Adaptation of the Concept Hierarchy Model with Search Logs for Query Recommendation 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
School of Computer Science
Tianjin University, China &
Department of Computing
Open University, UK
dawei.song@open.ac.uk

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

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

ABSTRACT

A concept hierarchy created from a document collection can be used for query recommendation on Intranets by ranking terms according to the strength of their links to the query within the hierarchy. A major limitation is that this model produces the same recommendations for identical queries and rebuilding it from scratch periodically can be extremely inefficient due to the high computational costs. We propose to adapt the model by incorporating query refinements from search logs. Our intuition is that the concept hierarchy built from the collection and the search logs provide complementary conceptual views on the same search domain, and their integration should continually improve the effectiveness of recommended terms. Two adaptation approaches using query logs with and without click information are compared. We evaluate the concept hierarchy models (static and adapted versions) built from the Intranet collections of two academic institutions and compare them with a state-of-the-art log-based query recommender, the Query Flow Graph, built from the same logs. Our adaptive model significantly outperforms its static version and the query flow graph when tested over a period of time on data (documents and search logs) from two institutions' Intranets.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

Keywords

Log Analysis, Query Suggestions, Concept Hierarchy

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

1. INTRODUCTION

The need to assist a user in finding relevant information has led to continuous changes in the functionalities of search engines. Prominent internet search engines now provide user-friendly features such as query auto-completion, query refinement lists and visual representation of related concepts to the query in addition to the typical ranked list of retrieved documents (e.g. Google, Bing and Yahoo! Web search engines). These extra features are expected to make it easier and quicker for a user's information need to be satisfied. We focus on the techniques used for generating suitable refinements to a current (or an initial) query, known as *query recommendation*, which is one of the interactive features in modern search engines.

Although lots of models have been proposed for recommending query refinement terms, most have exploited the document collection or previous search logs but not both. Our intuition is that these two approaches to query recommendation provide complementary conceptual views on the same search domain. Recommendations from the collection typically discover relationships between terms based on statistical analysis giving a conceptual view of the domain. On the other hand, the search logs provide a conceptual view of the domain based on collective user intelligence. Each approach also has its inherent weakness. For instance, the search logs approach might not be able to give any suggestions for new queries that were never used in the past. On the other hand, the document collection approach might give too many unsuitable suggestions for current queries that have a very high document frequency leading to a statistical relationship with many of the terms in the collection. Therefore, our hypothesis is that the integration of the two approaches will improve the query recommendation quality and in turn the retrieval performance. A conceptual view of the search domain would be a natural bridge and common foundation for the two approaches.

Specifically, we propose the integration of these two approaches by continuously adapting the concept hierarchy model built from a document collection with user interactions derived from query logs. The concept subsumption

hierarchy [24] is a good choice as a simple but effective approach to query recommendation. Our research focuses on Intranet search which (similar to Enterprise search) is different from Web search [15]. The document collection on an Intranet is relatively small and changes less frequently. It is thus feasible to build a domain model to reflect the conceptual structure within the collection. In this scenario, the topic structure generated by the concept hierarchy might map onto a pre-defined structure in the organisation. For example, a concept hierarchy model generated from documents in an academic 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. On an Intranet, it can also be expected that identical queries are more likely to be related to the same user information need since the context of search is more limited than in Internet search. Therefore, previous user interactions can be used to improve a new user search by suggesting query refinement terms from a previous session with a similar query. Query terms used in the same session also implies that they are related and can be used to improve query recommendation models.

In this paper, we propose a novel method to adapt the model with search logs and compare with its static equivalent as a baseline which does not incorporate any previous user interactions. We also compare these hierarchy models (static and adaptive) with a state-of-the-art adaptive query recommendation approach called the query flow graph model [6] which is built solely from search logs. We test the effectiveness of these models using an automated evaluation based on query log data by experimenting with search logs obtained from two academic institutions' search engines.

Section 2 discusses related work on query recommendation and utilization of search logs to improve information seeking. We then provide in-depth details of the concept hierarchy model, how we utilise it for generating query refinement terms and our proposed method for adaptation with search logs in Sections 3, 4 and 5. Section 6 explains our experimental set-up while evaluation results are discussed in Section 7. We conclude the paper in Section 8 by outlining our main contributions and plans for future work.

2. RELATED WORK

Recommendation of refinements to queries is a common feature of modern search engines, both general and specialised ones. 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 their usefulness [23, 26, 27]. Knowledge of the query's context, which can be a function of the user's knowledge about the domain, previous search queries, etc., is a crucial factor when suggesting terms that help a user find specific information faster. The two main resources available for query recommendation are the document collection (including anchor logs) [28, 24, 10, 18, 20, 13] and search logs [3, 19, 12, 4, 17, 6, 5, 7, 8], which can also be used as forms of implicit or explicit feedback to re-rank retrieved documents.

The approaches that utilise the collection discover relationships between terms using all documents (global) [9, 13] or those retrieved as relevant to a query (local) [24, 10]. Terms considered 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 recommendation models when there is a significant change in the underlying collection if their suggestions are mined from all documents. This method is perhaps most useful for new search engines with little or no query logs.

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 [19], while global recommendations are derived from a group of users or all users [6, 8]. The main drawback is that logs have to be collected over a period of time before such techniques can be applied since data from one search engine might not be easily applicable to others, especially for Intranets.

Mining post-query click behaviour has been studied and applied in information retrieval tasks. For example, using landing page information to derive query suggestions [12] and mining user search trails for search result ranking [26], where the presence of a page on a trail increases its query relevance. Click graphs were also used to derive labels to shortcut search trails to help users reach target pages efficiently [25]. Since users are typically reluctant to provide explicit feedback on the usefulness of ranked results returned for a search query, automatic extraction of implicit feedback has continued to be exploited by researchers. Click-through data is one form of implicit feedback left by users, which can be used to learn the retrieval ranking function [17]. Queries and clicks can be interpreted as "soft relevance judgements" [11] to find out the user's actual intention and interests. Query recommendations can then be derived, for example, by looking at submitted queries and building query flow graphs [6, 7], query-click graphs [11], cover graphs [4] or association rules [14]. Mining query logs has also been combined with query similarity measures to derive query modifications [19].

The Query flow graph model (QFG) [6] has been successfully applied to mine query suggestions from query logs. It is a directed graph, which contains a set of nodes consisting of all distinct queries submitted to a search engine, while the edges between two nodes represent a user query refinement. Edges are weighted primarily based on the frequency of such refinements, though the actual weighting function depends on the application. Recommendations can be made for a query if it is found in any of the QFG nodes by obtaining terms at the end of all edges that emerge from the query node. These terms can then be ranked based on the weights of the edges leading to them from the query node. In this paper, we compare the concept hierarchy used in our work to QFG since they are both directed graph structures but one is typically built from a document collection while the other is built from query logs.

3. CONCEPT HIERARCHY MODEL

The concept hierarchy model [24] was initially introduced 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, we adopt the acronym SHReC (**S**ubsumption **H**ierarchy for **Re**sult

Clustering) of an existing implementation[1] and use this to refer to the concept hierarchy model in this paper. Although SHReC was introduced for clustering retrieval results, it can easily be applied to an Intranet collection. This model can be utilised for suggesting terms for query refinements (addition or replacement) by showing a sub-tree from the initial hierarchy encapsulating the given query as used in [18]. A more intuitive way to utilise the model for query refinement is to transform terms in the sub-tree encapsulating the query into a ranked list based on the strength/closeness of their links to the query in the SHReC graph.

3.1 Building the SHReC Model

The SHReC model is automatically derived as a hierarchical organization of concepts from a set of documents without using training data or standard clustering techniques [24]. The basic idea 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 are therefore dependent on document frequencies. It should be noted that a term as used in this paper for the SHReC model is the equivalent of a concept and does not necessarily mean a single keyword. A term could be an entire query which can be formulated as a word, keyword, phrase or n-grams. A term 'x' is said to subsume 'y' when Equation 1 is met; where α is the strength of subsumption.

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

The relationship between 'x' and 'y' are derived from statistical probabilities of their co-occurrence in the collection. Operationally, if *df* is a function that returns the document frequency of a given term, $df(x) > df(y)$ and $df(x \wedge y)/df(y) \geq \alpha$. Although $\alpha = 1$ is ideal for a complete subsumption, a slightly smaller value can be chosen in order to allow few occurrences of term 'y' with other terms other than 'x'. $\alpha = 0.8$ was chosen in [24] based on empirical studies and we used the same α value in our work.

3.2 SHReC from an Intranet collection

Although the original SHReC model was built from top documents returned by a search engine given a query, we extended this to build the model from a substantial crawl of an Intranet collection. SHReC can be viewed as a directed graph $G_{SHReC} = (V, E, w)$ where:

- V is a set of nodes containing all the unique concepts or terms derived from documents in the Intranet collection with a special node 'SHReC root' representing the root of the graph tree;
- $E \subset V \times V$ is the set of directed edges;
- $w : E \rightarrow (0..1]$ is a weighting function that assigns to every pair of terms $(x, y) \in E$ a weight $w(x, y)$.

Generating a list of suitable candidate terms or concepts (V) to occur in the model is not trivial in our case since unlike in the original subsumption hierarchy approach, we do not have a current query to guide the selection of related terms. We must therefore ensure that our candidate terms cover a reasonable proportion of the vocabulary that users might employ to formulate their queries. Meanwhile, the computational complexity associated with building a SHReC model must also be considered. This is because each candidate term should be compared with all other terms to obtain their co-occurrence document frequency. For example, a list of 50,000 candidate terms will require about 2.5 billion (50000^2) co-occurrence checks between terms while building the model.

A generic approach will be to extract all unique keywords found in the collection as candidate terms. Additional candidates can then be generated by extracting n-grams (phrases) from documents in the collection that contain these unique keywords to cater for queries that contain more than one keyword. The maximum size of n in each n-gram should be limited to a small number (≤ 5) as users are less likely to use longer phrases from an analysis of the query logs available to us. The extracted n-grams can also be restricted to those that match a particular syntactic pattern (e.g. Noun Preposition Noun for a 3-gram) [21].

4. SHReC AS A QUERY RECOMMENDER

We employ the SHReC model for suggesting useful terms for query refinement in order to satisfy a user's information need. We propose to do so 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 weighting function shown in Equation 2. The function implicitly combines the subsumption ratio ($df(x \wedge y)/df(y)$) and document frequency ratio ($df(y)/df(x)$) between any two terms ('x' subsumes 'y') in the model. The subsumption ratio is the quotient of the co-occurrence document frequency ($df(x \wedge y)$) and the child's document frequency, and is used to determine subsumption between terms in the hierarchy (see Equation 1). The document frequency ratio is the ratio of the document frequencies of two connected terms in SHReC. It ensures that the closer the document frequencies of two directly linked terms, the more useful they are as a refinement for one another.

$$\text{If 'x' subsumes 'y', then } w(x, y) = df(x \wedge y)/df(x) \quad (2)$$

Figure 1 shows a sample SHReC model to illustrate its use for query recommendation. We only show the computation of weights for the link between terms 'A' and others to avoid cluttering the diagram. Given a query such as term 'F' in the figure, our technique will suggest terms 'H', 'C', 'G', 'B' and 'A', ranked in order of their computed weights (w).

The example above assumes that term 'F' would be found in the model irrespective of whether it is a keyword or phrase. This is not intuitive as it means that a lot of queries might not be matched in the model. Since many queries are likely to be formulated as phrases and might contain more than one word, one or more of the query words might occur separately in the model. Therefore when a query is not found in the model, its keywords are extracted and suggestions are provided by aggregating the recommendations from each of this keyword. The combined recommendations are re-ranked based on magnitude of their weights with respect to the keyword that recommend them. This should reduce the number of times a suggestion is not made, since previous works [27, 26] have found that users prefer to be provided suggestions regardless of their usefulness.

[1] http://shrec.sourceforge.net/

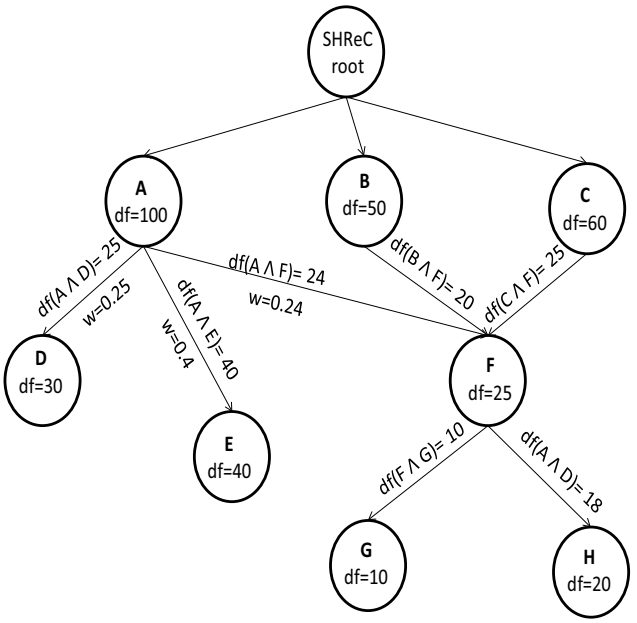

Figure 1: An example SHReC model

5. ADAPTIVE SHReC MODEL

In order to improve the effectiveness of query refinement recommendations from the SHReC model, we propose its adaptation with users' search logs. This ensures that query terms not captured during the creation of the SHReC model are incorporated into the model. Adaptation with query logs can be seen as an amalgamation of the query flow graph model [6] (see Section 2) and SHReC. The adaptation process is carried out for pairs of query refinements found at periodic intervals (e.g. every week) to minimise its computational cost. In a live system, such adaptation will be programmed to take place at a time when the search engine is least busy such as past midnight on Sunday morning in preparation for the new week. We start with an initial SHReC model built as described in Section 4 and then continuously update the model with the refinement terms found in the search logs. The weight of each refinement in the log is computed as shown in Equation 3. Such refinements might be restricted to only those in which there was at least a click after refining the current query or all refinements found in the logs for the adaptation period. The total frequency is cumulative so that collective user intelligence is taken into account rather than those for just that periodic interval.

$$\text{for x} \rightarrow \text{y}, \; logWeight(x, y) = \frac{totalFrequency(x \rightarrow y)}{totalFrequency(x \rightarrow *)} \quad (3)$$

where * is any term apart from 'x' found as a refinement of 'x' in the logs from inception of SHReC adaptation. Function $totalFrequency$ is also computed cumulatively from the beginning of the adaptation process.

The adaptation process is formalised with the pseudo-code listed in Algorithm 1. It starts with the normalisation (as listed in Algorithm 2) of weights w for each edge in the initial SHReC built from the document collection only (Lines 1-3 of Algorithm 1), so that the total weight of all edges originating from a term node sums up to 1. The weights of new refinement terms added to the model from the logs will also have values between 0 and 1. The function $norm(G_{SHReC})$ therefore ensures that the weights of newly added terms are comparable to those already in the SHReC graph. The weights of edges from the initial SHReC are kept in w' for updates with the log weights. This ensures the original SHReC weights remain influential and comparable to the log weights.

Each unique query refinement (reformulation) pair found in the query logs between the periodic interval used for adaptation is then incorporated into the model. We cater for all possible scenarios as shown by the conditional statements listed on Lines 5, 7, 9 and 11 of the main adaptation algorithm. We update the weights between refinement terms in the model if both already exist by adding the weight calculated from the query logs to the current normalised SHReC weights (Line 6). When only one of the terms exist in the current SHReC, the other term is added (as parent or child) with the weights of the edge between them initialised to the computed weights from the logs (Line 8 & 10). Both

Algorithm 1 Adapting SHReC with query logs

Require: $G_{SHReC} = (V, E, w)$, a concept hierarchy graph
Require: $G_{normSHReC}$, normalised SHReC
Require: $R = \{r_1, \ldots, r_n\}$, reformulations in test period
Require: $r_i = x \rightarrow y$, query 'x' reformulated as query 'y'
Require: $lw(r_i) = logWeight(x, y)$, log weights (Eqn 3)
1: **if** (Inception of original SHReC's adaptation) **then**
2: $G_{normSHReC}$=$norm(G_{SHReC})$= $(V, E, w_{adapted}, w')$ (from Algorithm 2)
3: **end if**
4: **for each** $r_i \in R$ **do**
5: **if** $x \in V$ AND $y \in V$ **then**
6: $G_{normSHReC}$.updateweights$(x, y, lw(r_i))$ i.e. $w_{adapted}(x, y) \leftarrow w'(r_i) + lw(r_i)$
7: **else if** $x \in V$ AND $y \notin V$ **then**
8: $G_{normSHReC}$.addDescendant$(x, y, lw(r_i))$ i.e. $w_{adapted}(x, y) \leftarrow lw(r_i)$
9: **else if** $x \notin$ SHReC AND $y \in$ SHReC **then**
10: $G_{normSHReC}$.addAscendant$(y, x, lw(r_i))$ i.e. $w_{adapted}(x, y) \leftarrow lw(r_i)$
11: **else**
12: $G_{normSHReC}$.add$(x, y, lw(r_i))$ i.e. $w_{adapted}(x, y) \leftarrow lw(r_i)$
13: **end if**
14: **end for**

Algorithm 2 Function $norm(G_{SHReC})$

Input: $G_{SHReC} = (V, E, w)$, a concept hierarchy graph
Output: $G_{normSHReC} = (V, E, w_{adapted}, w')$
w' stores normalised weights for terms in original SHReC
1: **for each** $v \in V$ **do**
2: $sum \leftarrow 0$
3: OUT $\leftarrow$ getAllOutNodes(v), where $OUT \subset V$
4: **for each** $o \in OUT$ **do**
5: $sum+ = w(v, o)$
6: **end for**
7: **for each** $o \in OUT$ **do**
8: $w'(v, o) \leftarrow w(v, o)/sum$
9: **end for**
10: **end for**
11: **return** $G_{normSHReC}$

terms are added into the graph if they do not currently exist using weights computed from logs (Line 12). We do not re-normalise after updates because w' is always from the original SHReC and the weights from the logs are computed from cumulative frequency. Normalising and adding cumulative log weights will mean that refinements in previous adaptation periods contribute more than to the weight updates thereby giving them undue advantage.

Figure 2(i) illustrates the normalisation step of the adaptation process using a sub-tree from Figure 1. The sum of w' from node 'A' equals to 1. We then exemplify other steps of the adaptation process in Figure 2(ii). Here, the refinements from the query logs during the adaptation period from inception are assumed to be the three shown in the rectangle at the top-right corner of the figure with the log weights calculated using Equation 3. 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: $w_{adapted}(A, F) \leftarrow w'(A, F) + lw(A, F)$ from Line 6 of Algorithm 1. The ranking of the recommendation also changes since some of the log weights are higher than those from the original SHReC. For instance, term 'Q' will be ranked higher than 'D' during query recommendation.

6. EXPERIMENTAL SETUP

Our aim is to improve a user's search experience by suggesting refinement terms related to a current query. Recommended terms can be useful as they can assist users in choosing the right query terminology that leads them to most relevant documents. 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 evaluate the effectiveness of query refinement terms recommended by the following three models.

1. S_SHReC, static SHReC built from only the Intranet document collection (see Section 4) and is not updated in any form throughout the test period.

2. A_SHReC, our adaptive SHReC discussed in Section 5 in which refinements extracted from search logs are continually incorporated at periodic intervals. Two forms of adaptation are carried out.

   (a) Using query logs without click information. In this case, we use all refinements from the query irrespective of whether or not there is a click on resulting documents after refinement.

   (b) Using query logs with click information. In this case, we consider the click information as another signal for implicit feedback where we use only refinements in which there is at least one click on results immediately after refinement.

3. QFG, a Query Flow Graph[6] discussed in Section 2 built continually at periodic intervals from the same search logs. Our version of QFG ranks only the immediate nodes coming out of a query node rather than all nodes in the graph. This is because the top ranked recommendations are always those nearest to the query node and we are able to reduce the expensive computational costs associated with ranking all nodes in the graph for the query. It also ensures that QFG's recommendation method is comparable to SHReC's, extracted only from immediate nodes related to a query node in SHReC. QFG was built using all refinements in the logs as well as only those in which there was at least one click immediately after reformulation, for comparison with the adaptive SHReC versions. Lastly, when a query is not found in the model, it is tokenised into keywords (just like SHReC) and suggestions aggregated from the recommendations that resulted from each of its token. The combined recommendations are then re-ranked based on magnitude of their weights with respect to the tokens that recommend them.

These models are evaluated extensively with an automated evaluation methodology which utilises real user logs. The datasets (web crawl and search logs for two academic institutions' Intranets) used in our experiments are described in Section 6.1. Our evaluation methodology is described in Section 6.2, followed by specific details of how we built the SHReC model used in our experiments in Section 6.3.

6.1 Datasets

The data (documents collection and search logs) used in our experiments were obtained from the Intranets of two UK academic institutions namely University of Essex and The Open University, referred to as *Essex* and *OU* respectively. The document collections were created by crawling the main websites of the institutions with Nutch[2]. We limited each crawl to a maximum depth of fifteen and simulated the Intranet search engines of the institutions on our machines using Apache Solr[3]. This allows us to obtain the document frequency of any term and co-occurrence document frequency of any two terms required for SHReC creation. The crawls for *Essex* and *OU* were done in July 2011 and October 2011 containing 77,841 and 220,059 documents respectively.

The search log data used in our experiments are obtained from the Intranet search engines of *Essex* and *OU*. Each search record contains the user query, a transaction time stamp, a session identifier and URLs visited by the user. Query refinements are extracted from queries in each session based on their time stamp. We used logs of 12 weeks between February and May 2011 for our evaluation. The total number of queries entered during this period were 32,212 and 117,358 for *Essex* and *OU* respectively.

6.2 Evaluation methodology

We use an automated evaluation framework, *AutoEval*, which measures the effectiveness of query recommendations over a period of time based on actual query logs [2]. The evaluation framework was validated with a user study which showed that scores from the automated evaluation correlated highly with user evaluation of the recommendations. The query recommender models are evaluated automatically by comparing the actual refinements observed in the log files, for which there is at least one user click on results after refinement, to those proposed by a model. In other words, we interpret a user click after a reformulation as presence of a relevant suggestion.

Each model is evaluated continuously at periodic intervals. In our experiments, this is done on a weekly basis. The logs

[2]http://nutch.apache.org/
[3]http://lucene.apache.org/solr/

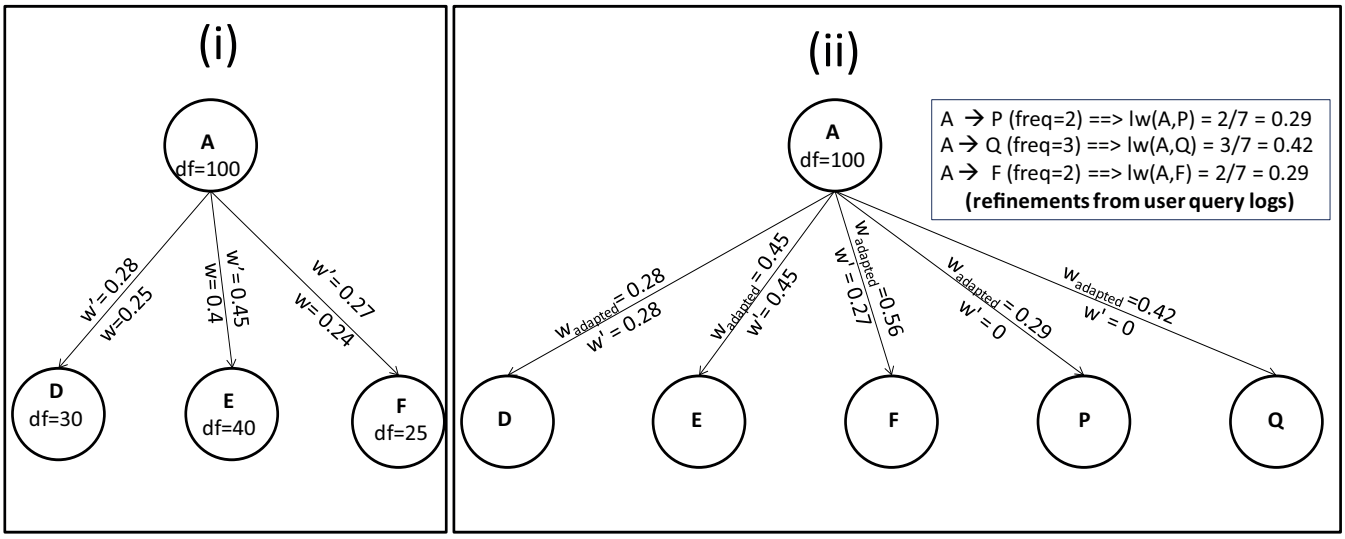

Figure 2: Adaptation of SHReC with search logs

for each successive week are used for training and testing respectively. In other words, we adapt the models with week i's log refinements (training) and use the logs from week $i+1$ as test data; where $1 \leq i \leq n$, the number of weeks in the test period. The training and test data are therefore different for each week. The model is first evaluated on a particular weekly batch and then updated from the logs of the same batch. The evaluation methodology is not circular as the data is first used for evaluation then for learning.

For all Q, i.e. query refinements with user clicks found in the query logs for a particular week, we compute each model's Mean Reciprocal Rank (MRR) score as shown in Equation 4. The evaluation process results in a score for each logged week. Overall, the process produces a series of scores for each query recommendation model being evaluated. These scores allow the comparison between different models. One query recommender model can therefore be considered superior over another if a statistically significant improvement can be achieved over the specific period.

$$MRR = \frac{1}{Q}(\sum_{i=1}^{Q} \frac{1}{r_i}) \tag{4}$$

where r_i is the rank of an actual query refinement in the list of refinements 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.

6.3 SHReC creation

Creation of a SHReC model for an Intranet requires two main resources: a collection of documents and a list of candidate terms. The main challenge is in generating the list of suitable candidates that should occur in the model. As discussed earlier in Section 3.1, the computational complexity of the creation process necessitates that the size of the list is manageable. In order to determine the reasonable size of candidate terms based on the trade-off between computational cost and effectiveness of recommendations, we tested different sets of candidates extracted from *Essex*'s collection. Firstly, we extracted all keywords from the collection and up to two other words around each keyword to form bi-grams and tri-grams leading to 54,830 unique candidates. This set is labelled as CS_{all} (candidate set for all terms). CS_{15k} consists of 15,086 terms from the list of keywords with document frequency between 20 and 13,226 (the maximum document frequency), and terms not in this set occurred in less than 0.026% (20/77,841) of the crawled Intranet collection. We also created CS_{10k}, CS_{5k} and CS_{1k} consisting of 10,148, 5,005 and 1,002 terms respectively. The terms with the least document frequency were 35, 89 and 631 for CS_{10k}, CS_{5k} and CS_{1k}.

Table 1 shows a comparison of the candidate sets across some important features. The number of subsumptions increases with the size of candidate set although there is no direct proportionality. For example, whilst the number of candidates in CS_{all} is more than triple of those in CS_{15k}, the proportion of subsumptions is just about twice. This might mean that the most important terms for subsumption are those with medium document frequencies. The computational cost on the other hand is like a geometric progression of the number of terms. The cost here refers to the time it takes to completely build a SHReC model on an Intel Duo core CPU (3GHz each), 4GB RAM with Windows 7 enterprise edition OS running 64-bit Java codes and using 4GB virtual memory. A candidate set of a thousand terms (CS_{1k}) takes less than 30mins to complete, while a candidate set ten times bigger (CS_{10k}) completes its SHReC creation in over 36hrs, which is about seventy-two times the amount of time. This is due to the amount of co-occurrence document frequency checks whose time complexity is $O(n^2 - n)$.

The average MRR results from the automated evaluation of the SHReC built from these candidate sets over a 12 weeks test period are also shown in Table 1. The query recommendations from the static SHReC built from CS_{all} are significantly better than those from the other candidate sets (paired t-test at 95% confidence). However, there is only a marginal difference among the adaptive versions of the models after 12 weeks of adaptation with all observed refinements from search logs. This implies that though the size of candidate set is important for the original static SHReC, its

Table 1: Comparison of different *Essex* candidate sets for SHReC creation

	CS_{all}	CS_{15k}	CS_{10k}	CS_{5k}	CS_{1k}
Minimum document frequency	2	20	35	89	631
Maximum document frequency	13,226	13,226	13,226	13,226	13,226
Size of Candidate set	54,830	15,086	10,148	5,005	1,002
Number of subsumption in SHReC	4,275,737	2,281,606	1,284,458	444,129	25,084
Computational Cost (approx.)	8weeks	4days	36.5hrs	10hrs	25mins
Average MRR of Static SHReC	**0.0109**	0.0006	0.0004	0.0005	0.0002
MRR in 12th week of adaptation	0.1298	0.1292	0.1272	0.1276	0.1274

Table 2: Comparison of different *OU* candidate sets for SHReC creation

	CS_{15k}	CS_{10k}	CS_{5k}	CS_{1k}
Minimum document frequency	85	157	451	4,487
Maximum document frequency	134,505	134,505	134,505	134,505
Size of Candidate set	15,018	10,000	5,001	1,000
Number of subsumption in SHReC	196,406	171,711	57,039	13,959
Computational Cost (approx.)	3days 6hrs	38.5hrs	7.5hrs	29mins
Average MRR of Static SHReC	0.0003	0.0005	0.0007	0.0007
MRR in 12th week of adaptation	0.1618	0.1619	0.1620	0.1619

adaptation with search logs minimises the effect of the size of candidates when utilised for query recommendation.

We obtained identical patterns for similar candidate sets extracted from *OU* as shown in Table 2. The candidate set for all terms which had over 11,8820 terms was not tested as this would not be practical. Based on the above analysis of different candidate sets across the two datasets, especially with respect to the trade-off between time complexity and effectiveness, we selected CS_{10k} whose SHReC is used for further analysis and comparison to the QFG in the next section.

7. RESULTS AND DISCUSSION

Evaluation results from our experiments are analysed in this section. Firstly, we discuss results related to a comparative analysis of Static SHReC (S_SHReC), QFG and Adaptive SHReC (A_SHReC) which uses all query refinements found in the search logs for the test in Section 7.1. We then analyse results from experiments when SHReC is adapted with only refinements where there is a user click immediately after the refinement in Section 7.2. We also compare this adaptive SHReC to a query flow graph (equivalent to the query click graph [11]) which also uses only refinements with user clicks.

7.1 Adaptation with all query log refinements

Experimental results of the query recommendation models for the two Intranets are plotted on graphs shown in Figures 3 and 4. The average weekly MRR evaluation across the 12 weeks test period are given in 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. The z-values are also shown in the Tables to indicate other levels of significance. For example, a z-value above 2.58 indicates a significantly better or worse result at 99% confidence depending on polarity (positive or negative). 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 cumulatively on a weekly basis from the search logs and therefore has no evaluation in the first week. SHReC is first built from a collection and can be evaluated with the query logs from the first week as test data before its subsequent adaptation for other weeks.

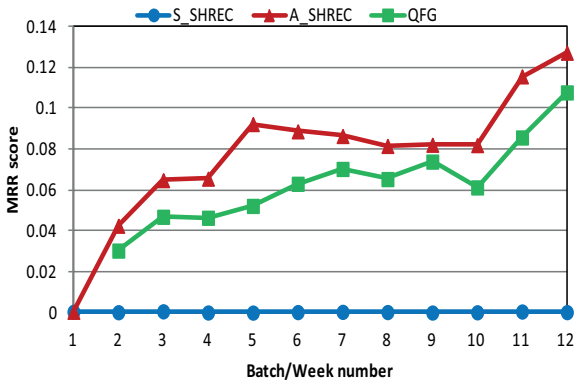

Figure 3: *Essex* evaluation results for adaptation with all query log refinements

Static SHReC (S_SHReC) was significantly worse than adaptive SHReC and QFG at 99% confidence ($z < -2.58$) as shown in Table 3. Its performance in proposing suitable query refinement terms is about the same with very little fluctuations every week throughout the evaluation for both *Essex* and *OU* (Figures 3 and 4). A possible reason for the low performance of S_SHReC might be that the statistical co-occurrence led to lots of noisy subsumptions between terms. The evaluation method might also have adversely affected S_SHReC since queries from search logs are used for the automated testing. Popular queries are likely to be

Table 3: Evaluation scores for adaptation with all query log refinements

Model	Average MRR (*Essex*)	MRR in 12th week (*Essex*)	Average MRR (*OU*)	MRR in 12th week (*OU*)
S_SHReC	0.00044 (z=-4.45)	0.00042	0.00078 (z=-4.45)	0.00066
A_SHReC	**0.07750** (z=3.23)	0.12717	**0.11038** (z=2.99)	0.16196
QFG	0.06400 (z=1.24)	0.10776	0.09764 (z=1.49)	0.14694

repeated often by different users thereby giving advantage to models that utilise the logs for their recommendation.

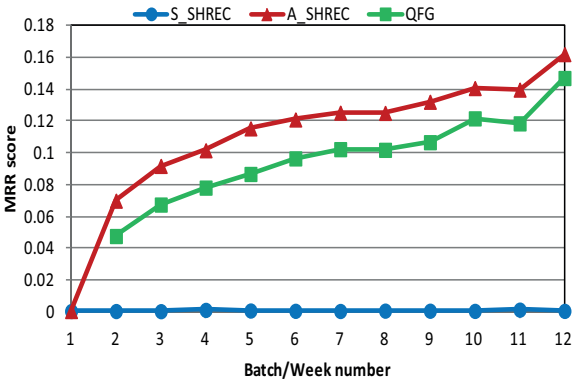

Figure 4: Evaluation results for *OU* for adaptation with all query log refinements

Our adaptive SHReC (A_SHReC) is significantly better than QFG from the evaluation results across both datasets as can also be seen in Table 3. This is mainly due to the fact that it is able to effectively integrate suggestions mined from both document collection and search logs. Users are therefore able to benefit from other previous users with similar information needs as reflected by their queries. The superior performance of A_SHReC over the other two models verifies our hypothesis that the collection and search logs are complementary sources for query recommendation. The continuous increase in MRR on every week is an indication that our adaptation method supports gradual learning and the increase is likely to continue for a long period afterwards. Although, the average MRR score was about 8% for *Essex*, a MRR score of about 13% in the 12th week of adaptation (Table 3) means that the average value might not give a true picture of the model's effectiveness. This is because the MRR score in the first week was very low, same as S_SHReC's. The same analysis applies to *OU* where the difference between the average score and score in the 12th week is over 5%. The MRR score from the second week onwards varied between 7-16%. The slightly higher scores obtained from *OU* compared to *Essex* is likely to be due to the availability of a bigger search log which enable more of the model's suggestions to be matched during evaluation.

The behaviour of the A_SHReC as compared to S_SHReC is similar to the decision stump [16] and its boosted version in machine learning. The decision stump on its own has a very low accuracy but a far better accuracy with boosting, performing better than or equivalent to some of the then state-of-the-art machine learning models. There is generally an increase in evaluation score every week for both A_SHReC and QFG with occasional decreases across the two Intranets. However, the graph of A_SHReC is generally above QFG for the two datasets throughout the test period (see Figures 3 and 4). This supports the fact that A_SHReC's performance is not solely due to its adaptation with search logs since QFG also learns from the same logs. The suggestions from the S_SHReC are therefore useful but their performance is boosted after adaptation with the logs.

The generally low MRR scores might be due to the harsh gold standard (a user's immediate refinement with click) employed in the automated evaluation framework compared to what might be obtained in a user study 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 context might have changed based on the resulting documents shown to the user by the search engine.

Table 4: Average weekly queries for models

	Test	S_SHReC	A_SHReC	QFG
Essex	1340	1067	1181	581
OU	5523	3727	4868	2711

Table 4 shows the average number of weekly queries used in our evaluation and how many of these queries for which the models were able to recommend at least one term for reformulation. A_SHReC is able to provide suggestions about 88% of the time on the average compared to 46% and 74% for QFG and S_SHReC respectively across both datasets. We observe that quite a few of the terms used in creating our initial SHReC model were found directly in the query logs. This is because the difference in the number of queries for which a suggestion was made between A_SHReC and S_SHReC is far smaller than that of QFG. For instance, only an average of 104 (1181 − 1067) new refinements were added into the SHReC model for *Essex* though an average of 581 new refinements were added to QFG every week. This means that the weights on the links between 477 terms already existing in the SHReC and other nodes were strengthened by the adaptation on a weekly basis. An identical trend is observed for *OU*. Nevertheless, users will still not get a suggestion 12% of the time from A_SHReC. This can be reduced further extracting similar terms to those in the graph from dictionaries or domain glossaries.

7.2 Incorporating Clicks for Adaptation

Figures 5 and 6 show results from our evaluation of the query recommendation models for the *Essex* and *OU* where only log refinements with at least one user click are used for SHReC adaptation as well as creation of QFG. The average values are also given in Table 5.

The results trends are very similar to those from the previous section, with A_SHReC significantly better than both S_SHReC and QFG. However, the evaluation scores for both

Table 5: Evaluation scores for adaptation with refinements having user clicks after reformulation

Model	Average MRR (*Essex*)	MRR in 12th week (*Essex*)	Average MRR (*OU*)	MRR in 12th week (*OU*)
S_SHReC	0.00044 (z=-4.54)	0.0004	0.00078 (z=-4.54)	0.00066
A_SHReC	**0.06453** (z=3.11)	0.1204	**0.10078** (z=3.15)	0.15474
QFG	0.05146 (z=1.46)	0.09697	0.08709 (z=1.42)	0.13904

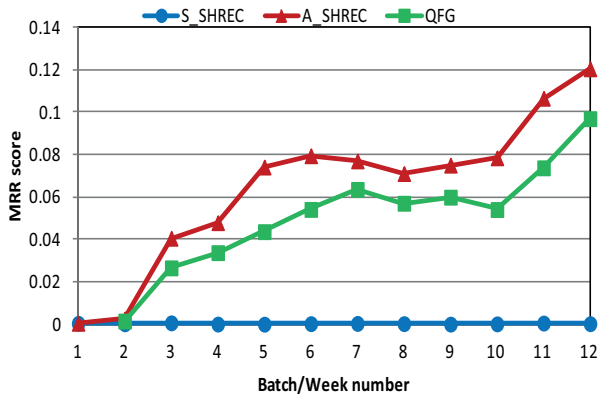

Figure 5: Evaluation results for *Essex* for adaptation with log refinements having user clicks

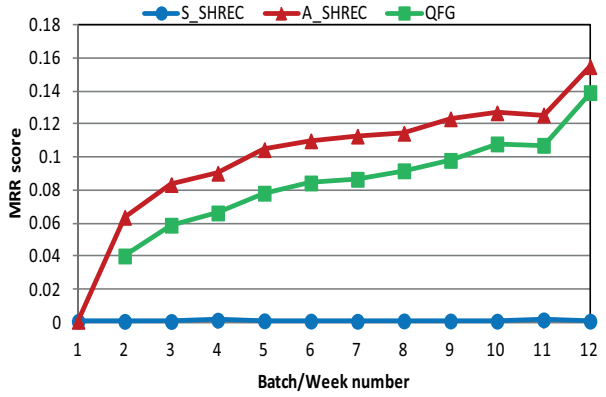

Figure 6: Evaluation results for *OU* for adaptation with log refinements having user clicks

A_SHReC and QFG when only log refinements with user clicks are used for adaptation are smaller than those for which all refinements in the logs are used (see Tables 3 and 5). For instance, the average MRR for *Essex* when adaptation is carried out with all logs is 0.0775 compared to 0.06453 for using only those with clicks. The same applies to *OU* (0.11038 versus 0.10078). This implies that users will benefit more from the models when all log refinements are used for adaptation rather than only those with clicks. Although this initially seems counter intuitive, a good explanation is that users do not necessarily have to click on the resulting documents before modify their queries even though they might have read the result snippets or titles which give hints on how to better formulate their queries.

7.3 Evaluation with click based metrics

A recent study showed that a better retrieval performance is obtained when users are able to reformulate their queries to more successful queries [1]. This was estimated by metrics which rely on click through data such as those introduced in [22]. Furthermore, we also found out that increase or decrease in the retrieval performance of the original query can be a good indication of the quality of query recommendation. This suggests we can automatically assess the quality of query recommendation through these quality metrics before involving real users.

buraries, midwifery courses, masters courses,
payday, foundation courses, copy shop,
credits, academic reference, study skills,
summer term, human rights centre, phd courses,
short courses, language lingistics,graduate diploma,
exam papers, pre sessional course, performing arts,
undergraduate admissions, extenuating cercumstances

Figure 7: Sample twenty (20) queries used for click based quality evaluation

Therefore, we conducted a further evaluation based on that finding to assess the recommendation coming from both A_SHReC and QFG models. The models were first trained (adapted) on all refinements in the log data for *Essex* between October and December 2010. For testing, fifty queries of medium frequency between 5 and 15 are sampled from the logs of *Essex* between February and May 2011. This is similar to the sampling approach in [6] where frequent and rare queries are avoided. We then selected the worst performing 20 queries (see Figure 7[4]) out of these queries using the MRR_C[5] quality metric. Here, MRR_C refers to the mean value of $\sum_i 1/r_i$ of all clicks on each occurrence of the specific query in the logs; r_i is the rank of document clicked within the result list displayed to the user. The intuition behind selecting the least performing queries is that they present a more challenging scenario for the query recommender models as they are the ones where the user is most likely to reformulate. For each of the 20 queries, we selected up to three recommendations from each recommender model, i.e. A_SHReC and QFG. For each model, query q and recommendation r, we calculate the difference $\Delta(MRR_C) = MRR_C(r) - MRR_C(q)$.

The averages are shown in Table 6. We observe that overall both models are capable of recommending queries that can increase the observed retrieval quality estimated by click through data. We also see that the adaptive concept hierarchy model outperformed the query flow graph model reinforcing our previous evaluation.

[4]Some of these queries are actually mis-spellings.
[5]MRR_C should not be confused with the MRR scores used in Equation 4.

Table 6: Retrieval performance differences achieved by recommendations from both models

	Mean $\Delta(MRR_C)$ for top 10 recommendations	$\Delta(MRR_C)$ for top 3 recommendations
A_SHReC	+0.1738	+0.069
QFG	+0.070	+0.059

Remark: Although we discussed only MRR results, other evaluation metrics such as MAP & Recall as well as using only the top ten recommendations gave results with identical trends to the ones discussed in this paper.

8. CONCLUSIONS AND FUTURE WORK

We have proposed a novel method for adaptation of the concept hierarchy model with query refinements from search logs on Intranets for query refinement recommendation. Experimental results indicate that our adaptive model improves in its query recommendation performance over a period of time and is significantly better than its static version as well as the query flow graph which is a state-of-the-art adaptive query recommender model. We also discovered that it was better to adapt the concept hierarchy model with all log refinements irrespective of whether there were user clicks or not. We intend to further validate our automated evaluation results with user studies, incorporate our model in a live system and compare its effectiveness to other query recommendation models.

Acknowledgements

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

9. REFERENCES

[1] M.-D. Albakour, U. Kruschwitz, N. Nanas, I. Adeyanju, D. Song, M. Fasli, and A. De Roeck. Analysis of query reformulations in a search engine of a local web site. In *ECIR'12*, pages 517–521, 2012.

[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'11*, pages 605–610. Springer, 2011.

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

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

[5] R. Baraglia, F. Cacheda, V. Carneiro, D. Fernandez, V. Formoso, R. Perego, and F. Silvestri. Search shortcuts: a new approach to the recommendation of queries. In *RecSys'09*, pages 77–84. ACM, 2009.

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

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

[8] D. Broccolo, L. Marcon, F. M. Nardini, R. Perego, and F. Silvestri. Generating suggestions for queries in the long tail with an inverted index. *Information Processing and Management*, 48:326–339, 2012.

[9] 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.

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

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

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

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

[14] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani. Using association rule to discover search engines related queries. In *Proceedings of the First Latin American Web Congress*, pages 66–71, 2003.

[15] D. Hawking. Enterprise search. In R. Baeza-Yates and B. Ribeiro-Neto, editors, *Modern Information Retrieval*, pages 641–683. Addison-Wesley, 2011.

[16] W. Iba and P. Langley. Induction of one-level decision trees. In *ICML*, pages 233–240. M. Kaufmann, 1992.

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

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

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

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

[21] U. Kruschwitz. *Intelligent Document Retrieval: Exploting Markup Structure*. Springer, 2005.

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

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

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

[25] R. White and R. Chandrasekar. Exploring the use of labels to shortcut search trails. In *SIGIR'10*, pages 811–812. ACM, 2010.

[26] 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. ACM, 2007.

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

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