Query Suggestions Using Query-Flow Graphs

Paolo Boldi[1*]
boldi@dsi.unimi.it

Francesco Bonchi[2]
bonchi@yahoo-inc.com

Carlos Castillo[2]
chato@yahoo-inc.com

Debora Donato[2]
debora@yahoo-inc.com

Sebastiano Vigna[1*]
vigna@dsi.unimi.it

[1]DSI, Università degli
Studi di Milano, Italy

[2]Yahoo! Research Labs
Barcelona, Spain

ABSTRACT

The query-flow graph [Boldi et al., CIKM 2008] is an aggregated representation of the latent querying behavior contained in a query log. Intuitively, in the query-flow graph a directed edge from query q_i to query q_j means that the two queries are likely to be part of the same search mission. Any path over the query-flow graph may be seen as a possible search task, whose likelihood is given by the strength of the edges along the path. An edge (q_i, q_j) is also labelled with some information: e.g., the probability that user moves from q_i to q_j, or the type of the transition, for instance, the fact that q_j is a specialization of q_i.

In this paper we propose, and experimentally study, query recommendations based on short random walks on the query-flow graph. Our experiments show that these methods can match in precision, and often improve, recommendations based on query-click graphs, without using users' clicks. Our experiments also show that it is important to consider transition-type labels on edges for having good quality recommendations.

Finally, one feature that we had in mind while devising our methods was that of providing diverse sets of recommendations: the experimentation that we conducted provides encouraging results in this sense.

1. INTRODUCTION

Query recommendations are an important tool that helps search engines users in their information seeking activities, also known as search missions [19]. Recommendations are typically queries similar to the original one, and they are usually obtained by analyzing the query logs, for instance, finding recommendations by clustering of queries [28], or by identifying frequent re-phrasings [3]. The main source of information for building search assisting systems are *query logs*.

Web query logs describe how users interact with a search engine. They are very large, very diverse (containing millions of different queries), heavy-tailed (containing queries at different orders of magnitude of frequency), and noisy (containing thousands of variants and spellings of the same information need). Distilling behavioral patterns from query logs is a key step towards improving the service provided by search engines and towards developing innovative web-search paradigms. In particular, and this is the focus of this paper, *by analyzing query logs we can build rich models of user searching activities*, and use these models in applications aimed at improving the user web-search experience, such as query recommendations.

One current research line we develop attempts to infer the hidden semantics of user interactions with search engines by extracting data from a query log in two steps. First, we identify *search mission* borders by distinguishing query transitions that are *reformulations*, i.e., queries with a similar information need [24, 19], from query transitions that represent a mission change. We tackled this in [8], where we built a machine learning model for predicting the probability that two subsequent queries are part of the same search mission.

After identifying the search missions, the query reformulations inside each mission can be classified into *query reformulation types* [26]. In particular, in our work [9] we identified 4 query reformulation types (abbreviated QRT): *generalization, specialization, error correction*, and *parallel move*, and showed that accurate automatic classification of QRTs is indeed possible. Learning automatically from a human-labeled query log sample, we built a model for automatic classification of QRTs which exhibited a very high accuracy, $\approx 92\%$ discriminating among 4 different reformulation types. Using such model we can annotate the edges of the *query flow graph* with QRTs obtaining a richer model of the users' querying behavior.

In this paper we show applications of this annotated query-flow graph for generating query recommendations. Our methods are essentially different versions of short random walks on different slices of the query-flow graph. We performed an extensive experimentation using the "Spring 2006 Data Asset" distributed by Microsoft Research. Our experiments show that other methods can match in precision (and often improve) query-click based recommendations without using clicks. Moreover our methods provide more diversity in the result sets. Our experiments also show that transition probabilities from one query to the next are not enough, and for obtaining good recommendations it is important to filter out

[*]Part of this work was done while the authors were visiting Yahoo! Research Labs, Barcelona

WSCD '09, Feb 9, 2009 Barcelona, Spain.
Copyright 2009 ACM 978-1-60558-434-8 ...$5.00.

queries that are not part of the same search mission, and to add QRT labels to edges.

The next section presents related work. Section 3 summarizes our previous results on the query-flow graph and Section 4 presents the experimental framework we use. Finally, Section 5 presents our evaluation results and Section 6 some concluding remarks.

2. RELATED WORK

Query graphs. Baeza-Yates [2] identifies five different types of query graphs. In all cases, the nodes are queries; a link is introduced between two nodes respectively if: (*i*) the queries contain the same word(s) (*word graph*), (*ii*) the queries belong to the same session (*session graph*), (*iii*) users clicked on the same URLs in the list of their results (*URL cover graph*), (*iv*) there is a link between the two clicked URLs (*URL link graph*) (*v*) there are l common terms in the content of the two URLs (*link graph*). Baeza-Yates and Tiberi [4] study a weighted version of the *cover graph.* Their analysis provides information not only about how people query but also about how they behave after a query and the content distribution of what they look at. Moreover the authors study several characteristics of *click graphs*, i.e., bipartite graphs of queries and URLs, where a query and a URL are connected if a user clicked on the URL that was an answer for the query. This framework is used to infer semantic relations among queries and to detect *multitopical URLs*, i.e., URLs that cover either several topics or a single very general topic.

Query-document graphs (also known as query-click graphs) are introduced by [6]. They are bi-partite graphs in which the nodes in the one set are queries and the ones in the other set are documents; an edge appears between a query q and a document d if the user that issued the query, clicked on the URL correspondent to d in the list of results. We note that a bi-partite query-document graph can be turned into a query-query graph by means of the following procedure, described in [23]. Let $G = (V, E)$ be a bipartite graph such that $V = Q \cup D$, $E \subseteq Q \times D$, where Q is a set of documents and D a set of documents clicked for those queries. Let $w(i, j)$ be a weighting function for an edge $(i, j) \in E$, for instance it can be the number of clicks there are for a query Q_i on a document D_j. Let

$$p_{ij} = \sum_{j \in D} \frac{w(i,k)}{\deg(i)} \frac{w(k,j)}{\deg(k)} \ ,$$

now p_{ij} describes a Markov process in which the states are the elements of Q

Query-flow graphs, introduced in [8], are graph representation of the interesting knowledge about latent querying behavior: a directed edge from query q_i to query q_j means that the two queries are likely to be part of the same "search mission". Such a graph could be enriched labelling each edge with the query reformulation types types): generalization, specialization, error correction, and parallel move [9]. Modeling query reformulation types and characterizing query reformulation patterns is of extreme utility for understanding and predicting user intents, in order to assist users in retrieving more effective information. Query-flow graphs are presenting more in detail in Section 3.

A concept similar to our query flow graph, but in the context of web browsing behavior, is introduced by Levene and Loizou[21]: *"Hypertext Probabilistic Automata"* are automata where the arcs of the reachability relations are labelled with probabilities that are computed from statistical information related to the frequency that users choose to navigate trough two states. The work however is focused on browsing behavior inside a Web site and not on querying behavior. Borges and Levene later introduced an improved method for measuring the ability of a variable-length Markov model to summarize user Web navigation sessions up to a given length [11].

Query recommendation. Query recommendation is a core task for large industrial search engines. Most of the work on query recommendation is focused on measures of query similarity [32, 14] that can be used for query expansion [5] or query clustering [5, 27]. A first attempt to model the users' sequential search behavior is presented by Zhang and Nasraoui [32]: the arcs between consecutive queries in the same session are weighted by a dumping factor d, and the similarity values for non consecutive queries are calculated by multiplying the values of arcs that join them. Instead, Fonseca et al. [14] discover related queries with a method based on association rules. Each transaction in the query log is seen as a session in which a single user submits a sequence of related queries in a time interval. Their notion of session is similar to the one we use in this paper.

Baeza-Yates et al. [5] study the problem of suggesting related queries issued by other users and query expansion methods to construct artificial queries. Their technique is used to recommend queries that are related to the input query but may search for different issues. The clustering is based on a term-weight vector representation of queries, obtained from the aggregation of the term-weight vectors of the URLs clicked after the query. Wen et al. [27] also present a clustering method for query recommendation that is centered around four notions of query distance: the first notion is based on keywords or phrases of the query; the second on string matching of keywords; the third on common clicked URLs; and the fourth on the distance of the clicked documents in some pre-defined hierarchy.

Jones et al. introduced the notion of query substitution. Similar queries can be obtained by replacing the query as a whole, or by substituting constituent phrases [20]. Similar queries and phrases are derived from user query sessions, and they proposed models for query re-ranking based on the similarity of the new query to the original query.

Antonellis et al. [1] generate query rewrites in the query-click graph. The basic concept they use is similar to co-citation: two queries are related if they reference the same document. For scoring the query rewrites, they use a variant of SimRank [17], which is a generalized measure of co-citation.

White et al. [29, 30] the query rewrites observed in a query log are used to generate query recommendation. Given an input query, they generate a set of candidates containing (a) the top 100 queries that contain the original query as a sub-string, and (b) the top 100 queries which followed the input query. Then, each candidate query is then scored by multiplying its smoothed overall frequency of following the target query in the past sessions, using Laplacian smoothing.

Recommendations based on random walks. Craswell and Szummer [13] describe a method based on random walks on the query-click graph [6], that can be used to provide query recommendations as follows: given the input query, it computes the personalized PageRank [18] of all the other queries and then picks the top ones as recommendations. There are more details about this method in Section 4.3. Fuxman et al. [15] experiment with a similar approach in the context of finding related keywords for advertising.

Mei, Zhou and Church [23] instead use a computation of hitting time: assume that Q_0 is the input query: they start setting $h(Q_i, 0) = 0$ for all queries Q_i except for the original query Q_0 which has $h(Q_0, t) = 1\ \forall t \geq 0$, and then iterate the following for a fixed number m of iterations:

$$h(Q_i, t) = \sum_{j \neq i} p_{ji} h(Q_j, t-1) \ .$$

For a query Q_i, what their process computes in $h(Q_i, t)$ is the probability that a random walk arrives to node Q_i within t steps or less.

Query recommendation systems can also be personalized by taking into account the user's history. Zhang and Nasraoui [32] bias recommendations using the a user's history and introducing a "forgetting factor" which discounts older queries to favor more recent ones. A similar approach is used in [8] where a random walk with restart to the queries in the history of the user is done, preferring recent queries over older ones. As a general observation, recent works have shown that not only the previous query, but the long-term interests of users, are important for understanding his/her information need [22, 25].

3. QUERY FLOW GRAPH

The query-flow graph [8] is a graph modeling user behavioral activities and query relationships. Such a structure is obtained aggregating information contained in large query-logs in order to extract meaningful patterns. Given a query log, the nodes of the query-flow graph are all the queries contained in the log.

These query graphs are effective in helping to detect logical sessions. Identifying the logical boundaries of users search sessions is a compulsory step for understanding users intents. A further step in this direction can be performed building an accurate model for automatically classifying user query reformulations into broad classes. An automatic query transition classifier, introduced in [9], is used in the current paper to annotate each edge of the query-flow graph with one of the possible query reformulation types.

We have a graph $G = (Q, E)$, with Q the set of queries, $E \subseteq Q \times Q$ query transitions. Let $r : E \to \mathbb{N}$ represent the number of times the transition was observed in the query log *and classified automatically as being part of the same search mission*, we use as weights

$$w(i, j) = \frac{r(i, j)}{\sum_{k:(i,k) \in E} r(i, k)} \ .$$

Let $\mathcal{T} = \{\mathtt{G}, \mathtt{S}, \mathtt{C}, \mathtt{P}\}$ be the transition types generalization, specialization, error correction and parallel move as described in [9]. Let $t : E \to \mathcal{T}$ be a transition type assignment with $t(i, j)$ representing the reformulation type inferred between queries i and j.

We consider different *slices* of the original query flow-graph, extracting only the edge belonging to one or more transition types. In particular we use five different slices named Queryflow-S, Queryflow-SP, Queryflow-SC, Queryflow-SCP, Queryflow-GSPC.

Composed query graphs. Since the query-transition graph is extremely sparse, the slices we consider are even more sparse. For this reason, we experimented the possibility of *composing* two or more graphs so to increase density. By "composition" of two graphs with the same node set we mean the multiplication of the respective adjacency matrices in the semiring having multiplication as sum and maximization as product. Essentially, after the product the weight associated to an arc from q to q' is the weight of the heaviest path of length two going from q to q', where the weight of a path is the product of the weights of its component arcs.

In the experiments, we used three composed graphs: Queryflow-(S^2) (specializing twice), Queryflow-(SS^T) (specializing and then doing an inverse specialization) and Queryflow-(SG), specializing and then generalizing. The intended usage of the latter two is that they should provide a type of recommendation more varied than simple specialization.

4. EXPERIMENTAL FRAMEWORK

4.1 Dataset

Our experiments are based on the "Spring 2006 Data Asset" distributed by Microsoft Research[1]. The data consists of a query log excerpt with 15 million queries, most of them in English, sampled over one month and including a query and query-id, an anonymous session-id, a timestamp, and the results (for each result, the position on the result page and a timestamp is also provided). Part of the adult queries was extracted and provided separately: we did not use it in our experiments, though.

We encoded the data using the WebGraph framework [10] (the framework has been originally built to represent web graphs, but it turns out to be useful to represent succinctly large graphs in general) and also the high-performance hashing classes from the Sux4J project [7].

For creating the Query Flow Graph, we used the model that we trained using a different dataset—a set of query pairs (q, q'), extracted from a query log of the Yahoo! UK search engine in early 2008. These query pairs were first used to build a model [8] for segmenting users sessions into *chains*, that is, topically coherent sequences of queries by one user. In a following paper [9], we developed a model that made us able to distinguish the *type* of the transition in a chain (i.e., specialization, generalization, correction, etc.).

4.2 Recommendation using query-flow graph

The query recommendation methods are based on the probability of being at a certain node after performing a random walk over a query graph. This random walk starts in the node corresponding to the input query. At each step, the random walker either remains in the same node with probability 0.9, or follows one of the out-links with probability equal to 0.1; in the latter case, the links are followed proportionally to $w(i, j)$.

We did not use "random jumps" since, setting the random jump probability to 0.1 or 0.2, we observed a worsening in the results so we did not include it in the analysis of results. The self-loop probability was set following Crasswell

[1] `http://research.microsoft.com/users/nickcr/wscd09/`

and Szummer [13]. We increased the number of iterations until we did not see any gains; we observed that a number of iterations greater than 10 does not improve the results and we omit those results. also given the large self-loop probability a large number of iterations does not add much, anyways much of the probability stays close to the original node. We show the results for 1, 5 and 10 iterations.

We compare two different scoring methods. In the first case the queries to present to the user are chosen based on the personalized PageRank values obtained by the random walk described above, this is the "absolute" scoring method in Tables 4 and 5. An alternative scoring method ranks the results based on the ratio between the values obtained in the previous case and the PageRank values obtained by using no personalization (starting at random at any node) and fixing the random jump value to 0.15 and letting the algorithm run until convergence; this is referred to as the "relative" scoring method in the same tables.

4.3 Baseline for query recommendation

We implemented a query-recommendation system based on the method by Crasswell and Szummer [13], which uses a bipartite query-document graph. This query-document graph is defined as $G' = (Q \cup D, E')$, $E' \subseteq Q \times D$ with Q the set of documents and D the set of pages. The edges are symmetric, $(i,j) \in E \Rightarrow (j,i) \in E$. Let $c' : E \to \mathbb{N}$ be the number of clicks with $c'(i,j) = c'(j,i)$ describing the number of clicks obtained by document j when shown as a result of query i.

There are several alternatives for the transition probabilities, we used the two different weighting schemes described in [13]. The "forward' weighting scheme corresponds to following edges proportionally to the number of clicks associated to them, it uses weights

$$w_f(i,j) = \frac{c'(i,j)}{\sum_{k:(i,k)\in E'} c'(i,k)} \ .$$

The "backwards" weighting scheme uses different weights

$$w_b(i,j) = \frac{w_f(j,i)}{\sum_{k:(j,k)\in E'} w_f(j,k)} \ .$$

In the paper introducing these weights, they observe that the "backwards" weighting scheme provides better results than the "forward" weighting scheme for their task of finding relevant images for an input query. In our experimental results we observe the same, with an even greater advantage for the "backwards" weighting scheme as will be presented below.

For generating the recommendation we proceed as above, except that we used 6 or 12 iterations to do an even number of steps and end the random walk in a query and not in a document. We also did experiments with 24 iterations that did not yield improvements over 12 iterations and are omitted in the experimental section.

5. EVALUATION

This section presents the user-evaluation method and the results obtained.

5.1 Assessment method

The evaluation of the recommendations produced by the different systems was done in the following way. A set of 114 input queries having frequencies between 700 and 15,000 was selected at random; we used these frequencies limit to avoid very frequent queries (which are often navigational and for which query recommendations are not useful) or very infrequent queries (for which in this dataset there will be no recommendations). Queries were very varied in nature, examples: "grey's anatomy", "juno", "Maggie Gyllenhaal', "cnn news", and "guitar tabs". We discarded all the queries containing a domain name.

Next, we generated the top 5 recommendations for each query using each system, and pooled the results together; this yielded on average 53.4 different recommendations per query. Next, a group of 5 assessors entered a simple assessment interface where each assessor was presented a random query and then in sequence all the different recommendations for that query in random order, without knowing which system(s) produced the recommendation.

The assessor was also able to see the search engine results for the original query and the recommended query that was being evaluated. The assessor was asked if the recommendation was **useful**, **somewhat useful** or **not useful**, considering the original query. A very broad instruction was given: a useful recommendation is a query such that, if the user submits it to the search engine, it provides new results that were not available using the original query, and that agree with the inferred user intent of the original query. Of course there is a great deal of subjectivity in this assessment as the original intent is not known for sure by the assessor.

Table 1: Example assessments for query "cnn news"

Useful	Somewhat useful	Not useful
cnn world news	abc7chicagonews	CNN
msnbc news	nba scores	cnn.com
fox news	cnnfyi	verizon netmail

Table 1 shows a sample assessment for the input query "cnn news". In practice, recommendations that are considered useful are typically either specializations of parallel moves in the sense of [26], while recommendations that are considered not useful tend to be either trivial variants of the original query, or completely unrelated queries.

In total, we received 6,093 assessments distributed as per Table 2.

Table 2: Distribution of assessments, $n = 6,093$

Assessment	Probability
Useful	25.1%
Somewhat useful	11.6%
Not useful	62.1%
Can not assess	1.2%

The assessment task was described as difficult by the assessors. We measured inter-assessor agreement on 560 overlapping query-recommendation pairs that were judged by two different assessors. We considered three scenarios: A. each label is a different category; B. labels "somewhat useful" and "not useful" are together in a category; C. labels "useful" and "somewhat useful" are together in a category. Next we measured the observed agreement P_a and Cohen's Kappa statistic which compares the agreement expected by chance P_c with the observed agreement using the formula $\kappa = \frac{P_a - P_c}{1 - P_c}$.

Table 3: Inter-assessor agreement as a probability P_a and in terms of Cohen's Kappa κ, $n = 560$

	Scenario	P_a	κ
A.	Useful vs Sw.useful vs Not useful	68%	0.43
B.	Useful vs (Sw.useful or Not useful)	86%	0.46
C.	(Useful or Sw.useful) vs Not useful	77%	0.59

As shown in Table 3, the scenario C. is the best of the three and shows a moderate amount of agreement between the assessors ($\kappa = 0.59$). The relatively small level of agreement can be compared with other similarly subjective web evaluation tasks such as $\kappa = 0.85$ for web page type classification [16], $\kappa = 0.72$ for query type classification [31], $\kappa = 0.61$ for link type classification [16], $\kappa = 0.63$ for web spam classification [12], etc.

5.2 Results

Usefulness score. The *Uscore* column in Table 4 is the probability that a recommendation issued by a system is labeled as "useful" or "somewhat useful". The column concerning significance (p-value, omitted when over 0.1) contains the probability of observing a score of *Uscore* or less by chance, assuming that all the systems have the same accuracy as the top one.

Small differences in p-value for systems having the same *Uscore* are because the significance is computed considering the number of valid assessments for each system among the 114 queries evaluated, excluding the "Can not assess" label in Table 2. Lines are drawn in the table at $p = 0.1, 0.05, 0.01$. Notice that we are here testing our systems against a very strong null hypothesis, because only the top 5 recommendations are being considered, and many of them are correct; so the probability of guessing *among them* is very high, even at random.

In the recommendations generated using the query-flow graph, the score decreases as we introduce more transition types: specialization transitions seem to produce the most useful recommendations (Queryflow-S), whereas adding parallel moves (Queryflow-SP), corrections (Queryflow-SPC), and eventually generalization (Queryflow-GSPC, different at $p = 0.06$) results in less useful recommendations.

The "absolute" scoring method works better than the "relative" scoring method for the queryflow-based recommendations at a significance of $p = 0.04$, and doing multiple iterations instead of only one (which corresponds to taking the maximum) is better at $p = 0.06$.

We also added a system named just "Queryflow" in Table 4, without including any slice name: in this system the weights are computed over all transitions, independently of whether they were part of the same mission or not. This is worse than the systems that selects only specializations and counts only over transitions in the same mission at $p = 0.01$.

The recommendations based on the baseline (query-document graph) have either the same performance as recommendations using Queryflow-S, or a lower performance at a significance of $p = 0.07$. In this case, the "backwards" weighting scheme performs much better than the "forwards" weighting scheme at $p < 0.01$. This was already noticed in [13]: the gap, in our case, is even larger.

Table 4: Usefulness score for each system: probability that a recommendation issued by the system is useful or somewhat useful

Uscore	p-value	System	Iter.	Scoring
0.58		Queryflow-S	10	Abs.
0.58		Queryflow-S	5	Abs.
0.57		Queryflow-SP	1	Abs.
0.56		Queryflow-SP	10	Abs.
0.56		Queryflow-SP	5	Abs.
0.55	0.10	Queryflow-SPC	1	Abs.
0.55	0.06	Queryflow-GSPC	1	Abs.
0.55	0.06	Queryflow-S	1	Abs.
0.55	0.07	Queryflow-SPC	5	Abs.
0.55	0.06	Queryflow-SPC	10	Abs.
0.55	0.07	QueryDocument-Bwd	6	Rel.
0.55	0.06	QueryDocument-Bwd	12	Rel.
0.54	0.04	Queryflow-S	1	Rel.
0.54	0.03	Queryflow-S	10	Rel.
0.54	0.02	Queryflow-SC	5	Abs.
0.54	0.02	Queryflow-S	5	Rel.
0.54	0.02	QueryDocument-Bwd	2	Rel.
0.54	0.04	QueryDocument-Bwd	12	Abs.
0.54	0.02	QueryDocument-Bwd	6	Abs.
0.53	0.01	Queryflow	1	Abs.
0.53	0.01	Queryflow-GSPC	5	Abs.
0.53	0.01	Queryflow-GSPC	10	Abs.
0.53	0.01	Queryflow-SC	10	Abs.
0.52	$< .01$	Queryflow	5	Abs.
0.52	$< .01$	QueryDocument-Bwd	2	Abs.
0.52	$< .01$	Queryflow-SC	1	Abs.
0.52	$< .01$	Queryflow-SC	1	Rel.
0.51	$< .01$	Queryflow	10	Abs.
0.51	$< .01$	Queryflow-SC	10	Rel.
0.51	$< .01$	Queryflow-SC	5	Rel.
0.47	$< .01$	Queryflow-SP	1	Rel.
0.47	$< .01$	Queryflow-SP	10	Rel.
0.47	$< .01$	Queryflow-SP	5	Rel.
0.45	$< .01$	Queryflow-SPC	10	Rel.
0.45	$< .01$	Queryflow-SPC	1	Rel.
0.45	$< .01$	Queryflow-SPC	5	Rel.
0.44	$< .01$	Queryflow	10	Rel.
0.44	$< .01$	Queryflow	1	Rel.
0.44	$< .01$	Queryflow	5	Rel.
0.44	$< .01$	Queryflow-GSPC	10	Rel.
0.43	$< .01$	Queryflow-GSPC	1	Rel.
0.43	$< .01$	Queryflow-GSPC	5	Rel.
0.39	$< .01$	Queryflow-(S^2)	1	Rel.
0.39	$< .01$	Queryflow-(S^2)	10	Rel.
0.39	$< .01$	Queryflow-(S^2)	1	Abs.
0.38	$< .01$	Queryflow-(S^2)	10	Abs.
0.32	$< .01$	QueryDocument-Fwd	12	Abs.
0.32	$< .01$	QueryDocument-Fwd	6	Abs.
0.30	$< .01$	QueryDocument-Fwd	2	Abs.
0.28	$< .01$	Queryflow-(SG)	10	Rel.
0.28	$< .01$	QueryDocument-Fwd	6	Rel.
0.28	$< .01$	QueryDocument-Fwd	12	Rel.
0.28	$< .01$	Queryflow-(SG)	10	Abs.
0.27	$< .01$	QueryDocument-Fwd	2	Rel.
0.23	$< .01$	Queryflow-(SS^T)	10	Abs.
0.23	$< .01$	Queryflow-(SS^T)	10	Rel.

Figure 1 is a chart of the best performing variant of each system.

Diversity score. Next we computed a measure of diversity in the resulting set. This is done by taking each sampled query, and each recommendation labeled as useful or somewhat useful, and issuing that recommended query to a search engine. Given that we are taking the top-5 recommendations per system, this generates a maximum of 25 URLs. The average *Dscore* in Table 5 is the average number of distinct URLs in this multiset across the 114 queries evaluated which were not present in the result set for the original query.

Significance is computed using the individual score (0 to 25) obtained by each system for each of the 114 assessed queries; we assume scores have a normal distribution and compute the probability of observing the scores we observe or less, assuming that all systems have the same performance as the top system (using a one-sided t-test). Lines are drawn in the table at $p = 0.1, 0.05, 0.01$. We observe a change in the relative position of different systems in the top half of the table with respect to Table 4, indicating that this measure is different from the measure based purely on the labels associated to the recommended queries.

6. CONCLUSIONS

The query-flow graph annotated with query reformulation types, can be used to generate query recommendations matching the ones obtained using query-click graphs. This means that the information contained in the annotated query-flow graph about consecutive queries in a session is as useful for this task as the user's clicks; given that both data sources are independent, recommendations produced by a composition of both methods are worth to be investigated as future work.

When using the query-flow graph, we have found that it is important to discard edges between queries in different chains, even if they are frequent transitions. Also, allowing only certain reformulation types (e.g.: only specializations, or only specializations and parallel moves) is better than using the entire graph. Finally, doing a few iterations is better than doing one iteration (this is picking directly the node connected by the edge of highest weight), and more than 10 iterations in our setting did not add precision to the results.

With respect to the evaluation methodology, the amount of user assessments we obtained, over 6,000 in total, was enough for drawing some conclusions but not enough for others. A more fine-grained study would probably require a substantially larger number of evaluations and assessors.

Acknowledgments: Aristides Gionis for helpful comments, and Marco Rosa for help in the assessment.

Table 5: Diversity score of recommended queries: distinct documents among the top-5 results for the top-5 useful or somewhat useful recommendations

Dscore	p-value	System	Iter.	Scoring
13.49		Queryflow-S	10	Abs.
13.44		Queryflow-S	5	Abs.
13.20		Queryflow-SP	1	Abs.
13.04		Queryflow-SP	10	Abs.
12.99		Queryflow-SP	5	Abs.
12.84		Queryflow-SCP	1	Abs.
12.73		Queryflow-SCP	5	Abs.
12.70		Queryflow-GSPC	1	Abs.
12.70		Queryflow-SCP	10	Abs.
12.52		Queryflow-S	1	Rel.
12.42		Queryflow-S	1	Abs.
12.40		Queryflow-S	10	Rel.
12.38		Queryflow	1	Abs.
12.38		Queryflow-GSPC	5	Abs.
12.37		Queryflow-S	5	Rel.
12.33		Queryflow-SC	5	Abs.
12.33		Queryflow-GSPC	10	Abs.
12.28	0.10	QueryDocument-Bwd	6	Rel.
12.25	0.10	Queryflow-SC	10	Abs.
12.21	0.08	QueryDocument-Bwd	12	Rel.
12.21	0.08	QueryDocument-Bwd	12	Abs.
12.16	0.08	QueryDocument-Bwd	2	Rel.
12.11	0.06	QueryDocument-Bwd	6	Abs.
12.01	0.08	Queryflow	5	Abs.
11.97	0.05	QueryDocument-Bwd	2	Abs.
11.92	0.05	Queryflow-SC	1	Rel.
11.89	0.07	Queryflow	10	Abs.
11.77	0.04	Queryflow-SC	1	Abs.
11.64	0.03	Queryflow-SC	10	Rel.
11.60	0.03	Queryflow-SC	5	Rel.
11.13	0.01	Queryflow-SP	1	Rel.
11.04	0.01	Queryflow-SP	5	Rel.
10.94	0.01	Queryflow-SP	10	Rel.
10.62	$< .01$	Queryflow-SPC	1	Rel.
10.61	$< .01$	Queryflow-SPC	5	Rel.
10.56	$< .01$	Queryflow-SPC	10	Rel.
10.43	$< .01$	Queryflow	10	Rel.
10.39	$< .01$	Queryflow	1	Rel.
10.36	$< .01$	Queryflow	5	Rel.
10.28	$< .01$	Queryflow-GSPC	10	Rel.
10.25	$< .01$	Queryflow-GSPC	1	Rel.
10.08	$< .01$	Queryflow-GSPC	5	Rel.
9.25	$< .01$	Queryflow-(S^2)	1	Rel.
9.21	$< .01$	Queryflow-(S^2)	10	Rel.
9.17	$< .01$	Queryflow-(S^2)	1	Abs.
9.00	$< .01$	Queryflow-(S^2)	10	Abs.
6.68	$< .01$	Queryflow-(SG)	10	Abs.
6.63	$< .01$	Queryflow-(SG)	10	Rel.
5.75	$< .01$	QueryDocument-Fwd	12	Abs.
5.71	$< .01$	QueryDocument-Fwd	6	Abs.
5.69	$< .01$	Queryflow-(SS^T)	10	Abs.
5.61	$< .01$	Queryflow-(SS^T)	10	Rel.
5.49	$< .01$	QueryDocument-Fwd	2	Abs.
5.05	$< .01$	QueryDocument-Fwd	6	Rel.
5.04	$< .01$	QueryDocument-Fwd	12	Rel.
4.83	$< .01$	QueryDocument-Fwd	2	Rel.

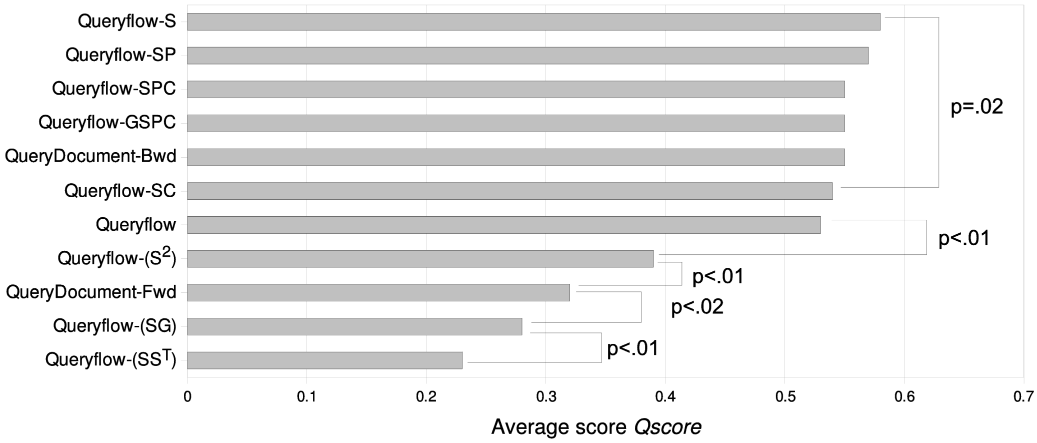

Figure 1: Usefulness scores, best variant per system

7. REFERENCES

[1] Antonellis, I., Garcia-Molina, H., and Chang, C.-C. Simrank++: Query rewriting through link analysis of the click graph. In *Proceedings of VLDB* (Dec 2008), pp. 408–421.

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

[3] Baeza-yates, R., Hurtado, C., and Mendoza, M. Query recommendation using query logs in search engines. In *In International Workshop on Clustering Information over the Web (ClustWeb, in conjunction with EDBT), Creete* (2004), Springer, pp. 588–596.

[4] Baeza-Yates, R., and Tiberi, A. Extracting semantic relations from query logs. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2007), ACM Press, pp. 76–85.

[5] Baeza-Yates, R. A., Hurtado, C. A., and Mendoza, M. Query recommendation using query logs in search engines. In *EDBT Workshops* (2004), vol. 3268 of *LNCS*, Springer, pp. 588–596.

[6] Beeferman, D., and Berger, A. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), ACM Press, pp. 407–416.

[7] Belazzougui, D., Boldi, P., Pagh, R., and Vigna, S. Theory and practise of monotone minimal perfect hashing. In *ALENEX 09: Algorithm Engineering and Experimentes* (2009), Lecture Notes in Computer Science, Springer–Verlag.

[8] Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., and Vigna, S. The query-flow graph: Model and applications. In *Proceedings of the ACM 17th Conference on Information and Knowledge Management (CIKM 2008).*

[9] Boldi, P., Bonchi, F., Castillo, C., and Vigna, S. From "dango" to "japanese cakes": Query reformulation models and patterns. Submitted for publication, 2008.

[10] Boldi, P., and Vigna, S. The WebGraph framework I: Compression techniques. In *Proc. of the Thirteenth International World Wide Web Conference (WWW 2004)* (Manhattan, USA, 2004), ACM Press, pp. 595–601.

[11] Borges, J., and Levene, M. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Trans. Knowl. Data Eng. 19*, 4 (2007), 441–452.

[12] Castillo, C., Donato, D., Becchetti, L., Boldi, P., Leonardi, S., Santini, M., and Vigna, S. A reference collection for web spam. *SIGIR Forum 40*, 2 (December 2006), 11–24.

[13] Craswell, N., and Szummer, M. Random walks on the click graph. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007), ACM Press, pp. 239–246.

[14] Fonseca, B. M., Golgher, P. B., de Moura, E. S., and Ziviani, N. Using association rules to discover search engines related queries. In *LA-WEB '03: Proceedings of the First Latin American Web Congress* (Washington, DC, USA, 2003), IEEE Computer Society.

[15] Fuxman, A., Tsaparas, P., Achan, K., and Agrawal, R. Using the wisdom of the crowds for keyword generation. In *WWW '08: Proceeding of the 17th international conference on World Wide Web* (New York, NY, USA, 2008), ACM, pp. 61–70.

[16] Haas, S. W., and Grams, E. S. Page and link classifications: connecting diverse resources. In *DL '98: Proceedings of the third ACM conference on Digital libraries* (New York, NY, USA, 1998), ACM, pp. 99–107.

[17] Jeh, G., and Widom, J. Simrank: a measure of structural-context similarity. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (New York, NY, USA, 2002), ACM Press, pp. 538–543.

[18] Jeh, G., and Widom, J. Scaling personalized web search. In *WWW '03: Proceedings of the 12th*

international conference on World Wide Web (New York, NY, USA, 2003), ACM Press, pp. 271–279.

[19] Jones, R., and Klinkner, K. L. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Conference on Information and Knowledge Management (CIKM)* (October 2008), ACM Press.

[20] Jones, R., Rey, B., Madani, O., and Greiner, W. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006* (2006), pp. 387–396.

[21] Levene, M., and Loizou, G. A probabilistic approach to navigation in hypertext. *Inf. Sci. 114*, 1-4 (1999), 165–186.

[22] Luxenburger, J., Elbassuoni, S., and Weikum, G. Matching task profiles and user needs in personalized web search. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining* (New York, NY, USA, 2008), ACM, pp. 689–698.

[23] Mei, Q., Zhou, D., and Church, K. Query suggestion using hitting time. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge mining* (New York, NY, USA, 2008), ACM, pp. 469–478.

[24] Radlinski, F., and Joachims, T. Query chains: learning to rank from implicit feedback. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining* (New York, NY, USA, 2005), ACM Press, pp. 239–248.

[25] Richardson, M. Learning about the world through long-term query logs. *ACM Trans. Web 2*, 4 (2008), 1–27.

[26] Rieh, S. Y., and Xie, H. Analysis of multiple query reformulations on the web: the interactive information retrieval context. *Inf. Process. Manage. 42*, 3 (2006), 751–768.

[27] Wen, J.-R., Nie, J.-Y., and Zhang, H.-J. Clustering user queries of a search engine. In *WWW '01: Proceedings of the 10th international conference on World Wide Web* (New York, NY, USA, 2001), ACM, pp. 162–168.

[28] Wen, J.-R., Nie, J.-Y., Zhang, H.-J., and Zhang, H.-J. Clustering user queries of a search engine. In *Proceedings of the 10th int. conf. on World Wide Web (WWW'01)*.

[29] White, R. W., Bilenko, M., and Cucerzan, S. Studying the use of popular destinations to enhance web search interaction. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007).

[30] White, R. W., Bilenko, M., and Cucerzan, S. Leveraging popular destinations to enhance web search interaction. *ACM Trans. Web 2*, 3 (July 2008), 1–30.

[31] White, R. W., Clarke, C. L. A., and Cucerzan, S. Comparing query logs and pseudo-relevance feedbackfor web-search query refinement. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (New York, NY, USA, 2007), ACM, pp. 831–832.

[32] Zhang, Z., and Nasraoui, O. Mining search engine query logs for query recommendation. In *WWW '06: Proceedings of the 15th international conference on World Wide Web* (New York, NY, USA, 2006), ACM, pp. 1039–1040.

Key references: [8, 13].