The Recommendation Click Graph: Properties and Applications*

Yufei Xue, Yiqun Liu, Min Zhang, Shaoping Ma, and Liyun Ru

State Key Laboratory of Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University,
Beijing, 100084, China
yufei.xue@gmail.com

Abstract. Query recommendations help users to formulate better queries and to obtain the desired search results. Users' clicks on query recommendations contain a great deal of information about search intent, query ambiguity and search performance. We use query recommendation click information contained in search logs to construct a recommendation click graph. A directed edge in the graph connects the prior query and the clicked recommended query. By analyzing the graph, we develop methods for finding ambiguous queries and improving the search results. The experimental results show that our method for finding ambiguous queries is effective, and using recommendation click information can improve the search performance of ambiguous queries.

Keywords: query recommendation, user behavior, search intent.

1 Introduction

Query recommendation technology is widely used by commercial search engines. A search engine provides related queries in a search result page. If the query does not present useful search results, users can click on any related query to find more web resources. Our analysis shows that users click recommended related queries in approximately 15% of search sessions.

The query recommendation click log contains a large volume of information about users' search intents and their perspectives on search results. A user's click on a recommended query implies that the recommended query describes what he wants, whereas the search results of previous do not satisfy his information needs well enough. Because search engine users usually click recommended queries after reviewing the result page, the clicked recommendation should be a direct and precise reflection of the user's intent. The query recommendation click logs contain less noise than the logs of query reformulation because all clicked queries

* This work was supported by Natural Science Foundation (60903107, 61073071) and National High Technology Research and Development (863) Program (2011AA01A205) of China.

M. Zhou et al. (Eds.): NLPCC 2012, CCIS 333, pp. 284–295, 2012.

are selected by both the search engine and the user. In this sense, the data in query recommendation click log is very reliable.

In this paper, we introduce the concept and properties of a query clicking graph, which is a graph that depicts all of the query recommendation clicking information contained in user logs. This graph aggregates all users' query recommendation click actions during a specific period. In the graph, each node represents a query. A directed edge (q_i, q_j) indicates that a user clicked the recommended query, q_j, when he searched q_i. We do both global and local analysis on the graph. These efforts can help us to learn the properties of queries and user intents. We introduce the following applications for the recommendation click graph:

Optimizing search results. Exploring users' click actions on query recommendations can help us to understand users' search intents. Thus, the recommendation click graph can help the search engine to improve the search performance.

Recognizing ambiguous queries. If users click disparate recommended queries, the previous query might be highly ambiguous. The ambiguity might be caused by the ambiguity of the query expression or the users' uncertain search intent. Graph analysis algorithms can help us to find ambiguous queries in a recommendation click graph.

The remainder of this paper is organized as follows. Section 2 introduces related work. Basic concepts and assumptions of our work are given in Section 3. In Section 4, we discuss the properties of recommendation click graphs. Section 5 discusses our local analyzing methods for recommendation click graphs to improve search performance. Section 6 shows our approach to identifying ambiguous queries. In Section 7, we summarize our work and discuss query recommendation and the recommendation click graph.

2 Related Work

The Web is naturally presented as a graph. Often, the nodes of a web graph represent the web pages, and the edges represent links between pages. Web graphs are used to estimate the quality of web pages. Link analysis is a data-analysis technique that uses a link graph to estimate page quality. Examples of well-known link analysis algorithms are the HITS [1], PageRank [2] [3] and TrustRank [4] algorithms. These link analysis algorithms have common assumptions: (1) the links imply recommendation and (2) the page quality can spread through the hyperlinks. With these assumptions, link analysis algorithms are often applied to other types of graphs.

Search queries can also be presented by graphs. Baeza-Yates defined 5 types of query graphs [5]: word, session, URL cover, URL link and URL terms. Three types of information are used to construct query graphs: the query terms, the searching and clicking behaviors during the query session and the content of clicked URLs. Baeza-Yates et al. used some of the query graphs to mine query logs for semantic relations between queries [6]. The five types of graphs are useful for recommending related queries. A query-flow graph is a representative query graph [7] and performs well in query recommendation applications [8] [9].

Unlike present methods, the recommendation click graph is based on existing query recommendations and the click log. This graph is used to detect ambiguity in queries and to learn users' intents.

Query recommendation is an important are of study. The query recommendation algorithms focus on finding similar queries. Zaiane et al. used query terms, search result snippets and other simple text information to find similar queries [10]. Because the semantic information from query and search results is limited and hard to analyze, most recent studies on query recommendation use user behavior data to detect related queries [11] [12] [13]. Some studies [14] [15] use a bipartite graph of search query and clicked URL to determine and recommend related queries. These studies assume that two queries are similar if they lead users to click on the same URLs. Kelly et al. studied the usage of query recommendation [16]. Their research showed that query recommendations are frequently used by users and can be very helpful.

For a search engine to understand a user's search intent, it must be able to identify an ambiguous query. Some queries are ambiguous in nature. The search intent of such queries can be detected by analyzing related documents and user behavior. Song et al. summarized three types of queries: ambiguous, broad and clear. They used a supervised learning approach to classify queries by analyzing the text of the search results [17]. He and Jhala developed a method to understand a user query based on a graph of connected related queries [18]. Veilumuthu and Ramachandran developed a clustering algorithm that uses the user session information in the user log and query URL entries to identify clusters of queries that have the same intention [19].

3 Preliminaries

3.1 Basic Concepts

Recommendation click log. A recommendation click log is a part of a search engine's user log. It records all clicks on query recommendations. Recommendation click pair. A user's click on a query recommendation is represented by an ordered query pair $\langle q_i, q_j \rangle$. This pair indicates that a user submitted query q_i to the search engine and clicked recommended query q_j on q_i's search result page. We call q_i the source query and q_j the destination query. Pairs with the same source and destination queries are recorded as a single recommendation click pair in our work.

Recommendation click graph. The recommendation click graph uses the previous two concepts.

Definition 1: The recommendation click graph is a directed graph $G_c = (V, E)$, where:

- the set of nodes, $V = \{q | q$ appear in the recommendation click log as a source or destination query$\}$, and
- $E = \{(q_i, q_j) | < q_i, q_j >$ is a recommendation click pair in the recommendation click log$\}$.

Like other web graphs, the recommendation click graph might have more than one weakly connected component. Because the recommendation click graph is based on recommendation click pairs, there are no isolated vertices in the graph, i.e., there are at least two queries in a connected component. More properties of the recommendation click graph are described in following sections.

3.2 Semantic Basis

Some basic assumptions on the semantics of query recommendation click actions are fundamental to the recommendation click graph developed here. When a user clicks a recommended query, this action expresses a relationship between the source and destination queries. Using the recommendation click pair $\langle q_i, q_j \rangle$ as an example, we assume that there are two possible latent meanings for a user's click on a query recommendation:

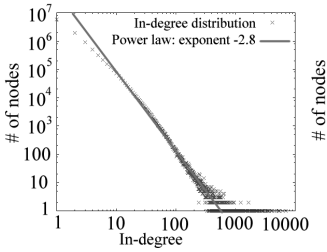
- *Assumption 1* q_j describes the user's information needs more precisely than q_i, or
- *Assumption 2* q_j does not describe the user's information needs more precisely than q_i, but the user is interested in q_j and wants more information.

Clicks described by assumption 1 usually make the user's search intent more precise. For example, the recommendation click pair ⟨ Lady Gaga songs, Lady Gaga poker face⟩ corresponds to a more precise intent description. Clicks described by assumption 2 appear frequently. The recommendation click pair ⟨ Lady Gaga songs, top 100 songs⟩ is an example of this type of click. In most cases, the source query and destination query are related. However, the query pair can be about different topics.

In our work, we are interested in recommendation click pairs that comprise related queries. The recommendation click graph can help researchers to view related queries from different perspectives.

4 Properties of the Recommendation Click Graph

We extracted a recommendation click log from an August 2010 user log of a Chinese commercial search engine. The log recorded 58,334,303 clicks on query recommendations. From these data, we constructed a recommendation click graph that contained 23,516,620 vertices and 31,569,262 directed edges. We obtained statistical distributions for the in-degree and out-degree of each vertex. The distributions are shown in Fig. 1. Both the in-degree and out-degree distributions approximately follow a power law distribution. This property is very similar to a hyperlink graph. Given this similarity, there are many effective and widely used analyzing algorithms developed for hyperlink graphs that might also be appropriate for recommendation click graphs.



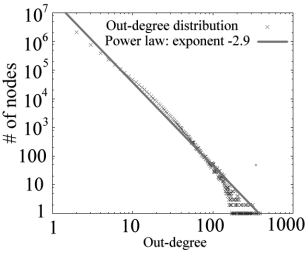


Fig. 1. Distribution of in-degree and out-degree

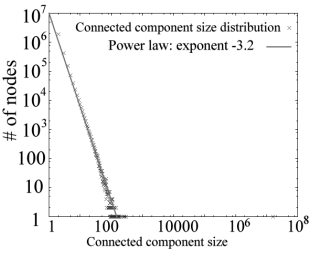


Fig. 2. Distribution of component size

4.1 Connected Components

We analyzed the connected components in the recommendation click graph without regard to the direction of the edges. Of the 2,668,331 components in the graph, 71% have only 2 vertices, i.e., most of the components are very small. Our statistics show that approximately 60 components have more than 100 vertices. The largest component has 16,298,916 vertices, which is approximately 70% of the vertices in the graph. This result indicates that many queries are connected in the recommendation click graph. These queries include most of the hot topics of the search engine during the period covered by the query log. As shown in Fig. 2, the distribution of the connected component sizes approximately follows a power law distribution.

4.2 Strongly Connected Components

A strongly connected component is a directed sub-graph in which there is a directed path between any ordered pair of vertices. In a recommendation click graph, queries that appear in the same strongly connected component can be strongly related.

In our recommendation click graph, there are 20,978,260 strongly connected components. The distribution of component sizes follows a power law distribution. Of the strongly connected components, there are 20,695,423 components that have only one vertex. These vertices account for approximately 88% of the vertex set. This result indicates that approximately 10% of the queries both recommend other queries and are recommended by other queries in the graph. This property is in accord with user behavior patterns and can be explained as follows. Because the user queries follow a power law distribution, most queries have a low frequency. Low frequency queries have no chance to be recommended. As mentioned in Section 3.2, for many recommendation click pairs, the destination query refined the intent of the source query. Therefore, few queries are both source and destination. In the recommendation click graph, the largest strongly connected component contains 1,816,759 vertices, approximately 7.7% of the vertices in the graph.

5 Local Analysis of a Recommendation Click Graph

In a recommendation click graph, an edge (q_i, q_j) denotes that a user has clicked the query recommendation q_j while he searched q_i. Because the recommendation click pair is selected by both the search engine and the user, we assume that adjacent queries in a recommendation click graph are semantically similar.

A user clicks a query recommendation when he finds the search result insufficient. He may try a recommended query that describes his needs more precisely or a query that appears to yield better results. The search result page of the related queries might provide what the user is seeking. The local analysis of a recommendation click graph can help us to improve the search result by including some search results of high-quality adjacent queries.

5.1 Local Subgraph

Definition 2: For query q_i in a recommendation click graph, we define a local subgraph of q_i as

$$G_{Sub(i)} = (V_{Sub(i)}, E_{Sub(i)}),$$

where

$$V_{Sub(i)} = q | distance(q, q_i) \leq 2 \cup q_i, E_{Sub(i)} = E \cap (V_{Sub(i)} \times V_{Sub(i)}).$$

The local subgraph of q_i contains the queries that are most related to q_i.

5.2 HITS applied to a Local Subgraph

HITS is a link analysis algorithm for evaluating web pages. The algorithm is applied to a subgraph of a hyperlink graph. The indicators for each web page are called hubs and authorities. The hub value indicates how efficiently users are led to other useful pages using the hyperlinks on the page. The authority value indicates how many good hub pages link to the page.

We apply the algorithm and concepts of HITS to a recommendation click graph. A query that has a high authority value is linked to several other queries. A query that has a high hub value leads users to other queries. In this work, we are interested in the queries that are frequently searched and do not lead to additional clicks of query recommendations. Queries that have a high authority value and a low hub value might satisfy more users' needs and be less ambiguous. We find such queries using the local graph of query q_i and use their search results to improve the search performance of q_i.

5.3 Experiments of Optimizing Search Results

We randomly selected 442 queries that had an out-degree 8 and extracted their local subgraphs. The local subgraphs of queries that have an out-degree less than 8 might not be large enough to support a HITS analysis.

We collected statistics on the numbers of nodes and edges in the 442 subgraphs. Fig. 3 shows that the numbers of nodes and edges are linearly related. Table 1 lists the maximum, minimum and average numbers of nodes and edges. These statistics show that the sizes of the local subgraphs can vary widely.

Table 1. Properties of local subgraphs

	Maximum	Minimum	Average
# of nodes	18271	14	865.1
# of edges	78808	13	3009.4
(# of edges)/(# of nodes)	12.2	0.93	2.8

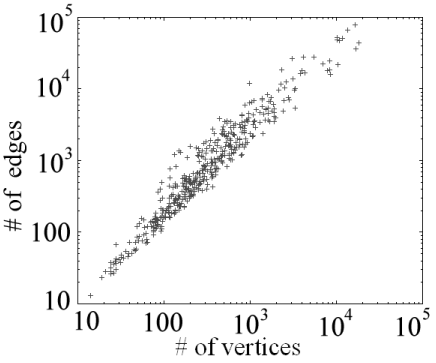


Fig. 3. Dimensions of local subgraphs

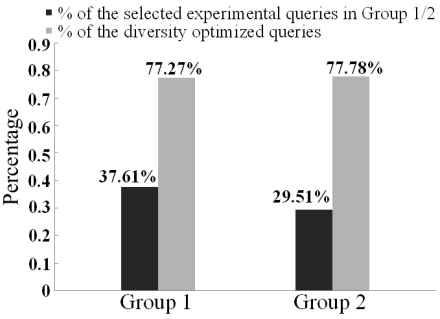


Fig. 4. Experiment Results on Algorithm 1

We used the query's local subgraph to optimize the search results. We randomly selected 117 queries (Group 1) that had an out-degree = 8 or 9. These queries were not the most frequent queries in the search engine, but their local subgraphs were large-enough to support the HITS algorithm. These queries might be a representation of a general user query. We examined Group 1 from the user's perspective. We identified 44 queries on which the search engine did not perform very well, but the user found related resources through a query recommendation. We performed Algorithm 1 on the 44 queries and obtained optimized search results. Three annotators were asked to compare the optimized search results with the original search results and vote for the list that contained results that are more diverse. For 34 of the 44 selected queries (approximately 77%) the search results optimized using Algorithm 1 were deemed to be more diverse.

Algorithm 1.

1: Select q_i as the query to be improved.
2: Extract q_i's local subgraph $G_{Sub(i)} = (V_{Sub(i)}, E_{Sub(i)})$ from the recommendation click graph.
3: Iteratively calculate the authority and hub values for all queries in the subgraph using the HITS algorithm.
4: Select a threshold hub value, h_t for hub value. Create a query set $Q = \{q|Hub(q) < h_t, (q_i, q) \in E_{Sub(i)}\}$.
5: Sort the queries in Q by authority values and take n largest queries.
6: Construct 2 search result sets: $R_i = \{r|r \in SearchResultsOf(q_i)\}$, $R = \{r|r = TopResultOf(q), q \in Q\}$
7: Sort the search results in $R_i \cap R$ according to the rank in q_i's search results and store in List L_1;
8: Sort the search results in $R \setminus R_i$ according to the order of the authority of the corresponding query, and store in List L_2;
9: Sort the search results in $R_i \setminus R$ according to its rank in q_i's search results and store in List L_3;
10: Output individual results from L_1, L_2, L_3 in turn, until there are 10 search results.

We repeated the above experiment on 61 randomly selected queries (Group 2) that had higher search frequencies and out-degrees. These queries might be more ambiguous than the queries in Group 1. We examined the queries and found 18 queries on which the search engine did not perform well. For 14 queries, the search results produced by our algorithm were more diverse. Because the queries in Group 2 are hotter than Group 1, the queries in Group 2 might perform better in the search engine. Note that the percentage of queries that do not perform well enough is lower for Group 2 (18/61) than for Group 1 (44/117). Approximately 77% of the queries selected in Groups 1 and 2 were optimized by Algorithm 1. The experimental results are shown in Fig. 4.

Our algorithm is naive and does not always yield better results than those obtained by the search engine, particularly when the search engine performs a query well. Nevertheless, our local subgraph method can improve the search results of queries.

6 Finding Ambiguous Queries

6.1 Inverse PageRank

Inverse PageRank is a link analysis algorithm used to determine the seed pages for the TrustRank algorithm [4].

For a directed graph $G = (V, E)$, inverse PageRank requires a related graph $G' = (V, E')$, where

$$(q_i, q_j) \in E' \iff (q_i, q_j) \in E.$$

We perform PageRank on the link-inverted graph G' to obtain the inverse PageRank for G.

The inverse PageRank can be explained using the concepts of PageRank as follows:

- In G, a vertex's inverse PageRank is higher if it has more out-links.
- In G, a vertex's inverse PageRank is higher if the vertices to which it points have high inverse PageRanks.

Some studies have shown that search engine users often issue short and ambiguous queries [20]. In this case, users might continue by clicking query recommendations. It is reasonable to assume that the level of ambiguity of a query is related to the dispersion of the query's recommendation clicks. Based on this assumption, the inverse PageRank values of a recommendation click graph show the level of query ambiguity.

6.2 Experiments

We applied the inverse PageRank algorithm on the recommendation click graph. We sorted the inverse PageRank values into descending order and divided them into 10 buckets. The sums of the inverse PageRank values in each bucket were equal. The sizes of buckets are shown in Fig. 5.

To verify if the inverse PageRank can be used to determine the ambiguity of queries, we asked 4 annotators to examine queries sampled from the 10 buckets. We randomly selected 1010 queries and asked the annotators to score each query: 3 indicated a clear query, 2 indicated an intent ambiguous or broad query, and 1 indicated a semantically ambiguous query. Through discussions, the annotators ensured that they had no serious disagreement on the semantics of any query. For 828 of the 1010 queries, at least three annotators give the same score. The kappa coefficient of their annotations was $\kappa = 0.466$ [21].

For each query, we calculated the average, maximum and minimum scores. In addition, we calculated the average of the 3 types of statistical scores in each bucket. The results are shown in Fig. 6. The scores in the first five buckets are higher than the scores in the last five buckets. Moreover, the scores increase from Bucket 1 to Bucket 5, whereas the scores of the last five buckets are not significantly different. Although the first 5 buckets contain approximately 30% of the queries, they contain most of the ambiguous queries.

We randomly selected 233 queries from the graph and sorted the queries into descending inverse PageRank order. We divided the queries into 10 buckets of approximately the same size. The annotators labeled the top 3 search results of the 233 queries in 5 levels, and we calculated the NDCG3 for all of the queries. For each bucket of queries, we computed the average and standard deviation of the NDCG3. Fig. 7 shows the statistical results. From Bucket 1 to 10, the average NDCG3 decreases, whereas the standard deviation increases. Although queries that have higher inverse PageRanks contain more ambiguity, these results indicate that they perform better than queries that have lower inverse PageRanks. This result seems counterintuitive, but it is reasonable. As mentioned above, the inverse PageRank algorithm gives higher values to the vertices that have more out-links.

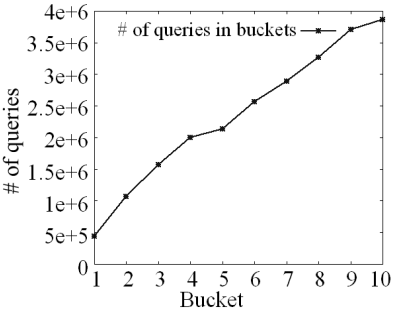


Fig. 5. Bucket size

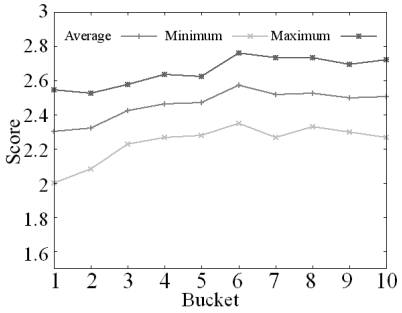


Fig. 6. Ambiguity score statistics

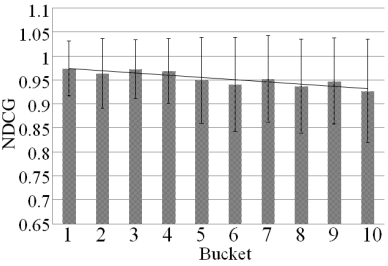


Fig. 7. Avg. and Stdev. of NDCG values

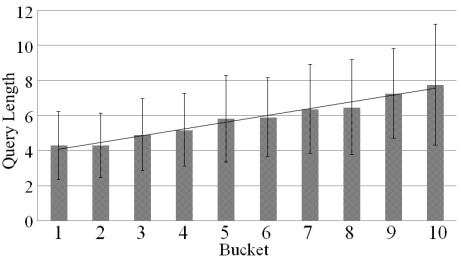


Fig. 8. Avg. and Stdev. of query lengths

In a recommendation click graph, the out-degree is related to the search frequency of the query. Therefore, hot queries are more likely to receive higher inverse PageRanks. The higher NDCG3 values obtained in the previous buckets are explained by the better performance of hot queries in commercial search engines. The average lengths of the queries increase from Bucket 1 to Bucket 10 (shown in Fig. 8). The work of Jansen et al. [20] showed that most queries in web search engines are short, and long queries are very infrequent. This work confirms that the queries in the preceding buckets are hotter in the search engine.

From the experimental results and analysis, applying the inverse PageRank algorithm on a recommendation click graph is an effective tool to evaluate query ambiguity.

7 Discussions and Conclusions

According to the definition and construction of recommendation click graphs, the graph is strongly related to user behaviors and the search engine's query recommendation algorithms. Because the user behaviors on query recommendations is the object of our research, the behaviors and algorithms do not affect our methods of constructing and analyzing the recommendation click graph.

However, changes in the recommendation algorithm can affect the properties of the graph.

In the commercial search engine that we used to build the recommendation click graph, there are at most 10 recommended queries for each input query. However, the query recommendation algorithm allows the related queries to be updated, and the out-degree may be greater than 10. In the graph for our experiment, the largest out-degree was 391, and there were 254,639 queries that had an out-degree greater than 10. In other search engines, the strategies for updating query recommendations might be different; thus, the distribution of the vertices' out-degrees might be different. In our work, we considered the out-degree as a representation of query ambiguity, and the recommendation algorithm can affect our judgment of query ambiguity. However, it is reasonable to believe that frequent changes in recommended queries imply that the source query is complex and unclear. Therefore, our methods are applicable in different search engines.

In this paper, we noted that the query recommendation click log contains a large volume of information about user search intent and query ambiguity. We proposed a recommendation click graph constructed from the recommendation click log. Our analysis showed that the properties of the recommendation click graph are similar to traditional web graphs. Furthermore, the edges in both web graphs and query graphs have the meaning of "like" or "recommend". Therefore, it is reasonable to apply well-used link analysis algorithms to the recommendation click graph. In this way, we can improve search performance by mining the recommendation click graph to learn more about users' behaviors and intents. We have applied the HITS algorithm on a query's local subgraph to select reliable recommended queries. These related queries can be used to improve the diversity and performance of the query. We have applied the inverse PageRank algorithm to a recommendation click graph to evaluate query ambiguity. The results showed that the inverse PageRank values reflect the query ambiguity. This result was confirmed by both annotation and query length statistics.

An important direction for future work is to analyze the recommendation click graph to learn users' search intents. Furthermore, an advanced ranking algorithm can be developed to improve search results using query recommendations.

References

1. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. J. ACM 46, 604–632 (1999)
2. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems 30(1-7), 107–117 (1998); Proceedings of the Seventh International World Wide Web Conference
3. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web (1998)
4. Gyöngyi, Z., Garcia-Molina, H., Pedersen, J.: Combating web spam with trustrank. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, vol. 30, pp. 576–587. VLDB Endowment (2004)

5. Baeza-Yates, R.: Graphs from Search Engine Queries. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 1–8. Springer, Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-69507-3_1
6. Baeza-Yates, R., Tiberi, A.: Extracting semantic relations from query logs. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2007, pp. 76–85. ACM, New York (2007)
7. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., Vigna, S.: The query-flow graph: model and applications. In: Proceedings of CIKM 2008 (2008)
8. Boldi, P., Bonchi, F., Castillo, C., Donato, D., Vigna, S.: Query suggestions using query-flow graphs. In: Proceedings of the 2009 Workshop on Web Search Click Data, WSCD 2009, pp. 56–63. ACM, New York (2009)
9. Bai, L., Guo, J., Cheng, X.: Query Recommendation by Modelling the Query-Flow Graph. In: Salem, M.V.M., Shaalan, K., Oroumchian, F., Shakery, A., Khelalfa, H. (eds.) AIRS 2011. LNCS, vol. 7097, pp. 137–146. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-25631-8_13
10. Zaïane, O.R., Strilets, A.: Finding Similar Queries to Satisfy Searches Based on Query Traces. In: Bruel, J.-M., Bellahsène, Z. (eds.) OOIS 2002. LNCS, vol. 2426, pp. 207–216. Springer, Heidelberg (2002), http://dx.doi.org/10.1007/3-540-46105-1_24
11. Baeza-Yates, R., Hurtado, C., Mendoza, M.: Query Recommendation Using Query Logs in Search Engines. In: Lindner, W., Fischer, F., Türker, C., Tzitzikas, Y., Vakali, A.I. (eds.) EDBT 2004. LNCS, vol. 3268, pp. 588–596. Springer, Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-30192-9_58
12. Yan, X., Guo, J., Cheng, X.: Context-aware query recommendation by learning high-order relation in query logs. In: Proceedings of CIKM 2011 (2011)
13. Szpektor, I., Gionis, A., Maarek, Y.: Improving recommendation for long-tail queries via templates. In: Proceedings of the 20th International Conference on World Wide Web, WWW 2011, pp. 47–56. ACM, New York (2011)
14. Ma, H., Yang, H., King, I., Lyu, M.R.: Learning latent semantic relations from clickthrough data for query suggestion. In: Proceeding of CIKM 2008 (2008)
15. Mei, Q., Zhou, D., Church, K.: Query suggestion using hitting time. In: Proceeding of CIKM 2008 (2008)
16. Kelly, D., Cushing, A., Dostert, M., Niu, X., Gyllstrom, K.: Effects of popularity and quality on the usage of query suggestions during information search. In: Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, pp. 45–54. ACM, New York (2010)
17. Song, R., Luo, Z., Wen, J.R., Yu, Y., Hon, H.W.: Identifying ambiguous queries in web search. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, pp. 1169–1170. ACM, New York (2007)
18. He, X., Jhala, P.: Regularized query classification using search click information. Pattern Recognition 41(7), 2283–2288 (2008)
19. Veilumuthu, A., Ramachandran, P.: Intent based clustering of search engine query log. In: Proceedings of the Fifth Annual IEEE International Conference on Automation Science and Engineering, CASE 2009, pp. 647–652. IEEE Press, Piscataway (2009)
20. Jansen, B.J., Spink, A., Bateman, J., Saracevic, T.: Real life information retrieval: a study of user queries on the web. SIGIR Forum 32, 5–17 (1998)
21. Fleiss, J.L.: Measuring nominal scale agreement among many raters. Psychological Bulletin 76(5), 378–382 (1971)