Learning from Users' Querying Experience on Intranets

Ibrahim Adeyanju[*]
IDEAS Research Institute
The Robert Gordon University
St Andrews Street, Aberdeen
AB25 1HG, Scotland, UK
i.adeyanju@rgu.ac.uk

Dawei Song
IDEAS Research Institute
The Robert Gordon University
St Andrews Street, Aberdeen
AB25 1HG, Scotland, UK
d.song@rgu.ac.uk

M-Dyaa Albakour
School of Computer Science
and Electronic Engineering
University of Essex, Wivenhoe
Park,Colchester C04 3SQ, UK
malbak@essex.ac.uk

Udo Kruschwitz
School of Computer Science
and Electronic Engineering
University of Essex, Wivenhoe
Park,Colchester C04 3SQ, UK
udo@essex.ac.uk

Anne De Roeck
Centre for Research in
Computing
Open University, Walton Hall
Milton Keynes MK7 6AA, UK
a.deroeck@open.ac.uk

Maria Fasli
School of Computer Science
and Electronic Engineering
University of Essex, Wivenhoe
Park,Colchester C04 3SQ, UK
mfasli@essex.ac.uk

ABSTRACT

Query recommendation is becoming a common feature of web search engines especially those for Intranets where the context is more restrictive. This is because of its utility for supporting users to find relevant information in less time by using the most suitable query terms. Selection of queries for recommendation is typically done by mining web documents or search logs of previous users. We propose the integration of these approaches by combining two models namely the concept hierarchy, typically built from an Intranet's documents, and the query flow graph, typically built from search logs. However, we build our concept hierarchy model from terms extracted from a subset (training set) of search logs since these are more representative of the user view of the domain than any concepts extracted from the collection. We then continually adapt the model by incorporating query refinements from another subset (test set) of the user search logs. This process implies learning from or reusing previous users' querying experience to recommend queries for a new but similar user query. The adaptation weights are extracted from a query flow graph built with the same logs. We evaluated our hybrid model using documents crawled from the Intranet of an academic institution and its search logs. The hybrid model was then compared to a concept hierarchy model and query flow graph built from the same collection and search logs respectively. We also tested various strategies for combining information in the search logs with respect to the frequency of clicked documents after query refinement. Our hybrid model significantly outperformed the concept hierarchy model and query flow graph when tested over two different periods of the academic year. We intend to further validate our experiments with documents and search logs from another institution and devise better strategies for selecting queries for recommendation from the hybrid model.

[*]Corresponding author

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval; I.2.8 [**Computing Methodologies**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Experimentation, Performance

Keywords

Query Reformulation, Intranet, Users' Search Experience

1. INTRODUCTION

Query recommendation is becoming a common feature of web search engines because of the need to support users to find relevant information quickly. Such recommendations can come in form of suggestions for query completion at the start of a user search session (e.g. Google, Bing and Yahoo search engines). However, it is also typical to recommend suitable refinements after the initial query in case the user does not find relevant information in the ranked list of retrieved documents. Visual representation of related concepts to the query and topics mined from the retrieved documents have also been proposed as a query recommendation tool [19, 13]. These user-friendly features have led to continuous changes in the interface design of search engines and are expected to make it easier and quicker for a user to satisfy his/her information need. In this paper, we focus on the generation of suitable query refinements once a user has entered an initial query.

Several models have been proposed for suggesting query refinements. Most have exploited the knowledge and information in the documents collection or previous search logs but not both. We propose the combination of these two knowledge sources by enhancing a query recommendation model typically built from a documents collection with user interactions derived from previous search logs. The concept hierarchy model [19] is our choice for mining suggestions from the documents collection due to its simple but effective and intuitive approach to query recommendation. However, we use terms extracted from previous search logs as concepts in the model since they are more representative of the users' view of the domain. With respect to mining from search logs, we chose the query flow graph model [5], a state-of-the-art adaptive query recommendation model. These two models are combined into a hybrid model using an approach that utilises the best components of both models. We evaluate the effectiveness of the hybrid model using an automatic evaluation method (AutoEval) which is based on search logs data [2] by experimenting with search logs obtained over two different periods of the academic year by a higher institution's search engine (University of Essex, http://www.essex.ac.uk). We also compare the hybrid model to the two individual models.

Other sections in this paper are as follows. Related works on query recommendation and utilization of user logs to improve information seeking are discussed in Section 2. We then give in-depth details of the concept hierarchy model and how we used it to generate query refinement terms, the query flow graph, and the amalgamation of both models in Sections 3, 4 and 5 respectively. This is followed by experimental set-up in Section 6. Evaluation results are discussed in Section 7 and we conclude in Section 8 by outlining our main contributions and plans for future work.

2. RELATED WORK

A lot of modern web search engines now offer query refinements to users. Determining the best terms to recommend remains a challenge as this depends on the combination of several factors. However, studies have indicated that users prefer having suggestions regardless of the usefulness of the suggestions [18, 20, 21]. Knowledge of the query's context, which can be a function of the user's previous search queries, is a crucial factor when suggesting terms to help a user find specific information in less time. The two main resources available for query recommender systems are the documents collection (including anchor logs) [22, 19, 8, 13, 16, 15, 11] and search logs [3, 14, 10, 4, 12, 5, 6].

The approach in techniques that utilise the documents collection is to discover relationships between terms using all documents (global) [7, 11] or those retrieved as relevant to a query (local) [19, 8]. Terms determined as related to the query are then recommended for query refinement or automatically used for query expansion. These techniques are therefore less dependent on any user feedback (implicit or explicit) but require updates to their recommendations when there is a significant change in the underlying collection. This method is most useful for new search engines with little or no search logs available.

Query search logs on the other hand provide real user experience which can be mined to discover useful local or global patterns. Query recommendations can be said to be local if they are based only on each user's previous searches [14] while global or collective recommendations will be derived from a group of users or all users [4, 5]. The main drawback is that logs have to be collected over a period before such techniques can be applied since data mined from one search engine might not be easily applicable to another.

We explore the potential of integrating the query flow graph built from search logs with the concept hierarchy built from documents, but using terms from search logs, into a hybrid model for deriving query recommendation suggestions.

3. CONCEPT HIERARCHY MODEL

The concept hierarchy model [19] was proposed to assist users to browse the top documents returned by a search engine by showing the topic structure in the retrieved documents. A hierarchical tree, where specialised concepts or terms are subsumed by more generic ones, is generated automatically using an unsupervised algorithm based on a statistical co-occurrence measure. Hence, the model is also known as the **S**ubsumption **H**ierarchy for **Re**sult **C**lustering (SHReC). We use SHReC from now onwards to refer to the concept hierarchy model in this paper. This model can easily be utilised for suggesting terms for query refinements by showing a sub-tree from the initial hierarchy encapsulating the given query[13]. A more intuitive way to utilise the model for query refinement is to transform terms in the sub-tree encapsulating a query into a ranked list based on strength and closeness of their links to the query in the tree.

Although SHReC was introduced for clustering retrieval results, it can easily be applied to the entire document collection when the collection is relatively small and changes less frequently (e.g. collection from an Intranet). In this scenario, the topic structure generated by SHReC might map onto a pre-defined structure in the organisation. For example, a SHReC model generated from documents in a higher institution's Intranet might show the relationship between lecturers and the courses they teach, research students and their supervisors, or academic staff and their research interests. A common feature of Intranet searches is that identical queries are more likely to be related to the same user information need since the context of any query search is more limited than in an Internet search. Therefore, previous user interactions can be used to improve a new user search by suggesting previous users' query refinements terms from previous sessions with identical or similar queries.

3.1 Building SHReC with search log queries

The SHReC model is automatically derived as a hierarchical organization of concepts from a set of documents without using training data[19]. The basic idea in a building a SHReC model is to use term co-occurrence to create a subsumption hierarchical tree. The generality or specificity of a term (or concept) in the model is determined mainly by its document frequency. The more documents a term appears in, the more general it is assumed to be. The conditions for subsumption conditions are therefore dependent on document frequencies. It should be noted that a term as used in the SHReC model does not necessarily mean a keyword. A 'term' is an entire query which can be formulated as a keyword, phrase (n-grams) or even a clause/sentence. A term 'x' is said to subsume another term 'y' ideally when Equation 1 is met.

$$P(x \mid y) = 1 \land P(y \mid x) < 1. \tag{1}$$

In other words, $df(x) > df(y)$ and $co\_df(x, y) = df(x \land y) = df(y)$ where df is a function that returns the document frequency of a given term and $co\_df(x, y)$ returns the co-occurrence document frequency between two terms.

However, the subsumption rule can be relaxed as shown in Equation 2 in order to allow few occurrences of term 'y' with other terms other than 'x'. This implies that $df(x) > df(y)$ and $co\_df(x, y)/df(y) = df(x \land y)/df(y) \geq \alpha$. The original

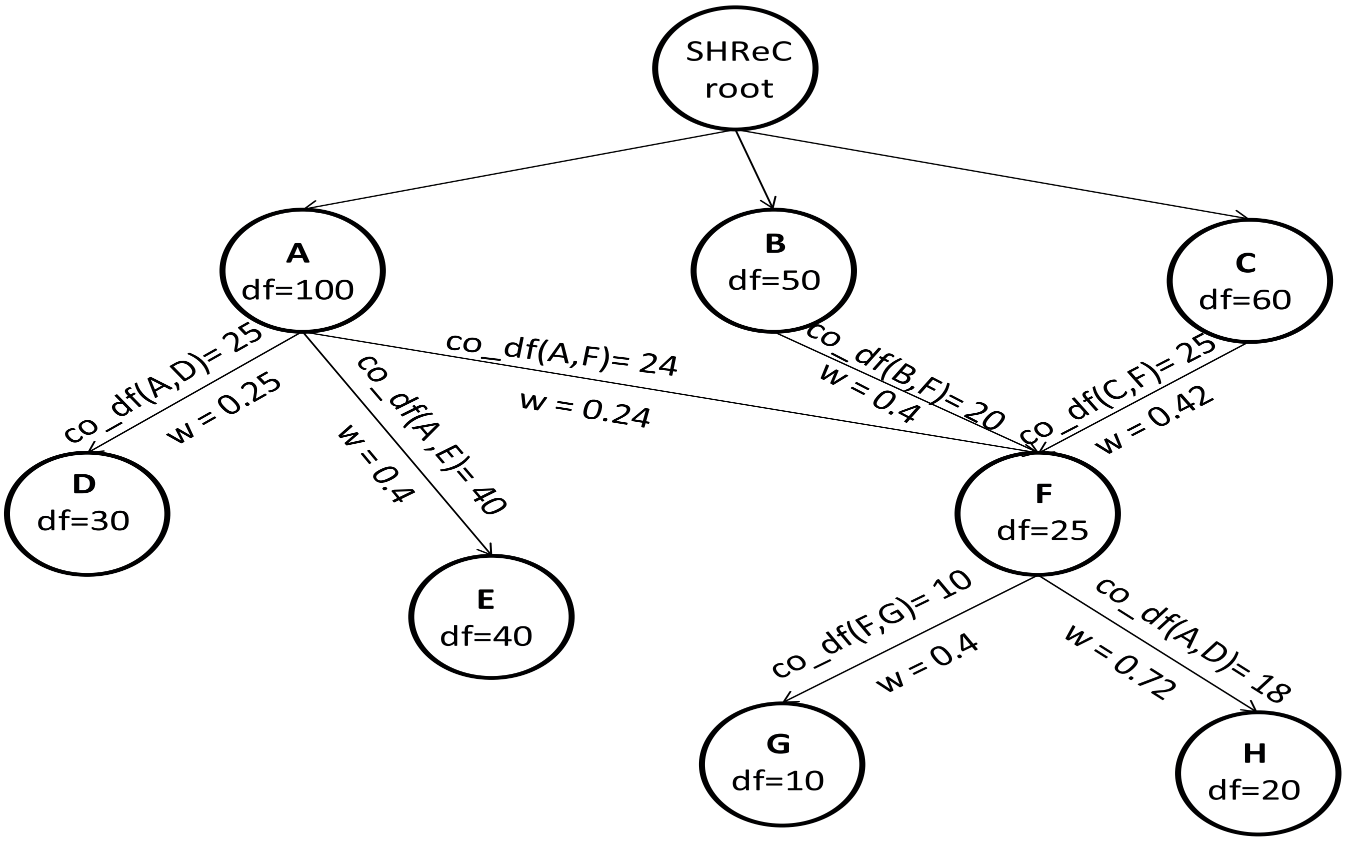

Figure 1: An example SHReC model

SHReC authors chose $\alpha = 0.8$ through empirical studies. We also used the same α-value in our experiments.

$$P(x \mid y) \geq \alpha \wedge P(y \mid x) < 1 \quad (2)$$

where α indicates whether the subsumption is complete ($\alpha = 1$) or partial ($\alpha < 1$).

Although the original SHReC model was built from top documents returned by a search engine given a query, we extended this to build a SHReC model from an entire Intranet collection but using terms from search logs. Using log terms as candidates implies that the concepts in the SHReC model would be of various length and are likely to be repeated by other users seeking similar information. Figure 1 shows a sample SHReC model for better illustration. The subsumption between 'F' and 'G' is complete ($\alpha = 1$) since the co-occurrence document frequency is same as the document frequency of 'G'. This is unlike the partial subsumption ($\alpha = 0.9$) between 'F' and 'H'.

3.2 SHReC for Query Recommendation

We employ the SHReC model for suggesting terms that should be useful for query refinement in order to satisfy a user's information need. This is done by finding a given query in the model and suggesting more generalised (immediate ancestor) and specialised (immediate descendants) terms related to the query in the hierarchy. These terms are ranked based on the combination of two weighting parameters on the edge connecting them to the query and can easily be derived from the SHReC model. The combination is a simple multiplication of the two weights as shown in Equation 3 with the importance of each weight explained afterwards.

$$w = w1 * w2 = co\_df(x, y)/df(x), \text{ for 'x' subsumes 'y'} \quad (3)$$

$$\text{if 'x' subsumes 'y', then } w1 = co\_df(x, y)/df(y) \quad (4)$$

$$\text{if 'x' subsumes 'y', then } w2 = df(y)/df(x) \quad (5)$$

i Subsumption ratio denoted as $w1$: This is the default weight on any edge connecting two terms in the hierarchy. It is the quotient of the co-occurrence document frequency and the parent's document frequency and is used to determine subsumption between terms in the hierarchy (see Equation 2). The formula for this weight is given in Equation 4. Thus, $df(y)$ is the document frequency of the query for when more generalised terms are recommended from the SHReC model while $co\_df(x, y)$ is the co-occurrence document frequency between 'x' and 'y'. However, $df(y)$ becomes the document frequency of any recommended terms when extracted from below the query in the hierarchy since they are descendants of the query. This weight is restricted by the threshold α used in creating the SHReC graph; $\alpha = 0.8$ implies that $w1$ is between 0.8 and 1.0 for all edges in the model. In Figure 1, weight $w1$ for the link between terms 'A' and 'D' will be computed as 0.83 ($\frac{25}{30}$).

ii Document Frequency ratio denoted as $w2$: This is the ratio of the document frequencies of a suggested term and that of the given query. Equation 5 encodes this information. The value at the numerator or denominator of the equation will depend on the recommended term's relationship (ascendant/ descendant) to the query. In other words, 'x' in Equation 5 will represent the query for its descendant terms but 'y' will represent the query for ascendant terms from the model in the recommended refinement list. The idea here is that the closer the document frequencies of two terms linked in a subsumption, the more useful they are as a refinement for one another. Weight $w2$ for the link between terms 'A' and 'D' in Figure 1 will be computed as 0.3 ($\frac{30}{100}$).

4. QUERY FLOW GRAPH

Query flow graphs (QFG) [5] have been successfully applied to mine query suggestions from search logs. The query flow graph G_{qf} is a directed graph $G_{qf} = (V, E, w_{qf})$ where:

- V is a set of nodes containing all the distinct queries submitted to the search engine and two special nodes s and t representing a *start state* and a *terminate state*;
- $E \subseteq V \times V$ is the set of directed edges;
- $w_{qf} : E \to (0..1]$ is a weighting function that assigns to every pair of queries $(q, q') \in E$ a weight $w_{qf}(q, q')$.

The graph can be built from search logs by creating an edge between two queries q, q' if there is one session in the logs in which q and q' are consecutive. A session is simply defined as a sequence of queries submitted by one particular user within a specific time limit. The weighting function of the edges w_{qf} depends on the application. The original authors of QFG [5] developed a machine learning model that assigns to each edge on the graph a probability that the queries on both ends of the edge are part of the same chain. The chain is defined as a topically coherent sequence of queries of one user. This probability is then used to eliminate less probable edges by specifying some threshold. For the remaining edges the weight $w_{qf}(q, q')$ is calculated as:

$$w_{qf}(q, q') = \frac{freq(q, q')}{\Sigma_{r \in R_q} freq(q, r)} \tag{6}$$

Where:

- $freq(q, q')$ is the number of times query q is followed by the query q'.
- R_q is the set of all reformulations of query q in the logs.

Note that the weights are normalised so that total weights of the outgoing edges of any node is equal to 1.

4.1 Enriching QFG with Click data

We extended the query flow graph model with click data [1]. The intuition is to use implicit feedback in the form of click-through data left by users when they refine their queries. This has been shown to be a good form of implicit feedback [9]. We consider the number of clicked documents by a user after submitting a query as an indication of the results' usefulness. This is line with previous work on evaluating search engines with click-through data [17].

Let $\phi(q, q') = \{\varphi_0(q, q'), \varphi_1(q, q'), \varphi_2(q, q'), ..\}$ be an array of the frequencies of the reformulation (q, q'), where $\varphi_k(q, q')$ is the number of the times the query q is followed by the query q' and the user has clicked k (and only k) documents on the result list presented to the user after submitting query q'. We aggregate over all users here.

We modify the weighting function in Equation 6 to incorporate the click information as follows

$$w_{qf}(q, q') = \frac{\Sigma_i C_i . \varphi_i(q, q')}{\Sigma_{r \in R_q} \Sigma_i C_i . \varphi_i(q, r)} \tag{7}$$

Where C is an array of co-efficient factors for each band of click counts. Choosing different values for C_i allows us to differentiate between queries that resulted in more or fewer clicks. For example, queries which result in a single click might be interpreted as more important than the ones which resulted in no clicks or more than one click as the single click may be an indication of quickly finding the document required by a user. We investigated how different values of the co-efficient C_i affect the quality of the query recommendations. Note that the weighting function of the standard graph in Equation 6 is the special case where $C_0 = C_1 = C_2 = .. = 1$.

4.2 Query Recommendations with QFG

Query recommendation is the problem of finding for a given query q relevant query suggestions. If we want to recommend only a single query, then we try to identify the "most important" query q'. The query flow graph can be used for this purpose by ranking related nodes in the graph according to some measure which indicates how reachable they are from the given node (query). Boldi *et al.* [5] proposed to use graph random walks for this purpose and reported the most promising results by using a measure which combines relative random walk scores and absolute scores. This measure is

$$\overline{s}_q(q') = \frac{s_q(q')}{\sqrt{r(q')}} \tag{8}$$

where:

- $s_q(q')$ is the random walk score relative to q i.e. the one computed with a preference vector for query q.
- $r(q')$ is the absolute random walk score of q' i.e. the one computed with a uniform preference vector.

Another approach to suggesting queries from the query flow graph is to utilise all queries that occur in nodes that have a connecting edge coming out of the current query. These nodes will contain refinement terms used after the current query as found in the search logs and can be ranked based on their normalised frequency of occurrence or weighting function as shown in Equations 6 and 7. We adopted this second approach for query recommendation with the QFG in our experiments as it seem more intuitive and better comparable to our SHReC which only makes suggestions from term nodes immediately linked to a current query rather than all nodes in its graph. Also, we observed that the topmost recommendations from the random walk measure are identical to this approach, which recommends only queries whose nodes are connected to edges coming out from the current query node.

5. HYBRID MODEL

We propose an amalgamation of SHReC (the concept hierarchy) and query flow graph (QFG) into a hybrid model which we call Query Flow SHReC (QF_SHReC). This is because all concepts in the new model will be extracted from query logs. However, the subsumptions in the initial SHReC are still mined from the document collection but using query terms from the search logs as concepts in the model. This way, we are able to harness and mine suggestions from both the document collection as well as the search logs. We start with an initial SHReC model using the process described in Section 3 and then continuously adapt the SHReC graph using terms and weights from the QFG. The QFG is also updated for all pairs of query refinements found at specified periodic intervals (e.g. every week) to minimise its computational cost. The amalgamation of SHReC and QFG models into QF_SHReC goes thus:

1. Normalise weights w for each edge in SHReC and initialise this as QF_SHReC: We normalise so that the total weights of all edges coming out of a term node sums up to 1. When new terms are added to the model, we also ensure that their initial weights are between 0 and 1. This normalisation process ensures that the weights of newly added terms are comparable to those already in the hierarchy. However, the weights on the terms from the query logs via the QFG are likely to be higher than most weights in hierarchy since the terms used for their normalisation should be less compared to those found in SHReC. Therefore, terms from the QFG will be ranked higher during refinement recommendation as we regard them as better since they come from real users' experience.

2. Each unique query refinement pair found in the query logs between the periodic interval used for adaptation is added to the hierarchy. We cater for each possible scenario using the following rules assuming term 'x' is refined to 'y' in the query logs.

   (a) Both x and y exist in QF_SHReC model with x subsuming y or vice versa: We update the weights between them in the model by adding the weight calculated from the query logs as found in the QFG to normalised weights from SHReC.

   (b) x exists in QF_SHReC but y does not: We add y as a descendant of x in the model with the calculated weights from QFG.

   (c) y exists in QF_SHReC but x does not: We add x as an ascendant of y in the model with the calculated weights from QFG.

   (d) Both x and y do not exist in QF_SHReC: We add them to the model with x subsuming y and using weights computed from logs via QFG.

3. Repeat step 2 above for all query refinement pairs found in the logs during the periodic interval in use.

Figure 2 illustrate the first step of the learning and adaptation process using a sub-tree from Figure 1. The sum of the normalised weights (w') from node 'A' equals 1. We then exemplify steps 2 & 3 of the adaptation process in Figure 3. Here, the refinements from the user logs during the adaptation period are assumed to be the three shown in the rectangle at the top-right corner of the figure with the weights ($w_{qf}(x, y)$) calculated as would be in a QFG having just the three refinements. The effect of the adaptation is that the link to 'F' is given more weights than others because this refinement is found in both the logs and SHReC model (Step 2(a) on updating weights). The new weights are computed as $w_{adapted}(x, y) \leftarrow w'(x, y) + w_{qf}(x, y)$, where $w' = 0$ for those links that are not in the initial SHReC created from only the documents collection. New terms could also have bigger weights than those that already exist in the graph. For example, the weight on the link between 'Q' with 'A' in Figure 3 is greater than that between 'D' and 'A' though 'D' existed from the initial SHReC. Therefore, the reuse of refinements from previous logs might change the ranking of terms that were in the initial SHReC.

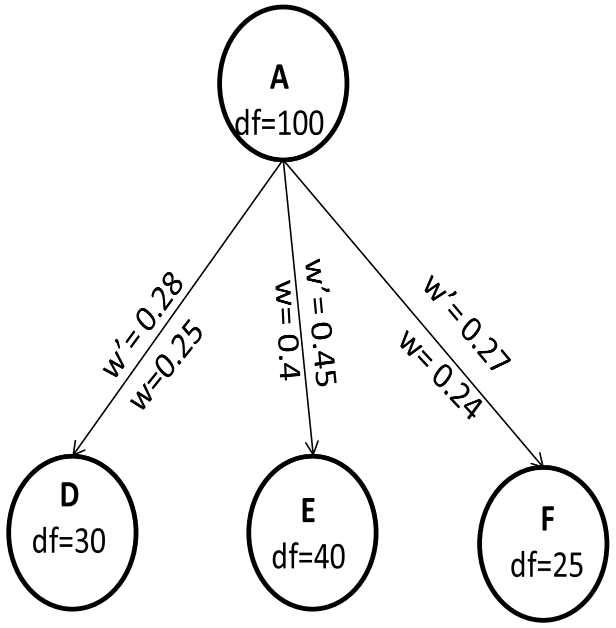

Figure 2: Normalising a SHReC model

6. EXPERIMENTAL SETUP

We aim to improve a user's search experience by recommending refinement terms related to a current query. The expectation is that the refined query will result in more relevant results than the current query thereby making it easier and quicker for a user to find required information. We evaluated the effectiveness of query refinement terms recommended by the following three models.

1. Concept hierarchy model (SHReC) created using query terms from the search logs whose subsumptions are mined from the document collection as discussed in Section 3.

2. Query Flow Graph, QFG, that uses the occurrence frequency to rank all nodes connected to edges coming out of the node that contains a current query. This model was discussed in Section 4.

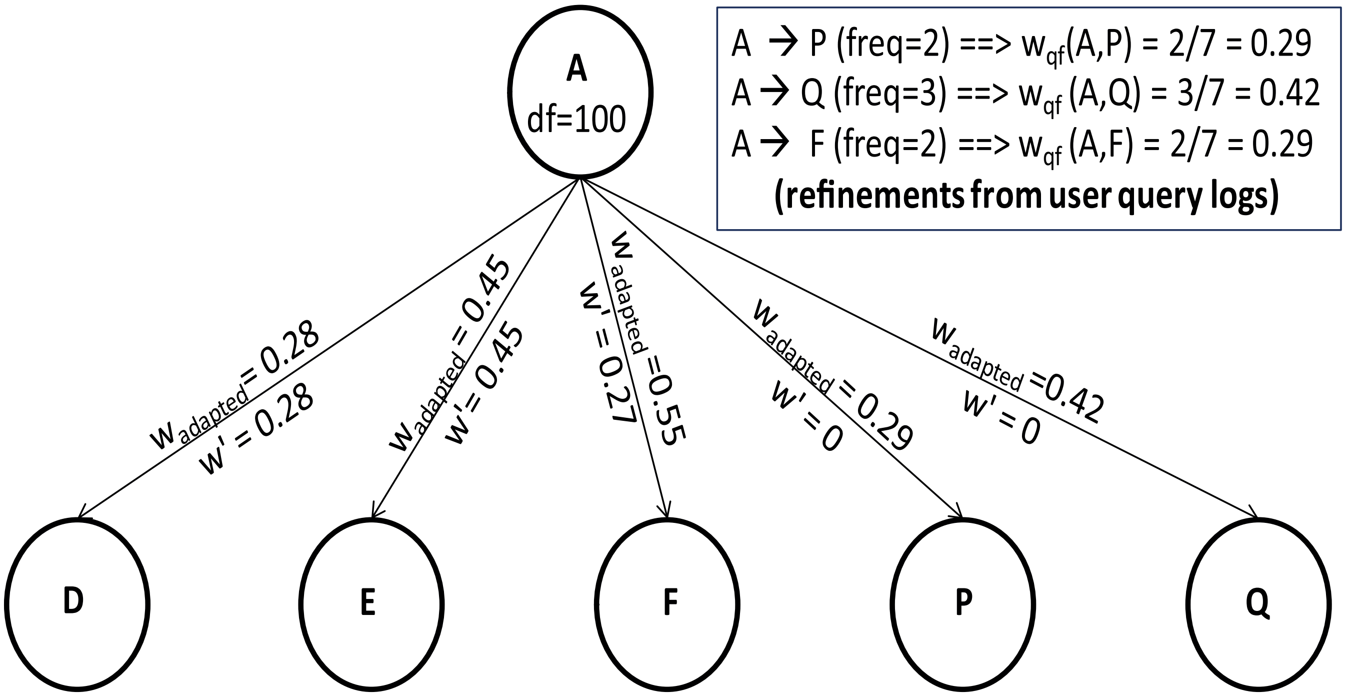
A
df=100
A → P (freq=2) ==> w_qf(A,P) = 2/7 = 0.29
A → Q (freq=3) ==> w_qf (A,Q) = 3/7 = 0.42
A → F (freq=2) ==> w_qf (A,F) = 2/7 = 0.29
(refinements from user query logs)
w_adapted = 0.28
w' = 0.28
w_adapted = 0.45
w' = 0.45
w_adapted = 0.55
w' = 0.27
w_adapted = 0.29
w' = 0
w_adapted = 0.42
w' = 0
D
E
F
P
Q

Figure 3: QF_SHReC: an amalgamation of the SHReC and QFG models

3. Query Flow SHReC denoted as QF_SHReC as discussed in Section 5.

We evaluated these models extensively with a method of automatic evaluation which utilises real user logs. Our evaluation methodology is described in Section 6.1 followed by specific details of how we built the SHReC model used in our experiments in Section 6.2.

6.1 Evaluation methodology

The search log data used in our experiments are obtained from the Intranet search engine of a higher education institution (University of Essex). Each search record contains the user query, a transaction time stamp and a session identifier. Query refinements are computed from queries in each session based on their time stamp and session identifier. We used two periods of logs for our evaluation. The first period consists of 13 weeks between October and December 2010 while the second period consists of 9 weeks between March and April 2011. The number of queries entered during the two periods were 42542 and 25259 respectively.

Our automatic evaluation framework measures the effectiveness of query recommendation over a period of time based on actual query logs. This is done by comparing the actual refinements observed in the search log files to those proposed by a query recommender model. In other words, the refinements in the query logs are viewed as ground truth. The framework has been validated with a user study [2]. We evaluate each model at periodic intervals; in our experiments, this is done on a weekly basis. The logs used as test data in a particular week are used for training in the next successive week. The training and test data for each week are therefore different. The evaluation method is also not circular because if there were n weeks of testing, the nth week of testing will be based on only the model adapted with log data from $n-1$ weeks. For all Q query refinements found in the query logs for the interval, we compute each model's Mean Reciprocal Rank (MRR) score as shown in Equation 9. We also calculate the mean Precision (Pr), Recall (Rc) and MRR for the top 10 recommended terms from a model. The overall Precision and Recall formulae for each query are given in Equations 10 and 11 respectively. We emphasize again that our evaluation only computed the precision and recall for the top 10 suggestions by each model not the overall precision or recall as shown in the equations.

$$MRR = (\sum_{i=1}^{Q} \frac{1}{r_i})/Q \tag{9}$$

where r_i is the rank of the actual query refinement in the list of refinement recommended by the model. Note that when the actual query refinement is not included in a model's list of recommended terms, then $1/r$ is set to zero.

$$\text{Pr} = \frac{|\text{Model's suggestions} \bigcap \text{Log Refinements}|}{|\text{Model's suggestions}|} \tag{10}$$

$$\text{Rc} = \frac{|\text{Model's suggestions} \bigcap \text{Log Refinements}|}{|\text{Log Refinements}|} \tag{11}$$

For each metric, the above evaluation process results in a score for each logged week. So overall, the process produces a series of scores for each query recommendation system being evaluated. These scores allow the comparison between different systems. One query recommender system can therefore be considered superior over another if a statistically significant improvement can be measured over a specified period of time.

6.2 SHReC creation with query log terms

Creation of a SHReC model requires two main resources: a collection of relevant documents and a list of terms. We created our Intranet collection by crawling the University of Essex's Intranet website. This institution is the same for which we have real user search logs. The crawl was done with Nutch (http://nutch.apache.org/) and converted into an index usable by Apache Solr (http://lucene.apache.org/solr/) search engine. This allows us to obtain the document frequency of any term and co-occurrence document frequency of any two terms for subsumption check. Our crawled collection contained a total of 77841 documents.

The main challenge in generating the list of important terms that should occur in the model is to ensure that the size of the list is manageable. We thought of extracting all keywords from the collection and up to two other words around each keyword to form tri-grams. However, this leads to a candidate list having over 100,000 terms which translates to querying the search engine over 10 billion times. This has adverse effect on the computational cost of building our SHReC model. In order to minimise the computational complexity associated with building the model, we opted to extract a smaller number of candidate terms from search logs of the same Intranet collection since they are readily available. We ensured that our training data (query terms from user logs) are extracted from period mutually exclusive of our two evaluation periods.

Using log terms as candidates implies that the concepts in the SHReC model would be of various length and are likely to be repeated by other users seeking similar information. We extracted 43,530 log terms from the period between mid-February and early May 2011 as training data for first evaluation period (i.e. 13 weeks). Also, 27,354 log terms between January and February 2011 were extracted and used as training data for the second evaluation period (i.e. 9 weeks). Our heuristic in selecting the training data was to use a period about the same size as the evaluation period so that the SHReC model is not at a disadvantage. The complete training (creation) of the model for the first period took about 2 weeks while the other took over 1 week.

6.3 QF_SHReC with click information

Based on the fact that less than 2% of all queries in our search logs resulted in more than 2 clicks, we simplified Equation 7 for QFG as follows:

$$w_{qf}(q,q') = \frac{C_0.\varphi_0(q,q') + C_1.\varphi_1(q,q') + C_k.\varphi_k(q,q')}{\Sigma_{r\in R_q}\Sigma_i C_i.\varphi_i(q,r)} \tag{12}$$

where C_k is the co-efficient factor of all click counts which are larger than 1. i.e. no matter whether a query has resulted in 2 or more clicks on resulting documents we treat all cases the same.

We then chose different combinations of C_0, C_1 and C_k for use in QFG and consequently QF_SHReC. Table 1 lists all the combinations we considered in running the automatic evaluation framework for our experiments.

The weighting scheme *standard* implies that our adaptation weights are from a standard query flow graph where no click information is incorporated. *no_zero* uses adaptation weights from an enriched query flow graph where reformulations which result with no clicks on the presented document list to the user are not considered. Both *boost_one* and *boost_one_more* use adaptation weights from enriched QFGs that boost queries with a single click on the presented list. Adaptation with the *penalise_many* weighting scheme means that the QFG weights used for adaptation penalises queries which attract two clicks or more.

	C_0	C_1	C_k
standard	1.0	1.0	1.0
no_zero	0.0	1.0	1.0
boost_one	1.0	2.0	1.0
boost_one_more	1.0	3.0	1.0
penalise_many	1.0	2.0	0.5

Table 1: Co-efficient factors of click counts.

7. RESULTS AND DISCUSSION

We discuss our evaluation results in this section. Firstly, we discuss results with respect to a comparative analysis of SHReC, QFG and our hybrid model (QF_SHReC) without differentiating query refinements with clicks on documents in Section 7.1. This is then followed by experiments with different co-efficient factors of click counts for computing the QFG weights before use in the hybrid model in Section 7.2.

7.1 Learning from all query log refinements

Experimental results of the query recommender models for the first test period between October and December 2010 are shown in Figure 4 and Table 2. Validation of this results on the second period (March to April 2011) are given in Figure 5 and Table 3. The values in bold font are significantly better than others across the same evaluation metric at 95% confidence while the underlined values are significantly worse. We employed a non-parametric measure (Kruskal-Wallis) since a plot of our results deviated from the normal distribution, that is p-value< 0.05. It should be noted that QFG evaluation results are always one week less than those from the SHReC models. This is because QFG is built periodically from the search logs and therefore has no evaluation in the first week for both periods.

SHReC was significantly worse than the hybrid model (QF_SHReC) and QFG across the four metrics as can be seen in Tables 2 and 3. Its performance in suggesting relevant and useful query refinement terms is about the same with little fluctuations every week throughout the evaluation for each period (Figures 4 and 5). This shows the importance of learning from other users' experiences. In SHReC, users do not benefit from previous users with similar queries; hence, the very poor performance. On the average, SHReC is only able to accurately suggest useful terms to a user about 1% of the time as shown by the mean MRR value of 0.0108. The MRR and precision at 10 are even much lower showing a mean value of about 0.2%. Also, only about 2% of the relevant suggestion terms according to our ground truth are in the top ten recommendations from S_SHReC.

Our hybrid model(QF_SHReC) is significantly better than QFG from the evaluation results across both evaluation periods. This is mainly due to the fact that it is able to mine suggestions from both the document collection and search logs. There is generally an increase in evaluation score every week for both models (QF_SHReC and QFG) with oc-

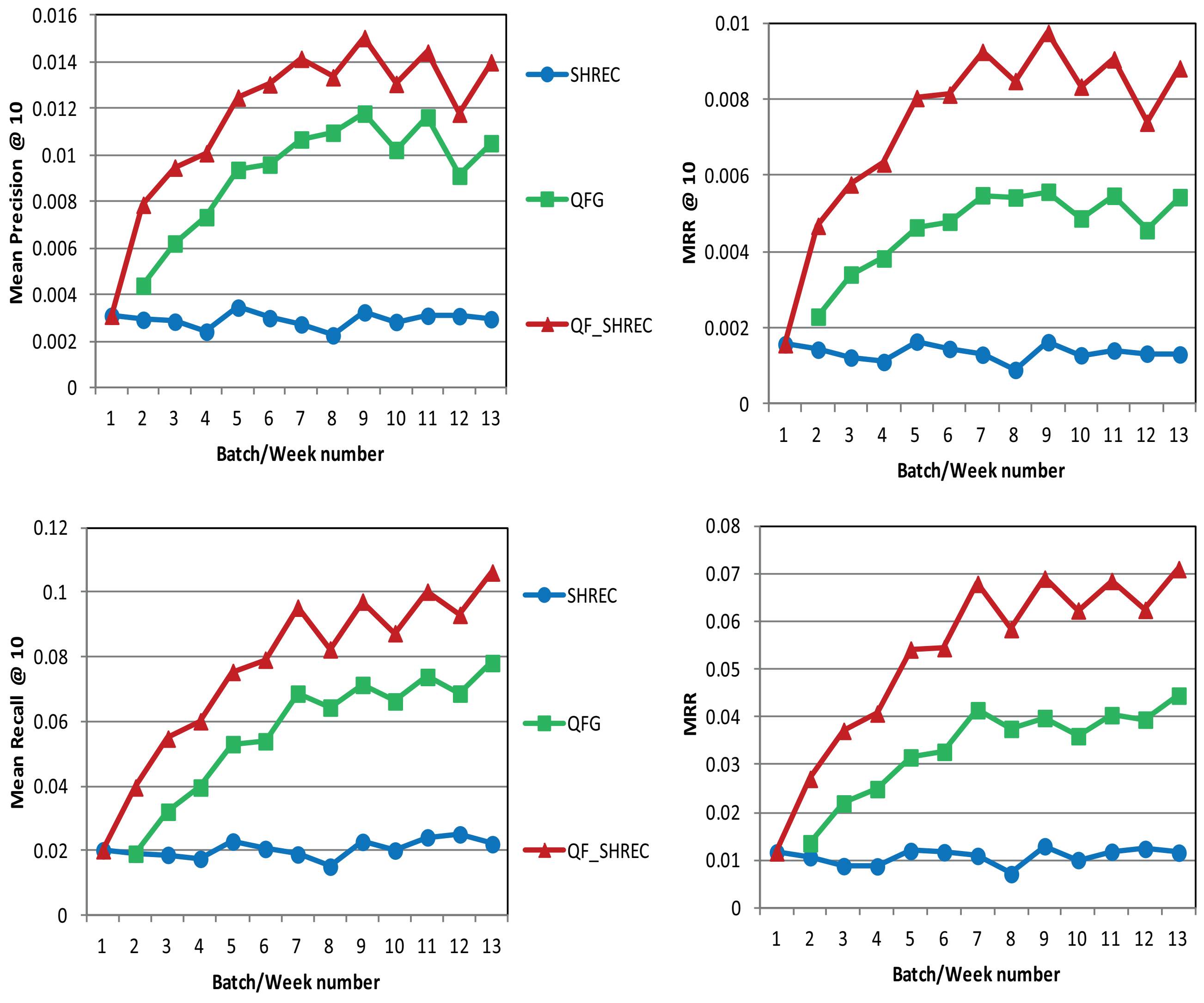

Figure 4: Evaluation results for Oct-Dec 2010

Table 2: Mean Weekly evaluation scores for period between Oct & Dec 2010

Model	MRR	MRR @ 10	Precision @ 10	Recall @ 10
SHReC	0.0108	0.0014	0.0029	0.0207
QF_SHReC	**0.0527**	**0.0074**	**0.0117**	**0.0763**
QFG	0.0337	0.0047	0.0093	0.0575

Table 3: Mean Weekly evaluation scores for period between Mar & Apr 2011

Model	MRR	MRR @ 10	Precision @ 10	Recall @ 10
SHReC	0.0097	0.0012	0.0027	0.0191
QF_SHReC	**0.0416**	**0.00605**	**0.0095**	**0.0597**
QFG	0.0269	0.0039	0.0074	0.0431

casional decrease. The graph of QF_SHReC is above QFG for the four evaluation metrics for both test periods.

The generally low evaluation scores across the different models might be due to the harsh gold standard (a user's immediate refinement) employed in our automated evaluation framework compared to what might be obtained in a

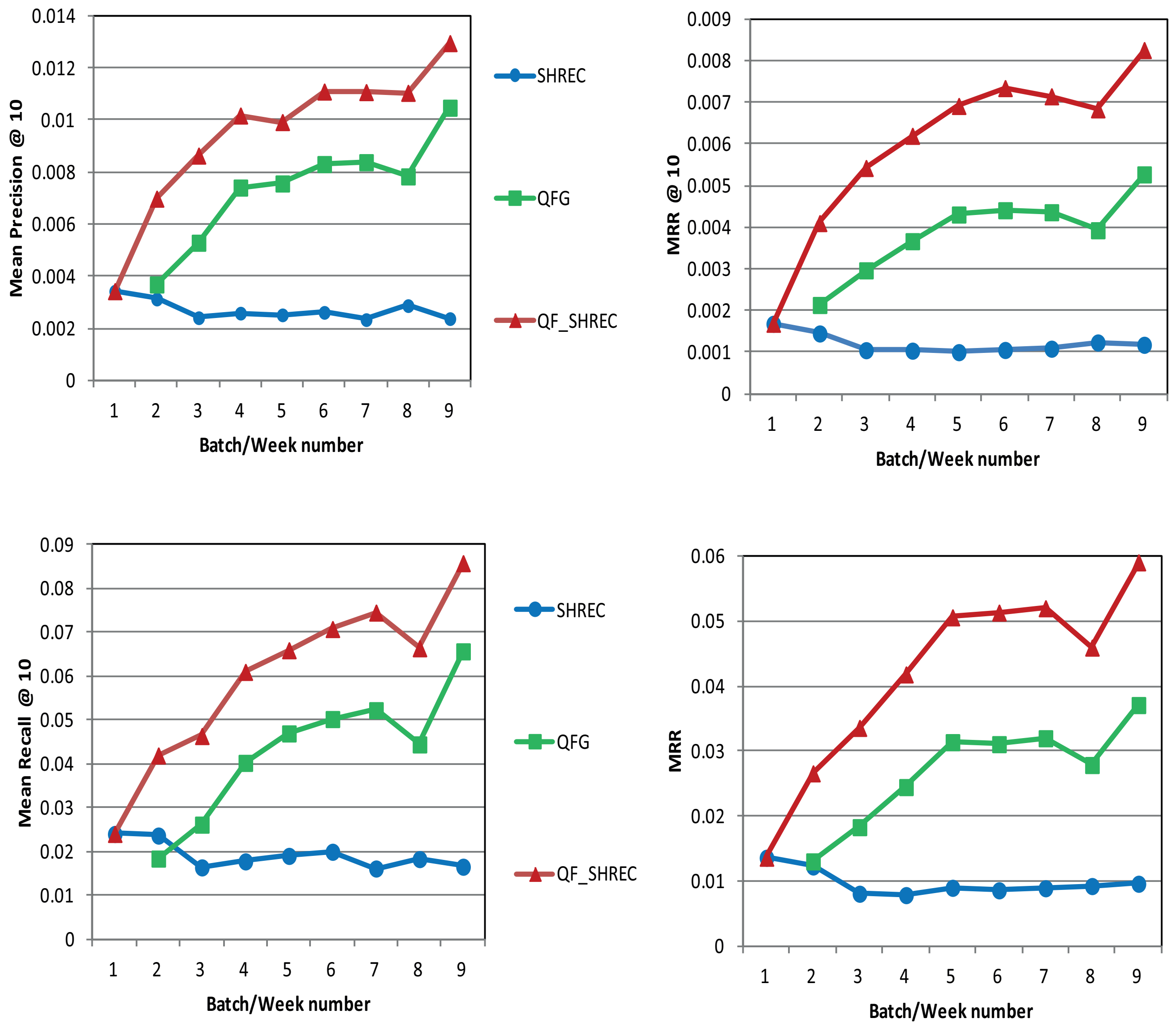

Figure 5: Evaluation results for Mar-Apr 2011

user studies where users are asked to rate the suggestions as relevant or not. The context of the ground truth refinements are also not fully captured by the evaluation framework and this might have changed based on the resulting documents shown to the user by the search engine.

Table 4: Mean Weekly query recommendations

Period	Test	SHReC	QF_SHReC	QFG
Oct-Dec 2010	3272	1381	1639	1426
Mar-Apr 2011	2806	1081	1280	1116

Table 4 shows the average number of weekly queries used in our evaluation and how many of these queries that the models were able to recommend at least one refinement term. We observe that all the three models recommended refinement terms for about half or less of the test queries terms. However, the difference in average queries for recommendations across all models is relatively small. Therefore, the significant difference in performance between SHReC and QF_SHReC is not solely due to the additional terms but also our novel method for re-ranking terms during the learning process.

7.2 Learning from refinements with click data

Table 5 shows the evaluation results when QFG weights used for adapting the original SHReC are based on different coefficients of click counts. Surprisingly, the hybrid that uses weights from the standard QFG which ignores the click data was generally better than other forms of weightings albeit not significantly. Also, boosting the terms with click gave significantly worse results across three of the four evalua-

Table 5: Mean Weekly evaluation with click information used [Mar - Apr 2011]

Model	MRR	MRR @ 10	Precision @ 10	Recall @ 10
QF_SHReC $_{standard}$	0.0416	0.0060	0.0095	0.0597
QF_SHReC $_{no\_zero}$	0.0347	0.0050	0.0079	0.0490
QF_SHReC $_{boost\_one}$	0.0297	0.0035	0.0068	0.0499
QF_SHReC $_{boost\_one\_more}$	0.0406	0.0058	0.0093	0.0593
QF_SHReC $_{penalise\_many}$	0.0407	0.0058	0.0093	0.0593

tion metrics. This contrasts with results from our previous work[1] on incorporating click information into QFG where boosting gave significantly better results. The unexpected results might be caused by terms already in the original SHReC mined from the documents collection as these are not dependent in anyway on click data.

8. CONCLUSIONS AND FUTURE WORK

The main contribution of this work is the amalgamation of two models for query recommendation. The first model (SHReC) is built from a document collection but using terms from search logs while the second (QFG) is built from search logs. We have shown that our hybrid model improves in its query recommendation performance over a period of time. Our experiments on an Intranet search engine over two different periods also showed that the performance of our hybrid model is significantly better than the individual models. We also discovered that click data did not improve the performance of our hybrid model when evaluated with our automatic evaluation framework.

We intend to repeat our experiments on an Intranet collection and search logs from a different academic institution. We are working to devise better strategies for selecting queries for recommendation from the hybrid model and also intend to incorporate our model in a live system and evaluate its effectiveness extensively with a user study.

Acknowledgements

This research is part of the AutoAdapt research project. AutoAdapt is funded by EPSRC grants EP/F035357/1 and EP/F035705/1.

9. REFERENCES

[1] M.-D. Albakour, U. Kruschwitz, I. Adeyanju, D. Song, M. Fasli, and A. De Roeck. Enriching query flow graphs with click information. In *AIRS'11 proceedings*, pages 193–204. Springer, 2011.

[2] M.-D. Albakour, N. Nanas, U. Kruschwitz, M. Fasli, Y. Kim, D. Song, and A. De Roeck. Autoeval: An evaluation methodology for evaluating query suggestions using query logs. In *ECIR'2011 proceedings*, pages 605–610, 2011.

[3] P. Anick. Using terminological feedback for web search refinement - a log-based study. In *SIGIR '03 proceedings*, pages 88–95. ACM, 2003.

[4] R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *SIGKDD'07 proceedings*, pages 76–85, 2007.

[5] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *CIKM '08*, pages 609–618, 2008.

[6] I. Bordino, C. Castillo, D. Donato, and A. Gionis. Query similarity by projecting the query-flow graph. In *SIGIR'10 proceedings*, pages 515–522. ACM, 2010.

[7] J. Callan, B. W. Croft, and S. M. Harding. The inquery retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83. Springer, 1992.

[8] D. Carmel, E. Farchi, Y. Petruschka, and A. Soffer. Automatic query refinement using lexical affinities with maximal information gain. In *SIGIR '02 proceedings*, pages 283–290. ACM, 2002.

[9] N. Craswell and M. Szummer. Random walks on the click graph. In *SIGIR '07*, pages 239–246, 2007.

[10] S. Cucerzan and R. W. White. Query suggestion based on user landing pages. In *SIGIR'07 proceedings*, pages 875–876. ACM, 2007.

[11] V. Dang and W. B. Croft. Query reformulation using anchor text. In *WSDM'10*, pages 41–50, 2010.

[12] T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. *IEEE Computer*, 40(8):34–40, 2007.

[13] H. Joho, M. Sanderson, and M. Beaulieu. Hierarchical approach to term suggestion device. In *SIGIR '02 proceedings*, page 454. ACM, 2002.

[14] R. Jones, B. Rey, and O. Madani. Generating query substitutions. In *WWW '06 proceedings*, pages 387–396, 2006.

[15] R. Kraft and J. Zien. Mining anchor text for query refinement. In *WWW'04*, pages 666–674, 2004.

[16] N. Nanas, V. Uren, A. De Roeck, and J. Domingue. Building and applying a concept hierarchy representation of a user profile. In *SIGIR'03 proceedings*, pages 198–204. ACM, 2003.

[17] F. Radlinski, M. Kurup, and T. Joachims. How does clickthrough data reflect retrieval quality? In *CIKM'08 proceedings*, pages 43–52. ACM, 2008.

[18] I. Ruthven. Re-examining the potential effectiveness of interactive query expansion. In *SIGIR '03 proceedings*, pages 213–220. ACM, 2003.

[19] M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *SIGIR'99 proceedings*, pages 206–213. ACM, 1999.

[20] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destination to enhance web search interaction. In *SIGIR'07*, pages 159–166, 2007.

[21] R. W. White and I. Ruthven. A study of interface support mechanisms for interactive information retrieval. *JASIST*, 57(7):933–948, 2006.

[22] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *SIGIR'96 proceedings*, pages 4–11. ACM, 1996.