Query Clustering using Click-Through Graph

Jeonghee Yi
Yahoo ! Inc.
2811 Mission College Blvd.
Santa Clara, CA 95054
USA
Jeonghee@yahoo-inc.com

Farzin Maghoul
Yahoo! Inc.
2811 Mission College Blvd.
Santa Clara, CA 95054
USA
fmaghoul@yahoo-inc.com

ABSTRACT
In this paper we describe a problem of discovering query clusters from a click-through graph of web search logs. The graph consists of a set of web search queries, a set of pages selected for the queries, and a set of directed edges that connects a query node and a page node clicked by a user for the query. The proposed method extracts all maximal bipartite cliques (*bicliques*) from a click-through graph and compute an equivalence set of queries (i.e., a query cluster) from the maximal bicliques. A cluster of queries is formed from the queries in a biclique. We present a scalable algorithm that enumerates all maximal bicliques from the click-through graph. We have conducted experiments on Yahoo web search queries and the result is promising.

Categories and Subject Descriptors
H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - *Clustering;* G.2.2 [**Discrete Mathematics Graph Theory**]: Graph Theory – *Graph algorithms*

General Terms
Algorithms, Experimentation

1. INTRODUCTION
Users who pose a query to a web search engine often have specific information needs in mind, such as finding the address of a business, an article about a historic event, a company's home page, and so on. Users select pages by clicking on links on a search engine result page that deem to be closely relevant to their intended information needs. Considering collective filtering, it is reasonable to assume a frequently clicked set of pages for a query reflects the kinds of information that the users intend to find by posing the query. Further, it is observed that users of similar information needs click on a similar set of pages, even though the queries they pose may vary, thus forming a cluster of queries and clicked pages that are more strongly connected to each other than with the rest of queries and clicked pages. Based on the observations, we hypothesize queries within such a cluster express highly similar information needs and intention and group such queries into a cluster.

We have designed a query clustering method that takes into account the query and clicked page relationship, not considering syntactic or semantic features on the query, such as keywords. It is known that approaches based on keywords are not very suitable for query clustering because of the short lengths of queries [5].

We represent the query and click-through page relationships by a directed bipartite graph that consists of a set of queries, a set of web page URLs, and a set of edges that connect a query node to a page node in the graph. The proposed query clustering method involves maximal biclique enumeration problem. Figure 1 illustrates a hypothetical click-through graph with three maximal bicliques. The proposed algorithm will prune the three edges that cross the *cut* lines, and discover three maximal bicliques as illustrated in Figure 1.

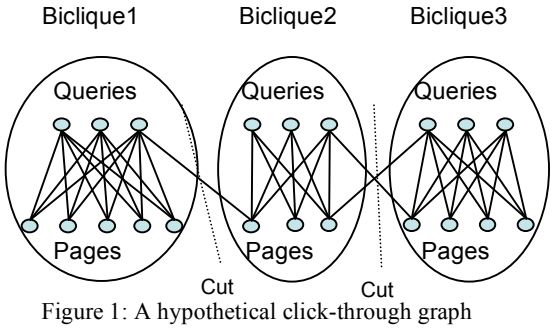

Figure 1: A hypothetical click-through graph

2. DATASET and PREPROCESSING
In this study we used a set of randomly sampled click-through data from the Yahoo web search query logs in Nov. 2008. The sample data consists of over 16M unique queries, over 10M page URLs, and over 92M edges connecting the nodes.

Like a typical web graph, the click-through graph exhibits the power law distribution in terms of the out-degree of query nodes, and the in-degree of clicked URLs. We prune the following nodes and their associated edges in the preprocessing step:

- Web pages with in-degree higher than 100 and their in-coming edges: these are pages of very broad topic and interest, such as `www.craigslist.org`, and `en.wikipedia.org`. With those types of pages, we would get too broad clusters, since the topics of queries connected to those pages tend to be vary. There are about 5% of unique URLs with this level of high in-degrees.
- Queries with out-degree greater than 10 and their out-going edges: most normal user queries result in only a small number of clicks, rarely more than 10 pages. Those with high out-degree may include ones by robots for scraping or some special types of queries. There are only about 0.1% of URL queries in this category.
- Web pages with in-degree 1 and their in-coming edge.
- Queries with out-degree 1 and their out-going edge.
- Edges with click frequency less than a threshold τ

3. QUERY CLUSTERING ALGORITHMS
3.1 Biclique Generation
[3] shows maximal bicliques generation from a bipartite graph is a special case of the maximal clique generation problem from a general graph. Let $G = (V_1 \cup V_2, E)$ be a bipartite graph, where V_1

and V_2 are the two disjoint sets of nodes, and E is a set of edges connecting nodes in V_1 and V_2. To generate maximal bicliques, G is transformed to a general graph $G'=(V_1 \cup V_2, E')$, where $E'=E \cup (V_1 \times V_1) \cup (V_2 \times V_2)$. Then the maximal clique generation algorithm for a general graph, such as the one in [4], can be applied on G'. This algorithm, however, requires an increased amount of the main memory space proportional to the entire expanded graph.

We deal with a very large click-through graph that would hardly fit into main memory. We instead modified the bipartite core generation algorithm in [2] to generate maximal bicliques. One of the major advantages of the algorithm is that it does not require to store the entire graph in memory. Instead it applies various pruning techniques on sorted lists of nodes; one for queries, and another for pages for click-through graph. The algorithm can be further optimized to sort only once, and build only a small index in main memory.

3.1.1 Iterative pruning

Since we are looking for query clusters larger than certain size, queries and pages of which in- and out-degrees do not meet the minimum size requirement can be eliminated.

To compute a biclique of size (i,j), query nodes with out-degree smaller than i and their out-going edges are pruned. Similarly, any page node with in-degree smaller than j and its associated edges are pruned. This pruning step is iteratively applied until there exists no more such nodes.

3.1.2 Biclique generation

At each step of the biclique generation algorithm we either generate a biclique, or exclude a node and the associated edges from the graph. After generating a biclique, the subgraph corresponding to the biclique is removed from the click-through graph. Starting from the maximum out-degree size, we repeat the following steps iteratively for each decreasing value of i:

1. From a sorted list of queries, find all queries, q_k, with out-degree i, and list the neighbors of each q_k, $P(q_k)$.
2. For each $P(q_k)$, generate the set of all in-coming queries, $Q(p_l)$, of each page, $p_l \in P(q_k)$. (An index on the page URLs is used for better performance.)
3. Find the intersection of all $Q(p_l)$; $\bigcap_{l \in P(q_k)} Q(p_l)$.
4. Let m denote the size of the query set, $\bigcap_{l \in P(q_k)} Q(p_l)$, and E be a set of all edges between $\bigcap_{l \in P(q_k)} Q(p_l)$ and $P(q_k)$.
   - 4.1. If $(m \geq j)$, generate a biclique of size (i,m), $(\bigcap_{l \in P(q_k)} Q(p_l) \cup P(q_k), E)$. After generating a biclique, remove all query and page nodes, and edges in the biclique from the click-through graph (unless the node is a part of another cluster).
   - 4.2. If $(m < j)$, remove all in-coming edges of nodes in $P(q_k)$ that do not connect to a query node in the intersection $\bigcap_{l \in P(q_k)} Q(p_l)$. (This removes edges between the bicliques, the edges that cross the cut lined on Figure 1.)
5. After removing all edges, apply the iterative pruning before continuing with the next iteration with out-degree size i-1.

The generated bicliques are maximal.

3.2 Query Cluster Generation

For each biclique generated, the query set, $\bigcap_{l \in P(q_k)} Q(p_l)$ forms an equivalence set that becomes a query cluster.

4. EXPERIMENTS

After preprocessing with τ=2, there remain ~1.15M and ~2M query and page nodes, and ~68M edges in the sample graph, reduced from ~16M and ~10M nodes and ~ 92M edges, respectively.

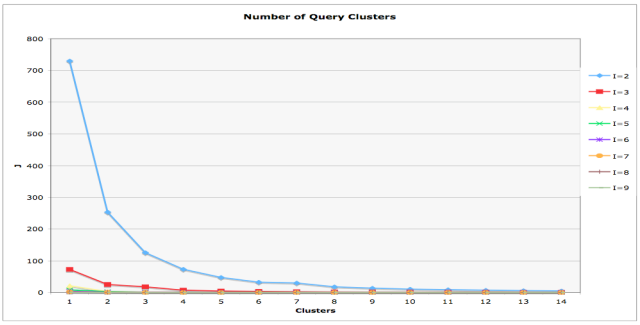

Figure 2: Number of query clusters

Figure 2 plots the number of query clusters extracted by our algorithm. As expected, the numbers of maximal bicliques drop significantly as the size of the cliques grow. The number may be interpreted as lower bound of query clusters, as our method considers only maximal bicliques. As we relax the equivalence condition and consider strongly connected, but not necessarily completely connected bipartite subgraphs as the candidates, it may further reveal interesting quasi-equivalence sets of queries. Due to the strict requirement of complete connectedness of the clusters by the current algorithm, many potentially interesting query clusters are excluded if they slightly violate the requirements.

5. DISCUSSION AND FUTURE WORK

We have studied a problem of discovering query clusters and proposed an algorithm that utilizes click-through data from search engine query logs. The proposed method identifies all bicliques and all queries in a biclique are equivalent in terms of user information needs.

In the future, we plan to extend the algorithm to consider *quasi-bicliques* as candidates of query clusters. This is expected to be a more robust algorithm under the presence of slightly irregular click patterns, if the threshold is properly set. We plan to apply a frequency weight of each edge between a query and a clicked page to distinguish noisy clicks.

6. REFERENCES

[1] J. J. Carrasco, D. C. Fain, K. J. Lang, and L. Zhukov. Clustering of bipartite advertiser-keywork grdaph. *Workshop on Large Scale Clustering, ICDM* 2003.

[2] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, Trawling the web for emerging cyber-communities. *The 8th Int. World Wide Web Conference*, 1999.

[3] K. Makino, and T. Uno, New algorithms for enumerating all maximal cliques, *The 9th Scandinavian Workshop on Algorithm Theory*, 2004.

[4] E. Tomita, A. Tanaka, and H. Takahashi. The worst-case time complexity for generating all maximal cliques and computational experiments. *Theoretical Computer Science*, 363(1), pp.28-42, 2006.

[5] J. Wen, J. Nie, H. Zhang. Query clustering using user logs. *ACM Transactions on Information Systems*, 20(1), pp. 59-31, 2002.