows that these graphs are complex objects with a non-trivial topological structure. We manipulate the graphs in several ways, and show that their underlying structure tends to be preserved. Our study of the structure of these objects is motivated by a rather pragmatical reason. In fact, to effectively exploit the semantic information that is hidden in the graphs the noise must be reduced. A proper understanding of the graphs' structure is crucial to achieve this goal.

In section 2, we discuss previous work in query similarity. In section 3, we describe a natural vector model to define similar queries [4, 6]. In section 4, we analyze a particular graph of several million queries, including measures used in social network analysis. We end with some conclusions and future work.

2. Previous work

Most of the work on query similarity is related to query expansion or query clustering. One early technique proposed by Raghavan and Sever [11] attempts to measure query similarity by determining differences in the ordering of documents retrieved in the answers. As this technique requires a total ordering in the document collection, the comparison of two rankings would require superlinear time. Considering the current size of the Web, this algorithm does not appear to be feasible.

Later, Fitzpatrick and Dent [9], measured query similarity using the normalized set intersection of the top 200 documents in the answers for the queries. Again, this is not meaningful in the Web as the intersection for semantically similar queries that use different synonyms can and will be very small.

Wen *et al* [14] 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. Beeferman and Berger [7] also proposed a query clustering technique based on a distance notion (3). As the average number of words in queries is small (about two) and the number of clicks in the answer pages is also small [1], notions (1)–(3) are difficult to deal with in practice, because the distance matrices between queries generated by them are very sparse. This sparsity can be diminished by using larger query logs, which are not available to most researchers. 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.

Fonseca *et al* [10] present a method to discover related queries based on association rules. Here queries represent items in traditional association rules. The query log is viewed as a set of transactions, where each transaction represents a *session* in which a single user submits a sequence of related queries in a time interval. The method shows good results, however two problems arise. First, it is difficult to determine sessions of successive queries belonging to the same search process; and on the other hand, the most interesting related queries, those submitted by different users, cannot be discovered. This is because the support of a rule increases only if its queries appear in the same query session, and thus they must be submitted by the same user.

Zaiane and Strilets [15] present seven different notions of query similarity. Three of them are mild variations of notions (1) and (3). The remainder notions consider the content and title of the URL's in the result of a query. Their approach is intended for a meta-search engine and thus none of their similarity measures consider user preferences in the form of clicks stored in query logs.

Baeza-Yates *et al* [2, 3, 5] 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, like the cosine function. This notion of query similarity has several advantages. First, it is simple and easy to compute. On the other hand, it allows us to relate queries that happen to be worded differently but stem from the same topic. Therefore, semantic relationships among queries are captured. We follow this approach in this paper.

Later, Sahami [12] used a query similarity based on the snippets of the answers to the queries (e.g. the first page of results). For that they treat each snippet as a query to the search engine in order to find a certain number of documents that contain the terms in the original snippets. Then, they use these returned documents to create a context vector for the original snippet. However, the main drawback is that this approach does not consider the feedback of the users (i.e. clicked pages) and hence, for example, it is affected by Web spam.

3. The cover graph

In this section, we define a graph that naturally comes from user actions after a query. It is based on notion (3) explained in the previous section and was formally introduced by Baeza-Yates [4]. Before continuing, we need to define the main concepts used in the following:

- Query instance: query (set of words or phrase) plus zero or more clicks related to that query. Formally:

$$QI = (q, u^*), \qquad \text{where} \quad q = \{\text{words or phrase}\}$$

  being q the query, and u a clicked URL.
- URL cover: set of all URLs clicked by a query instance. That is:

$$UC_p = \bigcup_{QI_q = p} QI_{c(u)}.$$

To reduce the number of different queries, they were normalized, by transforming all capital letters to lowercase, deleting non-alphabetical or non-numerical characters and sorting the terms lexicographically (because many people use the same terms in a different order). That is, every query was transformed to a bag of words.

We are interested in the aggregation of equal queries (e.g. same set of words or same phrase) independently of other attributes of the query. Hence we will use as *URL cover* the union of all covers for the equivalent queries, and we will use *query* to denote the set of all syntactically equivalent normalized queries. We now define a vectorial representation for each query.

Queries are represented as points in a high dimensional space, where each dimension corresponds to a unique URL u. That is, a query is based on all the different URLs in the URL cover. Given a query q, denote its representation with $\bar{q}$. Each component of the vector $\bar{q}$ is assigned a weight equal to the frequency with which the corresponding URL u has been clicked for that query q. We now define a graph based on this vectorial representation.

Each query is a node of the graph. Two nodes (queries) are connected by an edge iff they share at least one URL u (that is, their vectorial representations have one component that is positive in both queries). Hence we obtain a non-directed graph.

Edges are weighted according to the cosine similarity of the queries they connect. Hence, if $e = \{q, q'\}$ and the URL space has D dimensions (total number of different URLs), the weight of e is given by [6]:

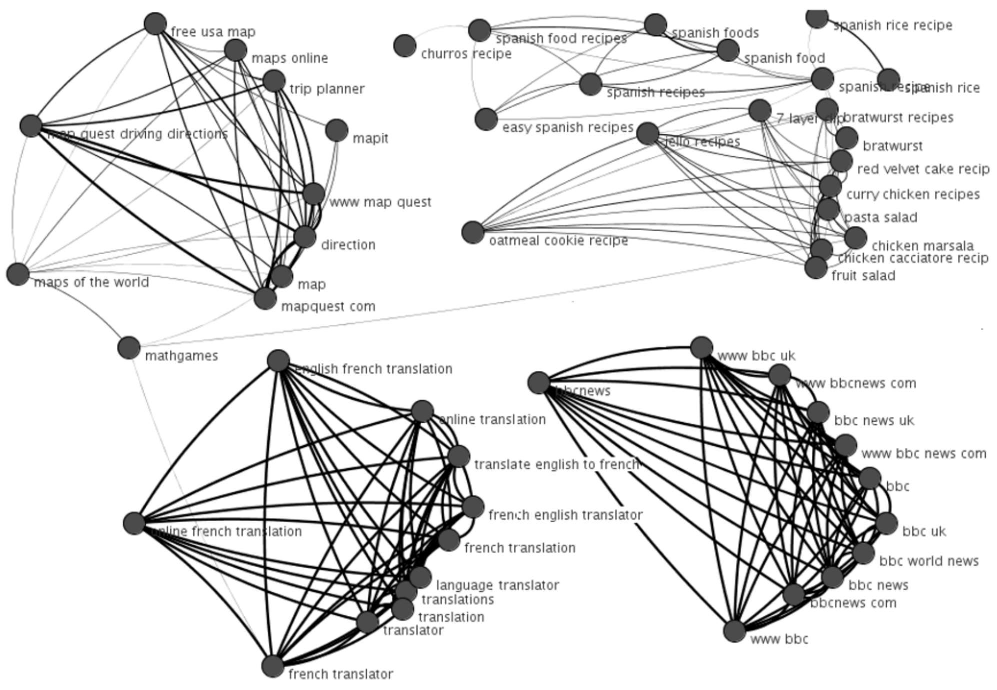

Figure 1. A sub-graph example considering all possible covers, that has two connected components.

$$W(e) = \frac{\bar{q} \cdot \bar{q'}}{|\bar{q}||\bar{q'}|} = \frac{\sum_{i \leqslant D} q(i) \cdot q'(i)}{\sqrt{\sum_{i \leqslant D} q(i)^2} \cdot \sqrt{\sum_{i \leqslant D} q'(i)^2}}.$$

We define the *weighted degree* of a node, the sum of the weights of all its edges divided by its degree $\delta(q)$. That is, the weighted degree $\delta_W(q)$ of a node q is:

$$\delta_W(q) = \frac{1}{\delta(q)} \sum_{q \in e} W(e).$$

This graph can be computed relatively fast. The edges can be computed by sorting the URLs clicked by a query. The weights of the edges can be computed in linear time in the worst case, much faster on average as queries do not share many URLs. Let M be the maximal number of URLs between two queries. Hence the graph can be computed in $O(\max\{ME, n \log n\})$ time, where E is the number of edges in the graph and n is the number of nodes.

In figure 1, we show a small part of a cover graph, that includes a cluster about the BBC plus three other clusters (maps, translation and food recipes), weakly connected with the query *mathgames*. This implies that there are pages that cover math games and each one of the topics on the other clusters. Note that edges with larger weights are darker.

4. Analysis of a large graph

We have used several query logs and the results are similar, so we present only the analysis of a log piece of 2005 coming from the Yahoo! search engine. For building this graph we have used all the queries in the log piece with at least one click. They were almost 21 million

Table 1. Some figures about the studied graphs.

	Full graph	Filtered graph (Clicks > 10)	Filtered graph (Edge weight > 0.5)
Nodes:	2841 520	100 057	100 057
Isolated nodes:	1412 208	25 437	74 532
Edges:	361 029 245	10 370 388	60 945
Unique URL:	4927 978	973 187	973 187

Table 2. Parameters of the fitted power laws.

Power Laws: $A \cdot x^B$	A	B
Query frequency		
Overall	10 191 819	−2.38
With clicks	–	–
No clicks	2521 970	−3.06
Query clicks		
Full graph	10 810 986	−2.39
Filtered	5541 161	−2.53
Node degree		
Full graph	548 004	−1.43
Filtered (clicks)	31 819	−1.37
Filtered (edge)	31 819	−2.05
Connected components		
Full graph	1765 795	−3.66
Filtered (clicks)	29 035	−3.94
Filtered (edge)	50 838	−3.13

queries that were reduced to 2.8 million different queries after normalization. This graph is very large, and we also study smaller graphs derived from it by filtering both its edges and its nodes. Concerning the nodes, we have filtered the queries with few clicks, while for the edges we have used a weight criterion. The weight and click criteria applied for filtering the graph are intuitively justified by an attempt to reduce the noise: when removing low weight edges we are also reducing the noise, since those edges represent weak relations between queries. Similarly, queries with few clicks tend to be more noisy since we have less information at disposal. Other ways of filtering the graph are of course possible, e.g. query frequency, URL frequency and are currently being investigated. Table 1 gives the main characteristics of the different versions of the graph.

For all the generated graphs, we have studied different parameters. In the following each of them is discussed, and the corresponding plots are presented. Often, the observed behavior clearly follows a power law in the main part of the distribution. Some readers may argue that the fit is not good at the extremes of the distribution, but we are interested in a simple model for each distribution that fits the majority of the data. When this is the case, the associated law fitted to the data using least squares is also plotted. Moreover, in table 2 the interested reader can find the coefficients of all the approximated distribution.

4.1. Basic characteristics

Figure 2 shows the frequency of the queries in the whole log. Each entry in the query log is counted as a distinct query occurrence. Three curves are shown, one counts the occurrences of

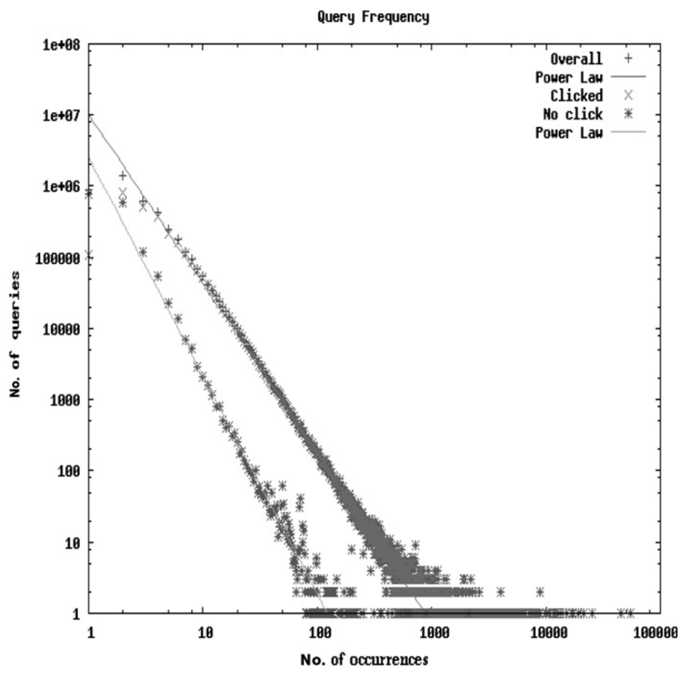

Figure 2. Queries frequency for the whole set of queries.

all the queries, one counts the occurrences of the queries for which at least one result has been clicked, and the last one the occurrences of the queries without any click. All the curves exhibit a power law behavior (the coefficients can be found in table 2). Clicked queries essentially show the same behavior (and actually they follow the same power law) of what is observed in the whole log. On the other hand, queries without clicks appear to be more biased.

Figure 3 shows the query click distribution. Each point represents the *fraction* of queries (y-axis) with a given number of clicks (y-axis). Again we can see that for the full graph as well as for the filtered one, there is a power law at work.

Figure 4 shows the clicks distribution from the URLs point of view. Each point represents the number of URLs with a given number of clicks. Not very surprisingly, also this distribution can be approximated by a power law.

In left side of figure 5 the query click distribution against the query frequency is shown. That is, each point represents the number of occurrences and the number of clicks for a given query. Both axis are normalized by dividing either by the total occurrences (x-axis) or by the total number of clicks (y-axis). The bottom left part of the graph clearly shows much more variance than the rest. That is because less frequent queries have more click variance while very frequent queries tend to have a bit less than one click on average. The right side of figure 5 shows the node degree distribution (number of queries related by clicks) and query frequency.

In figure 6 the nodes weight (left) and edges weight (right) distribution are plotted. In both graphs, the y-axis is normalized, showing the fraction of nodes (edges) with a given weight. The weight of a node is just its weighted degree, as previously defined. The curve for the graph that has been filtered according to the edges weight is obviously truncated at the threshold we have used. The same applies for the other graph. In both cases, we see that the filtered graph has a much more defined distribution.

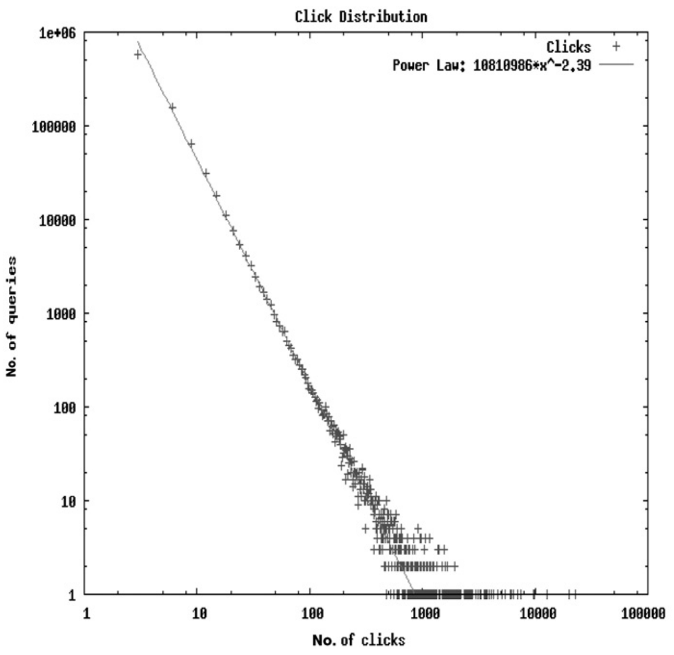

Figure 3. Query click distribution.

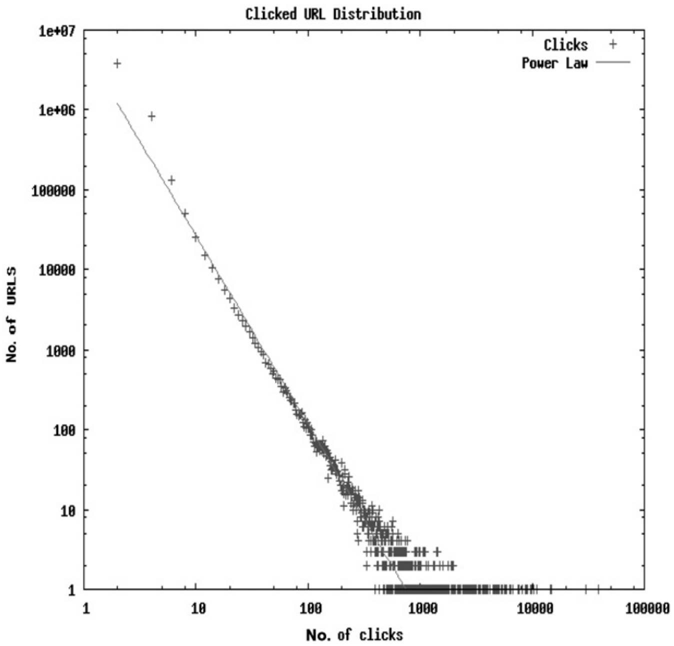

Figure 4. Clicked URL distribution.

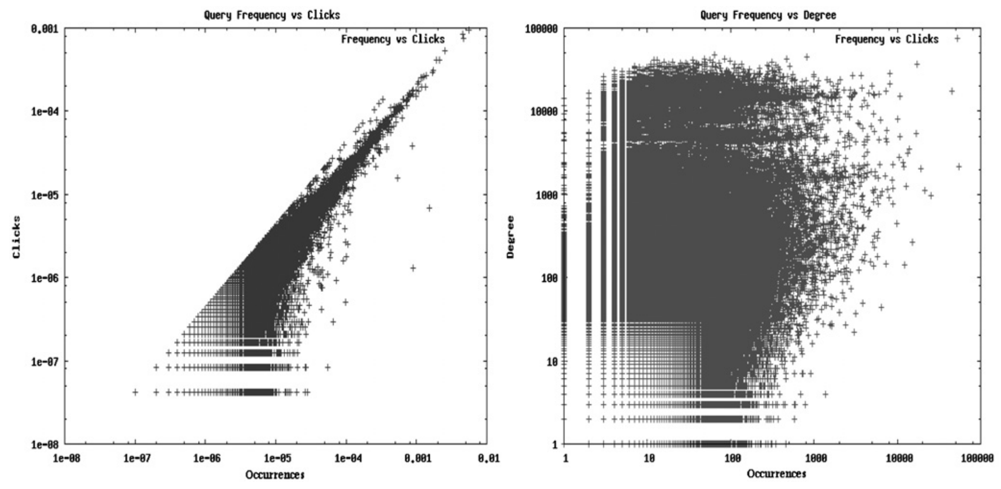

Figure 5. Queries frequency versus query clicks and node degree.

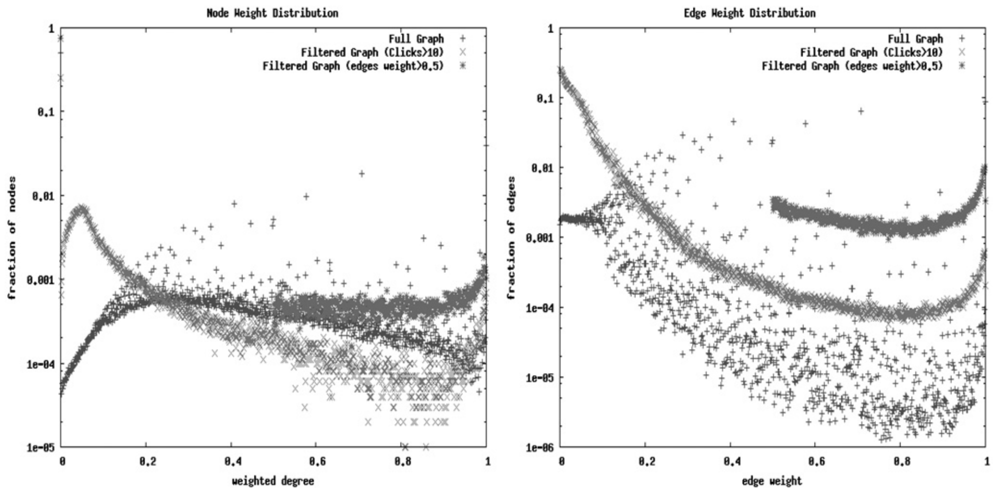

Figure 6. Node and Edge weight distribution (as fractions).

4.2. Connectivity

Figure 7 is probably one of the most interesting. On the left the node degree distribution (normalized so that each point shows the number of nodes with a given degree) is shown, while on the right one each node degree is plotted against the number of occurrences of the corresponding query. As we can see from the graph, the node degree distribution follows a power law, which is a typical behavior observed in scale-free networks. Moreover, this behavior is observed in all the graphs we studied, regardless of their size and of the criterion applied for filtering. Hence this property seems to remain constant even if you filter the graph, both on the nodes and on the edges. This is another strong evidence of the resemblance between the cover graph we have generated and free-scale and autosimilar networks. We also underline that the curve for the graph filtered according to the edge weight, shows even more clearly such a property. Thus, low weight edges (weak semantic relation between queries)

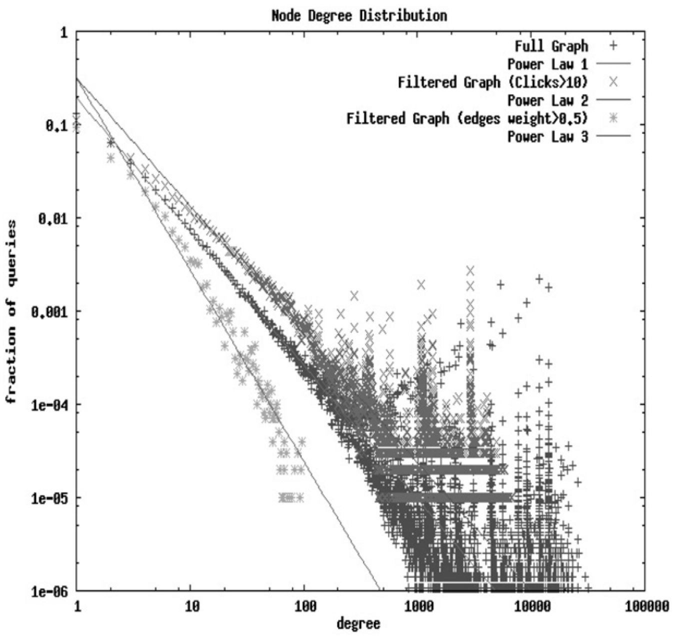

Figure 7. Node degree distribution.

are responsible of the outliers and probably many of them are just noise that can be easily removed.

The plot in figure 8 shows the connected components distribution. Each point represents the number of components of a given size (expressed as a fraction of the total number of nodes). Also in this case, we see that for all three graphs the distribution follows a power law. Interestingly enough, all graph exhibit a connected component that is a good fraction of the whole graph, essentially a giant component. This behavior resembles what is observed in the web graph. Moreover, the ratio between the size of the largest connected component and the number of nodes, varies according to which kind of filtering we apply. In particular, we can observe that this ratio increases when the click filter is used, while it gets smaller when the edge weight filter is used.

4.3. Dynamics

The two plots in figure 9 show how the number of edges varies respectively when increasing the number of nodes and the threshold according to which edges are trimmed. The left plot, shows that edges grow quadratically in the number of nodes, although with a small constant (1/21 000 in the example in the graph). This fact is more evidence that our graphs are much less sparse than that previously reported for similar objects. This also shows that computing these graphs will take time proportional to the number of edges which seems to be quadratic in practice. The right plot shows how the number of edges decreases when applying larger weight thresholds. With a relatively low threshold (0.3), the number of edges is already halved.

Next we explore varying the size of the graph with the results reported in figure 10. Each plot shows the behavior of a different measure when the size of the graph varies. Smaller graphs were generated from the original graph by randomly selecting subsets of nodes of given sizes. Then the subgraphs induced by these random sets were considered. A common trend

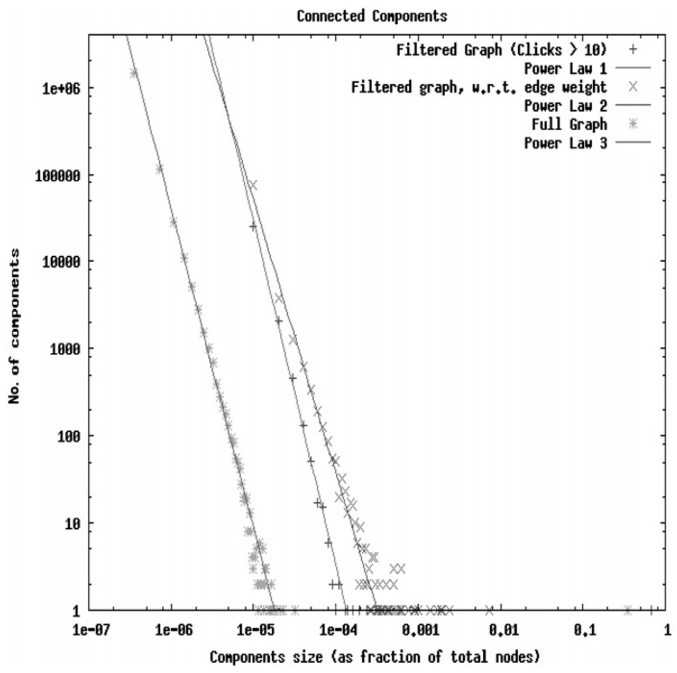

Figure 8. Connected components.

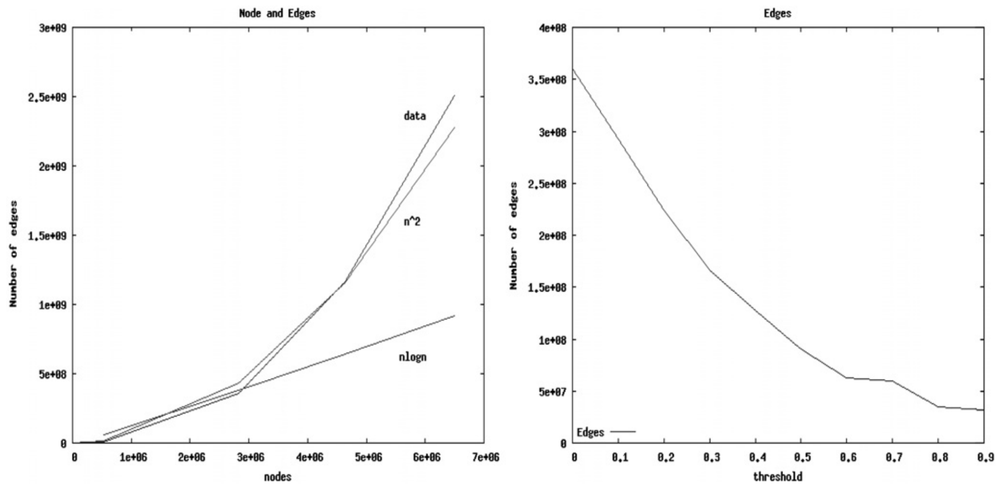

Figure 9. These two plots show the number of edges when increasing the number of nodes (left) and the threshold used to discard edges (right).

can be observed in all plots: as the size of the graphs changes, qualitatively the same behavior is observed in all graphs.

The plot at the left shows the node degree. Each point corresponds to the degree (y-axis) of a given node, with the nodes being ranked according to their degree. Clearly in each graph, no matter its size, the same behavior is observed.

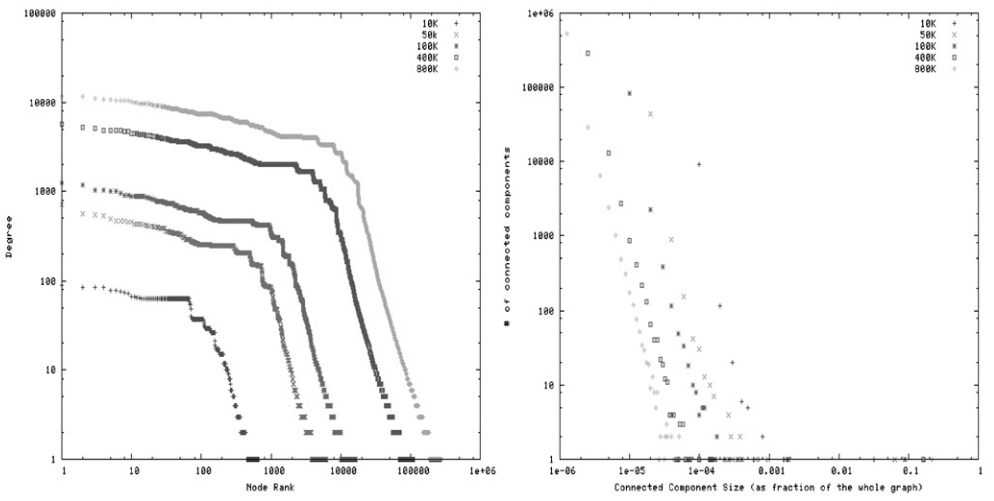

Figure 10. Two different measures for graphs of increasing size: node degree (left) and size of the connected components (right).

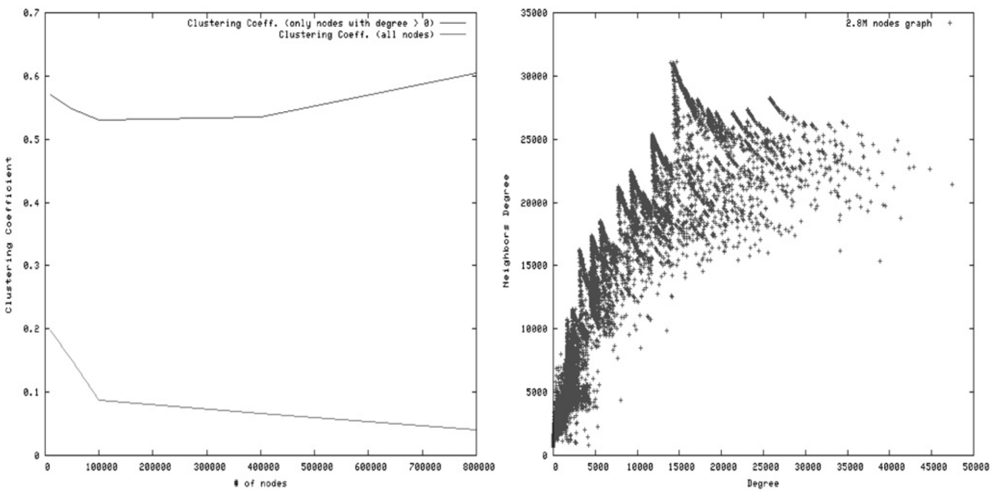

Figure 11. Clustering coefficient with respect to graph size (left). The distribution of the 'neighbors' degree' (right). From the plot we can note that high degree nodes tend to link nodes with similar degree.

The right plot shows, for each graph, the connected component distribution. The x-axis is normalized to show components' size as a fraction of total nodes in each graph, while the y-axis shows the number of components with a given size. Again, qualitatively a similar behavior is observed for all subgraphs.

In the left plot of figure 11 we show the clustering coefficient for each of the induced subgraphs. The clustering coefficient was introduced by Watts and Strogatz [13] for social network analysis. Informally, the clustering coefficient of a graph $G = (V, E)$ can be described as the ratio of the number of the triangles present in G over the number of possible triangles on its vertex. More formally, let $t(v) := |\{\{r, s\} \in E | \{r, v\}, \{s, v\} \in E\}|$. The

Table 3. Degree entropies for graphs of different sizes.

Nodes (thousand)	Degree entropy	Remaining
100	0.081	0.372
200	0.114	0.395
400	0.135	0.409
2800	0.177	0.432

clustering coefficient $C(G)$ of G is computed by considering the set V' of all the vertices whose degree is strictly greater than 1:

$$C(G) := 1/|V'| \sum_{v \in V'} t(v) \Big/ \binom{d(v)}{2}.$$

Alternatively, the clustering coefficient can be defined by considering also the nodes with degree equal to 1, defining their single contribution to be one. The upper curve in the last plot shows the behavior of the clustering coefficient according to the first definition, while the lower curve plots the other case. Especially for the first case, we can observe that the clustering coefficient varies very little with the size of the graph.

4.4. Social network measures

We also studied some of the measures that are typically evaluated for social networks. In particular, following [8], we computed the entropies of the degree distributions, the betweeness centrality of the nodes (for small graphs) and we evaluated the *assortativeness* of our graphs.

Table 3 reports entropies' values for graphs of different sizes. The first column of the table is the entropy of the degree distribution. In the second column the entropy of the *remaining degree distribution* is shown. The remaining degree distribution is defined as $R(k) = (k+1) \cdot P(k+1)/\bar{k}$ where $P(\cdot)$ is the degree distribution and $\bar{k}$ is the average degree. Both columns are normalized between 0 and 1, with 1 being the entropy of the uniform distribution. We can note that both entropies slightly increase as graphs get larger.

We also found out that our graphs can be classified as an *assortative* network: figure 11 shows the average degree of nearest neighbors for graphs of different size. This measure is defined as $k_{nn}(k) = \sum_k k' P(k'|k)$ where $P(k'|k)$ is the probability that a node of degree k is connected to a node of degree k'. We can see that the general trend of this function is increasing, hence nodes of high degree tend to connect with nodes of high degree. Moreover graphs of different sizes gave rise to very similar plots (not reported here).

The betweenness centrality of a node measures its 'routing' function: informally, it counts the fractions of shortest paths in which it is involved. For instance, the betweenness centrality of the central node of a star would be the maximum possible, while in a clique each node has betweenness centrality equal to zero. For a formal definition of this measure see [8]. For computing the *betweenness centrality* we first computed the largest connected components of the graph. Since this measure is very costly, we started from a 100 000 nodes graph. Its biggest connected component contained about 9000 nodes. Rather than plotting the measure for each node we computed an aggregated score which is the sum of the betweenness centrality of each node normalized so that it is 1 in the case of star graph. We found out that this score for our graph is quite high: 0.72. This means that there are several nodes that are involved in a good fraction of the shortest paths.

5. Conclusions

In this work, we have presented an extensive analysis of a large graph derived from a query log piece containing millions of queries. Several interesting characteristics of these graphs emerged from the analysis. In particular, the structure of these objects appears to be quite complex, and many of its measures are governed by power laws. These graph are noisy objects since they are derived from the clicks recorded in query logs, which in turn present a large amount of noise (in the form, for instance, of automatic clicks). We explored several ways to remove the noise. In particular we filtered the graph applying a threshold on the weight of the edges and on the number of clicks. From our analysis it seems that the underlying structure is quite resilient as the filtered versions of the graphs appear to preserve the structure of the original ones.

Our results, in particular the measures related to social networks, suggest that a query graph is an implicit social network, that relates people that do similar queries and click in the same pages. This alone is an important reason to explore this type of graph.

Other reasons for studying these graphs is to provide a support for applications such as query suggestion/reformulation/expansion. Hence the noise issue is quite crucial. In order to provide accurate suggestion, we should be able to remove as much noise as possible from the graph. Another application is to extract semantic relations from the graphs as shown in recent work by ourselves [6].

References

[1] Baeza-Yates R 2005 Applications of web query mining *European Conf. on Information Retrieval (ECIR'05) (Springer LNCS* 3408) ed D Losada and J Fernández-Luna (Berlin: Springer) pp 7–22

[2] Baeza-Yates R, Hurtado C and Mendoza M 2004 Query clustering for boosting web page ranking *Advances in Web Intelligence, AWIC 2004 Springer LNCS* 3034 (Berlin: Springer) pp 164–75

[3] Baeza-Yates R, Hurtado C and Mendoza M 2004 Query recommendation using query logs in a search engine *In EDBT Workshops* (*Springer* LNCS 3268) pp 588–96

[4] Baeza-Yates R 2007 Graphs from search engine queries *SOFSEM 2007: Theory and Practice of Computer Science (Harrachov, Czech Republic, January) (Springer LNCS* 4362) pp 1–8

[5] Baeza-Yates R, Hurtado C and Mendoza M 2007 Improving search engines by query clustering *J. Am. Soc. Inf. Sci. Tech.* **58** 1–12

[6] Baeza-Yates R and Tiberi A 2007 Extracting Semantic Relations from Query Logs *ACM KDD 2007 (San Jose, California, USA, August)* pp 76–85

[7] Beeferman D and Berger A 1999 Agglomerative clustering of a search engine query log *KDD (Boston, MA, USA)*

[8] Costa L, Rodrigues F, Travieso G and Boas P R 2007 Characterization of complex networks: a survey of measurements *Adv. Phys.* **56** 167–242

[9] Fitzpatrick L and Dent M 1997 Automatic feedback using past queries: social searching? *20th Annual Int. ACM-SIGIR Conf. on Research and Development in Information Retrieval,* pp 306–13

[10] Fonseca B M, Golgher P B, Silva de Moura E and Ziviani N 2003 Using association rules to discovery search engines related queries *First Latin American Web Congress (LA-WEB'03) (Santiago, Chile, November)*

[11] Raghavan V V and Sever H 1995 On the reuse of past optimal queries *18th Annual Int. ACM-SIGIR Conf. on Research and Development in Information Retrieval* pp 344–50

[12] Sahami M and Heilman T D 2006 A web-based kernel function for measuring the similarity of short text snippets *World Wide Web Conf.* pp 377–86

[13] Watts D J and Strogatz S H 1998 Collective dynamics of 'small-world' networks *Nature* **393** 440–2

[14] Wen J, Mie J and Zhang H 2001 Clustering user queries of a search engine *Proc. at 10th Int. World Wide Web Conference* W3C

[15] Zaiane O R and Strilets A 2002 Finding similar queries to satisfy searches based on query traces. *Proc. Int. Workshop on Efficient Web-Based Information Systems (EWIS) (Montpellier, France, September)*