Moving towards Adaptive Search in Digital Libraries

Udo Kruschwitz[1], M-Dyaa Albakour[1], Jinzhong Niu[1], Johannes Leveling[2], Nikolaos Nanas[3], Yunhyong Kim[4], Dawei Song[4], Maria Fasli[1], and Anne De Roeck[5]

[1]University of Essex, Colchester, UK
{udo, malbak}@essex.ac.uk
[2]Dublin City University, Dublin, Ireland
[3]Centre for Research and Technology, Thessaly, Greece
[4]Robert Gordon University, Aberdeen, UK
[5]Open University, Milton Keynes, UK

Abstract. Search applications have become very popular over the last two decades, one of the main drivers being the advent of the Web. Nevertheless, searching on the Web is very different to searching on smaller, often more structured collections such as digital libraries, local Web sites, and intranets. One way of helping the searcher locating the right information for a specific information need in such a collection is by providing well-structured domain knowledge to assist query modification and navigation. There are two main challenges which we will both address in this chapter: acquiring the domain knowledge and adapting it automatically to the specific interests of the user community. We will outline how in digital libraries a domain model can automatically be acquired using search engine query logs and how it can be continuously updated using methods resembling ant colony behaviour.

1 Introduction

Document retrieval systems have been around for more than fifty years, and early systems exploited similar structures to those we have in modern digital libraries, such as author name, book title, and keywords [33]. With the advent of the Web things have changed however and searchers are now very used to simple search interfaces that take a few keywords and return a list of matches. In fact, this is the type of search paradigm we might expect nowadays no matter what collection is being searched for. The problem is that Web search is fundamentally different to searches where users are not just interested in getting *some* matching documents but where they are looking for specific documents, memos, spreadsheets, books, etc. Such information requests are not necessarily best served by a single-shot unstructured query. This type of search is very common in enterprise search which runs on smaller, often more structured collections [21, 48, 39]. This suggests that search over digital libraries with their inherently structured contents resembles enterprise search much more than generic Web search and we argue that offering some guidance in an interactive search process could actively help the user find the actual documents he or she is after.

How can a user be guided in the search process? Library classification schemes like the *Universal Decimal Classification*[1] (UDC) have been used for decades and have

[1] http://www.udcc.org/

been demonstrated to be very useful when classifying books. The drawback that these manually encoded classification schemes have is that they lack flexibility. Furthermore, they represent a structured view of the world but that view may not be the view that an *online* searcher has. Suppose, a university's digital library contains a large number of books on information retrieval. They might all be classified under the same code but this will not tell us anything about their relevance or about how users would associate them with other books. We could on the other hand rely on automatically acquired knowledge structures (e.g. domain models, taxonomies, association graphs etc) derived from the document collection. But again, without continuously updating the models they will become out of date as the document collection changes or the users start to view it differently. For example, new concepts are introduced, others disappear and the books that are popular today may not have the same relevance in half a year's time.

The approach that we take is to use log data to build an adaptive domain model automatically. We are looking at search as well as navigation and our aim is to satisfy a user's information request effectively by learning from the entire user population and incorporating this learned knowledge in a constantly adapting domain model. This domain model would assist a user in the search process and reflect the collection characteristics. This is different from building *individual* user profiles. In other words, we exploit the "wisdom of the crowd" to build up knowledge structures automatically and update them constantly as new queries come in. Unlike UDC, the emerging structures are not semantically encoded. They will however encode relations between query terms that reflect how users see and navigate the collection and should represent a bridge between the users' vocabulary and the contents of the collection. To use the earlier example, a user who searches for *"information retrieval"* might be given suggestions to narrow down (or modify) the search such as *"rijsbergen"*, *"bruce croft"*, *"modern information retrieval 2nd edition"* or *"manning and schütze"*. This will allow users to benefit from each other by incorporating *social search* in digital libraries *without* trying to semantically interpret the actual relationships that might hold between a query and its refinement suggestions.

The chapter will be structured as follows. We will start with an overview of some related work (Section 2) before formulating our research questions (Section 3). Section 4 will focus on the domain model construction process and will outline how we use an ant colony optimization algorithm which keeps the domain model in a continuous update cycle. In Section 5 we will describe the log files we are going to use to build the domain model. These log files represent real user needs as they have been collected on the search engine of a digital library catalogue.The experimental setup is explained in Section 6. Results are presented in Section 7 which is followed by a discussion (Section 8). We will finish with some conclusions and an outlook on future work.

2 Related Work

Many ideas have been proposed to address the problem of information overload when searching or exploring a document collection. One very promising route is *Social Search* which combines ideas from personalization and social networking so that a searcher can benefit from past users' search experiences [41]. Applied to the digital libraries context,

this idea can also be expressed as *Social Navigation* which adds a social dimension to browsing by guiding future users with the navigation experiences learned collectively from the crowd [12]. The question is what search trails and information should be exploited in this process. Utilizing explicit user judgements about items or search terms seems to be most useful. The problem is however that users are reluctant to leave any explicit feedback when they search a document collection [34]. Nevertheless, implicit feedback, e.g., the analysis of log records, has been shown to be good at approximating explicit feedback. There is a wealth of related work in log analysis, interactive search and other areas [24, 40]. For example, users often reformulate their query and such patterns can help in learning an improved ranking function [26]. The same methods have shown to improve an adaptive domain model on a local Web site [32]. Log analysis has in fact developed into an entire research strand and it has been widely recognised that query log files represent a good source for capturing implicit user feedback [24, 40].

The next question is how such feedback should be applied to improve the search process. One possibility is to exploit it in order to build knowledge structures that can assist in interactive search. But do users want assisted search? First of all, digital libraries are characterized by much more structured knowledge than Web sites. This makes system-guided search a natural option as evidenced by the success of Aquabrowser[2] as a tool to access digital libraries. More generally though, there is also evidence that users want support in proposing keywords but they ultimately want to stay in control about what is being submitted as a query [50]. Furthermore, despite the risk of offering irrelevant suggestions in a system-guided search system, users might prefer having them rather than not [49]. On the other hand, it has also been shown that users are more inclined to submit new queries or resubmit modified queries than to navigate from the result set in a search environment that supports search and navigation [35]. Perhaps the best evidence for the usefulness of interactive search systems is the fact that even the big Web search engines have recently added more and more interactive features, e.g., Google's Wonderwheel[3].

Belkin calls the move beyond the limited, inherently non-interactive models of IR to truly interactive systems the *challenge of all challenges* in IR at the moment [9]. This is in line with what we propose, i.e. to go beyond static interaction patterns and move to adaptive retrieval exploiting the implicit feedback that users leave when searching and navigating a document collection. Building adaptive domain models for digital libraries and other collections is our approach to capturing and utilizing "collective intelligence" [45].

We wish to build a model that captures user interactions with a digital library and consolidates them to provide a dynamic model that will enable the combined knowledge to be examined e.g., a *learning network* in which algorithms build and extend network representations by acquiring knowledge from examples [43], in that we wish to capture user experience to update the model. One motivation could be that a large proportion of queries submitted to a search engine can be exact repeats of a query issued earlier by the same user [46]. However, our main motivation is to use the model to help make suggestions that can be used by other users.

[2] `http://www.serialssolutions.com/aquabrowser/`
[3] `http://www.googlewonderwheel.com`

There are many different ways of structuring such models. Models can be built by extracting term relations from documents, e.g. [38, 31, 51], or from the actual queries that users submit to search the collection by building query flow graphs, e.g. [11], or mining term association rules [16]. Past user queries appear to be preferred by users when compared to terms extracted from documents [29], which is one motivation for using log files in our work. Various Web log studies have been conducted in recent years to study the users' search behaviour, e.g. [5, 47, 14, 23], and log files have widely been used to extract meaningful knowledge, e.g. relations between queries [7], or to derive query substitutions [28]. Much of this work however is based on queries submitted on the *Web* and thus presents a very broad view of the world. Our work is different in that we start with a specific document collection for which suitable knowledge structures are typically not readily available (that collection could but does not have to be a digital library), extract relations from queries submitted within this collection to build and *evolve* a domain model automatically. It has been demonstrated that such an approach has the potential to learn useful relations over time in an intranet environment [15].

Digital libraries are however much more structured and represent a very different type of collection compared to the Web as a whole, a local Web site or an intranet. The question is whether domain knowledge can be acquired automatically from user queries within digital libraries, whether such relations can be improved over time and how this compares to alternative approaches. This leads us to our research questions.

3 Research Questions

The research questions we are trying to answer are as follows:

1. Can we employ the paradigm of the "wisdom of the crowd" to log files of digital libraries to extract useful query term relations that can assist in searching the collection?
2. How do these relations compare to sensible baseline approaches?
3. How do log files collected on digital libraries compare with intranet logs?

4 The Domain Model

Our domain model takes the form of a graph structure where nodes are query phrases and edges represent possible query refinements, higher weights denoting more common selections. Figure 1 gives an example of part of the domain model as it has been derived from our log data. Of inspiration for this model is the Nootropia system [36] for user profiling. This determines hierarchies of terms and disseminates energy using a method based on Artificial Immune Systems. We, however, take a related, if conceptually opposite method, to provide a model based on a *consolidated user* as opposed to learning differences between individuals.

As a reminder, the relation between two terms in the model is purely some form of association link that has been extracted from the logs and is therefore different from

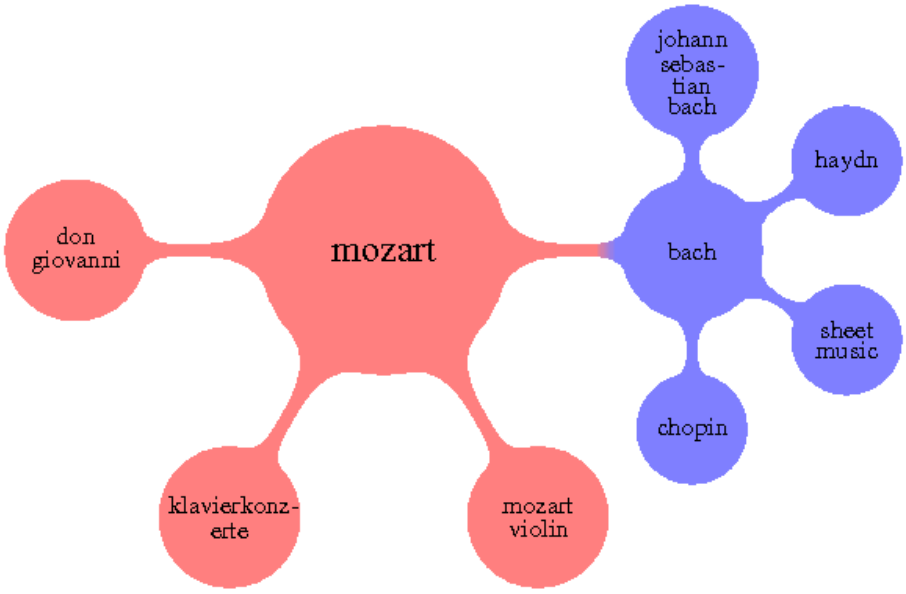

Fig. 1. Partial Domain Model Derived from TEL Log Data.

(and complementary to) the use of semantically encoded relations as used, for example, in the *Europeana* project[4], or the use of controlled vocabularies, e.g. [18, 10]

Using such an internal representation allows numerous potential display and interrogation techniques to be presented to the user. In this work we focus on using relations encoded in the model as query modification suggestions in guided search. Applied differently, the domain model could also be used to browse or navigate the collection.

A range of adaptation algorithms have been developed but models that are able to capture *evolving* trends in search query graphs are only just starting to emerge, e.g. [8]. We will use the analogy of ant colony optimization (ACO) to first populate and then adapt the graph. The user traverses a portion of the graph by using query refinements (analogous to the ant's journey), the weights on this route are reinforced (increasing the level of pheromone). Over time all weights are reduced by a set proportion (pheromone evaporation). To reduce noise we only associate immediate refinements, e.g., for a session containing a query modification chain q_1 to q_4, associations will be created between q_1 and q_2, q_2 and q_3, and q_3 and q_4 only (see Algorithm 1). The specifics are as follows:

– At the end of each day all edge weights are normalised to sum to 1 and the mean weight of all edges is then calculated.

[4] `http://eculture.cs.vu.nl/europeana/session/search`

Algorithm 1: The ACO-based algorithm to build and evolve the domain model.

Input: domain model as a graph G, daily query log L, number of days DAY_NUMS
Output: G

```
1  τ ← 1
2  for d ← 1 to DAY_NUMS do
       /* update weights of traversed edges                    */
3      foreach (q, q′) ∈ L_d do
           /* Query q′ immediately follows q in a session on day
              d.                                                */
4          n ← FindNode(G, q)
5          if n = NULL then n ← AddNode(G, q)
6          n′ ← FindNode(G, q′)
7          if n′ = NULL then n′ ← AddNode(G, q′)
8          e ← FindEdge(G, n, n′)
9          if e = NULL then
10             e ← AddEdge(G, n, n′)
11             SetWeight(G, e, τ)
12         else
13             SetWeight(G, e, τ + GetWeight(G, e))
       /* normalize weights of edges                           */
14     T ←+ TotalWeights(G)
15     c_e ← 0
16     foreach e ∈ G do
17         c_e ← c_e + 1
18         SetWeight(G, e, GetWeight(G, e) / T)
19     τ ← T / c_e
```

– For the next day, all queries in the log are extracted for that day where there are multiple queries in a particular user session.
– The queries are then time ordered and for each query phrase that follows an earlier phrase in the session an edge is created, or updated if it already exists, by the mean association weight of the previous day.
– A nominal update value of 1 is used for our first day, however, any positive real number could have been chosen without affecting the outcome of normalisation.

By normalising the weights at the end of each day we reduce the weight of non-traversed edges, hence, over time, penalising incorrect or less relevant phrase refinements. In addition we expect outdated terms to be effectively removed from the model, i.e., the refinement weight will become so low that the phrase will never be recommended to the user.

One would expect to use the model to provide suggested terms by first finding the original query phrase in the graph, then list the linked terms ordered by weight. To use an example, if a user searches for *"mozart"*, then the domain model can propose query modifications such as *"don giovanni"*, *"klavierkonzerte"*, *"mozart violin"* and *"bach"*.

Although not addressed in this chapter, indirect associations could also be used when data is sparse, or if we wish to investigate sub-trees with relatively high weights.

Although we have chosen to run the update in the described algorithm on a daily basis, update sessions could be run hourly or weekly, or even when a certain number of user sessions have completed. In addition, it is possible to run the algorithm from any point in the user log to any other, this allows us to compare how the model performs for particular time periods.

5 Log Files

We have used log data that have been collected on the search engine of The European Library (TEL)[5]. The TEL logs contain an entry for every user interaction with the TEL portal. Log entries contain the type of action performed (e.g. simple or advanced search, changing system options) and attributes such as user ID, session ID, the interface language, query, and timestamp. Figure 2 lists five sample entries, the first one describing a search in English for *"pomegranate fertilization"* submitted through the simple user interface.

```
...
903779;guest;83.33.xxx.xxx;83et8b7j010eh4vlht3ucj8dl1;en;
       ("pomegranate fertilization");search_sim;;0;-;;;2007-10-05 13:52:30
...
1889115;guest;71.249.xxx.xxx;8eb3bdv3odg9jncd71u0s2aff6;en;
        ("mozart");search_url;;0;-;;;2008-06-24 22:02:52
...
1889118;guest;71.249.xxx.xxx;8eb3bdv3odg9jncd71u0s2aff6;en;
        ("mozart");view_full;;1;;;;2008-06-24 22:03:03
...
1889120;guest;71.249.xxx.xxx;8eb3bdv3odg9jncd71u0s2aff6;en;
        Klavierkonzerte;search_res_rec_all;;0;-;;;2008-06-24 22:03:55
1889121;guest;71.249.xxx.xxx;8eb3bdv3odg9jncd71u0s2aff6;en;
        ("klavierkonzerte");view_full;;1;;;;2008-06-24 22:04:10
...
```

Fig. 2. Sample entries in the TEL log.

The logs record not just all queries submitted to the search engine but also other activities such as viewing a result. We are only interested in search queries (which make up about a quarter of all actions). We use the log file that has also been used in LogCLEF 2009 and 2010[6]. This log covers the period from 1 January 2007 till 30 June 2008 [2]. In the logs there is a great inclination towards using simple search compared to using advanced search [19]. In our experiments we do not consider queries submitted via the advanced search interface. We use the session numbers recorded in the log files to identify search sessions.

We processed the files as follows:

[5] http://www.theeuropeanlibrary.org
[6] http://www.uni-hildesheim.de/logclef/

1. Discard all actions that are not simple search queries
2. Remove all queries that do not have English specified as the query language
3. Remove all queries that contain non-ASCII characters
4. Case-fold all queries, replace all non-alphanumeric characters by space
5. If a query contains one or more Boolean operators, trim the query so that the left-most operator and everything that follows gets removed.
6. Delete all queries which have no session number specified
7. Finally, only keep those sessions that consist of at least two search queries.

The last point in particular reduces the number of selected queries dramatically because there is a large proportion of sessions that involve only a single user query. We end up with 152,863 queries. Figure 3 presents two sample entries in the processed query logs.

```
...
8eb3bdv3odg9jncd71u0s2aff6 xxxx 1889115 xxxx mozart xxxx 2008-06-24 22:02:52
8eb3bdv3odg9jncd71u0s2aff6 xxxx 1889120 xxxx klavierkonzerte xxxx 2008-06-24 22:03:55
...
```

Fig. 3. Sample session records after processing the TEL logs.

Note that we make the simplifying assumption that all queries within a session are related to the same search request. This is not always true and a session can easily consist of a number of *search goals* and *search missions* [27]. However, identifying exact session boundaries automatically is an inherently difficult task [20, 22].

Finally, we use the processed log file to build and adapt the domain model. The log records are ordered by session and in chronological order. Each consecutive query pair within a session is processed as outlined in Algorithm 1.

6 Experimental Setup

We assume that a high-quality domain model is one that makes sensible suggestions to the user. We employ two evaluation methods to assess the quality of the domain model. Our first evaluation method, *AutoEval*, is fully automated, in the second evaluation method we asked users to assess the quality of domain model relations that have been learned. These methods aim at evaluating the quality of term relations that emerge from the adaptation process.

As part of the automated evaluation we conducted two sets of experiments, one using all queries submitted to TEL. The second experiment only looked at frequently submitted queries. The first experiment will tell us how well the algorithm learns the relations covering the entire domain. The second approach targets high-frequency queries only. The reasoning behind this is that an interactive search system might go for either high recall (offer suggestions whenever there is any relation in the domain model, i.e. cover all possible queries) or for high precision (only suggest “reliable” terms, i.e. focus on highly frequent queries only). The high recall approach runs the risk of offering a lot

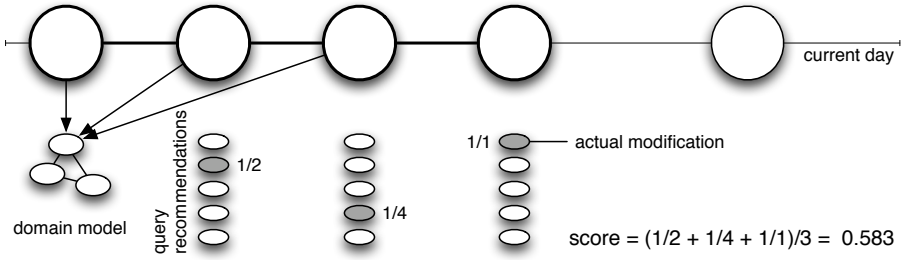

Fig. 4. Daily Model Evaluation

of noise, the other approach will not offer any suggestions for the long tail of infrequent queries.

For the second evaluation which involved actual assessors we only looked at frequent queries as we will discuss in more detail further down.

Clearly, our experimental settings are necessarily approximations of the real world. They will only be able to give us an indication of the usefulness of term relations learned from the log files. Any such findings will need to be validated by large-scale experiments that are used in realistic user search tasks. This will be left as future work.

6.1 Automatic Evaluation Method: AutoEval

Our first evaluation method, *AutoEval* [4], is based on the idea that we can assess the quality of a domain model by comparing suggestions derived from the model to query modifications actually observed in the log files. We use Mean Reciprocal Rank (MRR) to measure this. Given some initial search request, if a query modification observed in a session matches the suggestion derived from the model, we will reward the model, the highest reward is paid for a suggestion that comes top in the model, smaller rewards (MRR) for suggestions further down the list.

The model's evaluation is performed on an arbitrary interval basis as depicted in Figure 4 where the evaluation takes place on a daily basis. It only takes place for days with at least one query modification pair. For example, let us assume that during the current day, three query modifications have been submitted (Fig. 4). For each query modification pair, the domain model is provided with the initial query and returns a ranked list of recommended query modifications. We take the rank of the actual modified query (i.e., the one in the log data) in this list, as an indication of the domain model's accuracy. The assumption here is that an accurate domain model should be able to propose the most appropriate query modification at the top of the list of recommended modifications. This is based on the observation that users are much more likely to click on the top results of a ranked list than to select something further down [25], and it seems reasonable to assume that such a preference is valid not just for ranked lists of search results but for lists of query modification suggestions as well.

So for the total of three query modifications in the current day, we can calculate the model's accuracy score as $(1/r_1 + 1/r_2 + 1/r_3)/3$, where r_1 to r_3 are the ranks of the

actual query modifications in the list of modifications recommended by the model in each of the three cases. In the figure's example the model's score would be 0.583. More generally, given a day d with Q query modification pairs, the model's Mean Reciprocal Rank score MRR_d for that day is given by Equation 1 below.

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

Note that in the special case where the actual query modification is not included in the list of recommended modifications $1/r$ is set to zero. The above evaluation process results in an accuracy score for each logged day for which at least a query modification pair exists. So overall, the process produces a series of scores for each domain model being evaluated. These scores allow the comparison between different domain models. A model M_1 can therefore be considered superior over a model M_2 if a statistically significant improvement can be measured over the given period.

The described process fits perfectly a static model, but in the case of dynamic experiments as we are conducting here, the experimental process is similar. We start with an initially empty domain model, or an existing domain model. Like before, the model is evaluated at the end of each daily batch of query modifications, but unlike the static experiments it uses the daily data for updating its structure. This is essentially a continuous learning problem, where the domain model has to learn from temporal query modification data (applying the ACO algorithm in our specific example). Again, we treat a model as superior over another (possibly static one) if an improvement can be observed that is significant.

When testing our ACO algorithm we decided to compare the results against a simple alternative based on association rules [17]. Fonseca's approach represents a sensible baseline for a different way of adapting the search because it accesses exactly the same resources as our proposed method and it has been shown to work well on Web log data. The idea is to use session boundaries and to treat each session as a transaction. Related queries are derived from queries submitted within the same transaction.

6.2 User-based Evaluation Method: Mechanical Turk

The next evaluation was user-based to find out how users would assess the relevance of query modification suggestions learned by the adaptive model and how they compare against alternative approaches for constructing such suggestions. To do so we adopted a methodology proposed in the literature [38]. An online form was prepared, and participants were asked to determine whether queries and their refinements were relevant.

To avoid data sparsity issues we used the top 20 most frequently submitted queries as found in our processed log files (see Table 1) to derive suggestions. For each query we selected the three best (highest weighted) related terms using three different methods:

- **ACO**: For each query we selected the three top suggestions that have been learned after running the full log file through our ant colony optimization algorithm.
- **Fonseca**: Applying the same methods as in the *AutoEval* run, we selected the three top association rules for the given query applied to the full log file.

Rank	Query Phrase	Rank	Query Phrase
1	mozart	11	dante
2	harry potter	12	zagreb
3	meisje met de parel	13	bible
4	einstein	14	poland
5	shakespeare	15	history
6	bach	16	france
7	music	17	chopin
8	europe	18	paris
9	goethe	19	italy
10	london	20	cervantes

Table 1. Most frequent queries.

– **Baseline**: As a baseline we selected a method that does not rely on log data. We assume that the top matching results of a commercial search engine will be a useful resource to derive query modification suggestions. To restrict the results to a collection comparable to the digital library catalogue at hand we decided to search only for matches within the world library catalogue WorldCat[7]. We derived nouns and noun phrases from the top ten snippets returned by *Yahoo!* (restricting the search to the WorldCat Web site). We identify nouns and noun phrases using text processing methods applied in previous experiments, e.g. [31].

Therefore users had to judge 60 individual query suggestions derived for each of the three methods.

We recruited assessors using Amazon Mechanical Turk[8], a crowdsourcing market place that has been shown to work effectively, and it has been demonstrated that its aggregated results approximate expert judgement for a variety of tasks, e.g. [42, 13, 3]. Obviously, the recruited users might never use the digital library search functionality of TEL, but they do represent potential users as anybody can access the TEL portal and we wanted to learn to what extent potential users would find term suggestions extracted from the domain model useful *if* they were searching a digital library.

CrowdFlower[9] was used to build the assessment task and control access to Amazon Mechanical Turk. The task was built as an online form similar to the one used in [30], where assessors were asked to determine whether queries and their refinements were relevant.

The instructions gave the users a hypothetical context as follows:

> *Suppose that you are a user of a digital library's search engine. The digital library allows you to access all collections of libraries available on the Internet worldwide. You issue queries on the search engine to find what you are looking for. In addition to returning the best matching books or articles for any given*

[7] `http://worldcat.org`
[8] `http://www.mturk.com`
[9] `http://www.crowdflower.com`

query, this search engine also suggests modified queries that you could use to refine or replace the original one.
The form below gives a list of term pairs. For each pair, imagine the first term was your original query, and that the second is one of the terms proposed by the search system, which you could use to refine or replace the search. Please judge for each pair whether you think the second term is:
- *relevant (Choose 'Relevant').*
- *not relevant (Choose 'Not Relevant').*
- *If you do not know, then choose 'Don't know'.*

Here, 'relevant' means that you can imagine a situation where the second term is an appropriate refinement or replacement of the query given by the first term.

We also pointed out that if they found it difficult to judge the pair, that they might want to consult some online resources, e.g.Wikipedia or The European Library.

Subjects were not told that various different techniques have been used to generate these query pairs. The form contained a list of all query pairs in random order.

We asked 20 Mechanical Turk workers to do the assessment task and restricted the location of those workers to be UK-based. The reason for this restriction is that we know that UK searchers form a significant proportion of actual TEL users [2].

We paid 2 US dollars for each assessment task.

7 Results

For all significance testing we used paired t-tests (where appropriate) with confidence value $p<0.001$ unless otherwise specified.

7.1 AutoEval Results

Figure 5 illustrates the results of applying *AutoEval* over the entire period covered by the log file. We use monthly batches to update the domain model. Fonseca's association rules approach was evaluated with different settings. The minimum support parameter (MinSup) is the threshold used to infer an association. Fonseca *et al.* conducted their experiments on Web log data using *MinSup=3* [17]. However, due to the much smaller data set we are dealing with here we also provide a run using a weaker support of *MinSup=2* (in other words association rules may be selected even if the query pair has only been found in two sessions). A lower minimum support therefore increases the chance of inferring an association for any given query.

The main observation is that our ACO method is significantly more effective than learning based on association rules (with either minimum support setting). We see that despite a few spikes the general trend is upwards indicating that our adaptive learning method (and to a smaller extend the association rules) are able to learn from past log data over time. However, we also observe that the absolute score is relatively low and we will get back to that later.

The second round of *AutoEval* runs considered only the top 20 most frequent queries extracted from the query logs. Those queries are listed in Table 1. Note that these are

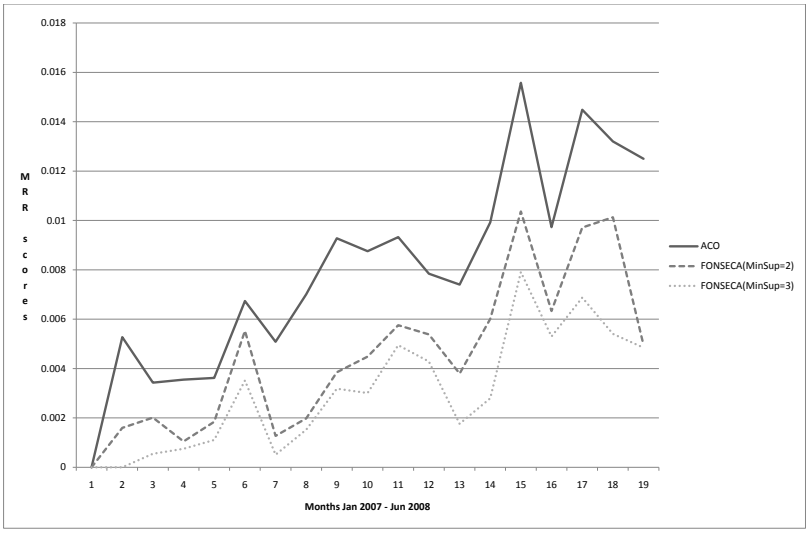

Fig. 5. ACO *vs.* Fonseca-Baseline.

the most frequent queries in our processed log files. One frequent query ("*sange for claveret*") was not considered because the baseline failed to produce a single result.

In this case, the MRR scores in Equation 1 were calculated for query modification pairs Q_{top} where the first query in the pair is one of those listed in Table 1. Figure 6 displays the obtained scores for ACO and Fonseca's association rules when we restrict the evaluation to the top 20 queries. The average scores are much higher (and for ACO they remain at a higher level throughout the learning period). Again we observe that ACO is better on average over the entire period ($0.041 > 0.035$) but there is no significant difference.

7.2 User Assessment Results

The assessments obtained from Mechanical Turk were aggregated and the results are shown in Table 2. For each user we calculated the percentage of pairs that were judged relevant using different criteria and then we aggregated the results among the 20 assessors. The 'Total Relevant' row gives the average judgement of query pairs considered relevant over all users and all three suggestions for each method. If we only take into account the top suggestion (i.e. the one highest ranked by the corresponding method) for each query, then we get the results listed under 'First Relevant'. Finally, the percentage for which the system in question had provided at least one suggestion that was judged relevant is shown in the bottom row ('At Least One Relevant').

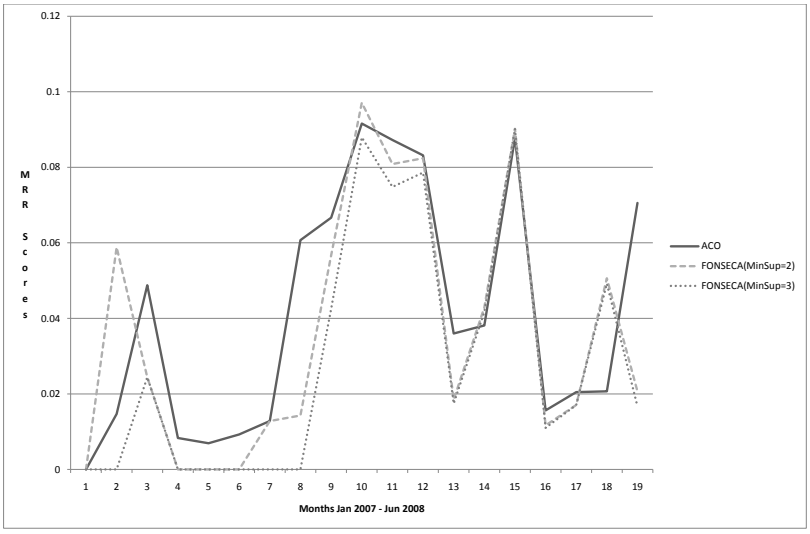

Fig. 6. ACO *vs.* Fonseca-Baseline (Frequent Queries).

	Ant Colony Optimization	**Fonseca's Association Rules**	**Snippet Baseline**
Total Relevant	45.67%	44.08%	57.17%
First Relevant	51.75%	47.75%	58.25%
At Least One Relevant	74.25%	76.00%	84.50%

Table 2. A comparison of user-judged relevant query suggestions for the 20 most frequent TEL queries generated by three different systems.

The user assessment indicates that ACO and Fonseca's association rules have a comparable performance on top queries although ACO is slightly better in learning query suggestions from query logs. This is in line with the *AutoEval* scores shown in Figure 6 as we see an improvement though not statistically measurable. The user assessment also shows that both learning methods were considered less effective than the baseline approach.[10] We will discuss the results in the next section.

Note, that for Fonseca's association rules we set the minimum support parameter to 2 in this experiment. The reason for that is we have already shown in the *AutoEval* experiments that this value yields a better performance. More importantly, when we

[10] The results closely correlate with an assessment that was conducted by an independent digital libraries expert and can be seen as another successful example of obtaining expert judgements by exploiting the wisdom of the crowd.

increase this value to 3 no refinements were actually generated for some of the most frequent queries.

8 Discussion

The main finding is that our continuous learning model is capable of learning useful relations from digital library query logs as evidenced by the results of the *AutoEval* runs. Using the automatic evaluation over the entire logs has furthermore shown the superiority of the ACO approach over a method that extracts association rules from query pairs found in the same session. This superiority is less measurable when learning on top queries only.

However, when discussing the results we need to start with an interesting observation regarding data sparsity. Queries submitted to search engines approximately follow the power-law distribution [6]. That means that we can capture a large proportion of user requests with a relatively small number of unique queries. Now, we found that TEL queries are particularly sparse. Whereas the top 20 most frequently submitted queries on a university Web site can make up as much as 15% of the entire query corpus [15], we observe that in a TEL log file of a comparable size to the university log the top 20 queries only cover about 2% of all queries submitted.[11] In that respect, the TEL queries appear to be more similar to Web queries than intranet queries [44]. The difference of course is that Web logs are magnitudes larger which means that the actual count of even less frequent Web queries may still be large. In our processed query logs we found the most frequent query occurs only 414 times and the 20th-most-frequent query only 70 times! Learning on the TEL logs is therefore particularly challenging. This is reflected in the low *MRR* scores we obtained in the automatic evaluation and possibly also in the assessors' relevance judgements. The sparsity of the data and the open nature of the library domain makes it harder to learn useful query refinement suggestions from the logs.

One possible way forward is to make use of more of the log data (we reduced the log file of more than one million interactions to a fraction of the size). What we have shown here is that the general idea of an adaptive domain model gives very promising results. A more customized and fine-tuned algorithm is likely to improve the learning rate.

Regarding our experiments with top queries, we argue that even after deleting known test queries we still find a number of queries that are unlikely to be typical user requests. One example is *"sange for claveret"* which is frequent but only delivers a single result (from the Danish national library). Furthermore, there are a significant number of sessions in which *"sange for claveret"* co-occurs with other frequent (and perhaps unusual queries) such as *"meisje met de parel"*.

In any case, the experiments we conducted on the top queries nicely correlate with the earlier experiments. On *AutoEval* we see that the ACO method slightly outperforms an association-based approach and this is mirrored when asking users to assess the top three (or the top one) suggestions derived from each model. The user assessments also

[11] For this comparison we used another TEL log file used in LogCLEF covering the period from January 2009 till December 2009.

tell us that almost half of the query suggestions derived by ACO for top queries are considered relevant. Another finding is that users found the query refinement suggestions provided by the baseline more relevant than the ones learned from the logs. This confirms other experiments run on the same data [1]. Interestingly however, the suggestions derived from the logs and the ones found in the snippets appear to be complementary. In fact, there is only a 3% overlap when considering the suggestions derived from the query log learning approaches (ACO and association rules) on the one hand and the snippets baseline on the other hand. For example, for the query *"europe"*, ACO has learned *"europe map"* as the refinement with the highest weight. The top baseline suggestion is *"council of europe"*. Both suggestions were considered relevant by 90% of our assessors.

To put the results in context, running ACO on a university intranet log file results in relations that are considered more relevant by users (above 60% when considering all suggestions or just the top one) [15]. We argue that this is largely due to the aforementioned data sparsity issue.

9 Conclusions

In this chapter we outlined how log files collected on a digital library portal can be exploited to learn query suggestions which can assist users of the portal. We shall now return to our research questions and will try to draw conclusions based on the results we obtained.

1. *Can we employ the paradigm of the "wisdom of the crowd" to log files of digital libraries to extract useful query term relations that can assist in searching the collection?*
   We have demonstrated that an ant colony optimization learning algorithm is capable of learning useful query relations over time. There is certainly a lot of hidden knowledge in log files of digital libraries that should allow us to move towards adaptive system-guided systems in digital libraries.
2. *How do these relations compare to sensible baseline approaches?*
   A sensible baseline approach that does not rely on log data can be difficult to beat. However, the suggestions derived from query logs and those derived dynamically from top matching documents appear to complement each other.
3. *How do log files collected on digital libraries compare with intranet logs?*
   An important finding of this study related to our third question is that digital library logs appear to be much more sparse than for example search engine logs of an intranet or a local Web site. This in itself might not be surprising but the implication is that an effective learning algorithm will either have to rely on large log files or will have to exploit the logs much more effectively than what is needed when extracting relations from intranet or Web logs.

An additional conclusion we would like to draw is that our automatic evaluation methodology *AutoEval* has been shown to be a useful evaluation framework to compare the performance of different approaches for building domain models to provide query suggestions (in the fairly restricted evaluation settings employed here).

10 Future Work

There are a number of areas that will need to be addressed in future. First of all, we have so far not involved *real* users in *realistic* search tasks. Furthermore, and related to that point, in our experiments we simplified the evaluation task by collecting user judgements once only at a fixed point in time. To get more realistic assessments the adaptive algorithms need to be incorporated in a live search engine of a digital library catalogue to allow longitudinal studies. In any case, we would assume that an automatically acquired (and evolving) domain model will allow users to find relevant information more quickly and allow a better navigation experience over time, in particular when deriving query modification suggestions using a variety of approaches. These experiments are on our agenda for future work.

One way of addressing data sparsity is to extract more knowledge from the logs. So far we kept the pre-processing of the log files deliberately simple. A large proportion of queries do however make reference to specific fields in the structured data entries (e.g. author name, topic etc). A natural modification to the simple domain model building described in this chapter will make more use of the query structure as well as other actions recored in the logs. Furthermore, our domain model is not linked to the actual documents in the collection. By using clickthrough information it will be straightforward to link the model into the collection.

Regarding the ACO algorithm, the approach we applied to automatically adapt the domain model is the simplest possible way of using ACO in this context. Here we assume that the weight of a relation between two queries increases as soon as this query pair is observed, or to use the ants analogy, a pheromone trail is left every time an ant moves from A to B. However, we could modify that in a number of ways. Pheromone might, for example, only be placed if the ant has discovered some valuable resource. Applying this to search we could strengthen the relation only between any query that is part of a session and the final query given that this final query is followed by viewing some result set as the final action. Similar ideas have been shown to work effectively when associating queries with landing pages in Web search [37].

Acknowledgements

This research is part of the AutoAdapt research project. AutoAdapt is funded by EPSRC grants EP/F035357/1 and EP/F035705/1. We would also like to thank Sally Chambers and The European Library as well the organisers of the LogCLEF track for providing the log files, in particular we would like to thank Thomas Mandl. Without realistic log data we would not have been able to conduct this research. We would also like to thank Vivien Petras for helping with the expert assessments and finally we thank the anonymous reviewers for valuable feedback.

This material is based upon works supported by the Science Foundation Ireland under Grant No. Grant 07/CE/I1142.

References

1. M. Agosti, D. Cisco, G. M. Di Nunzio, I. Masiero, and M. Melucci. i-TEL-u: A Query Suggestion Tool for Integrating Heterogeneous Contexts in a Digital Library. In *Research and Advanced Technology for Digital Libraries*, volume 6273 of *Lecture Notes in Computer Science*, pages 397–400. Springer Berlin / Heidelberg, 2010.
2. M. Agosti, F. Crivellari, G. M. Di Nunzio, Y. Ioannidis, L. Stamatogiannakis, M.-L. Triantafillidi, and M. Vayanou. Report on Search Engines and HTTP Log Analysis. Technical report TELplus D5.2, TELplus Project, 2009.
3. M.-D. Albakour, U. Kruschwitz, and S. Lucas. Sentence-level attachment prediction. In *Proceedings of the 1st Information Retrieval Facility Conference*, volume 6107 of *Lecture Notes in Computer Science*, pages 6–19, Vienna, 2010. Springer.
4. 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 *Proceedings of the 33rd European Conference on Information Retrieval (ECIR'11)*, Dublin, 2011.
5. P. Anick. Using Terminological Feedback for Web Search Refinement - A Log-based Study. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 88–95, Toronto, Canada, 2003.
6. R. Baeza-Yates and F. Saint-Jean. A Three Level Search Engine Index Based in Query Log Distribution. In *Proceedings of the 10th International Symposium on String Processing and Information Retrieval (SPIRE)*, Lecture Notes in Computer Science 2857, pages 56–65, Manaus, Brazil, 2003.
7. R. Baeza-Yates and A. Tiberi. Extracting semantic relations from query logs. In *Proceeding of the 13th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 76–85, San Jose, California, 2007.
8. R. Baraglia, C. Castillo, D. Donato, F. M. Nardini, R. Perego, and F. Silvestri. The Effects of Time on Query Flow Graph-based Models for Query Suggestion. In *Proceedings of RIAO'2010*, Paris, 2010.
9. N. J. Belkin. Some(what) grand challenges for information retrieval. *SIGIR Forum*, 42(1):47–54, 2008.
10. B. Berghaus, T. Mandl, C. Womser-Hacker, and M. Kluck. An entry vocabulary module for a political science test collection. In *Business Information Systems*, Lecture Notes in Business Information Processing, pages 1–11, 2008.
11. P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *Proceedings of the 2009 Workshop on Web Search Click Data (WSCD'09)*, pages 56–63, 2009.
12. P. Brusilovsky, L. Cassel, L. Delcambre, E. Fox, R. Furuta, D. Garcia, F. Shipman, P. Bogen, and M. Yudelson. Enhancing Digital Libraries with Social Navigation: The Case of Ensemble. In *Research and Advanced Technology for Digital Libraries*, volume 6273 of *Lecture Notes in Computer Science*, pages 116–123. Springer Berlin / Heidelberg, 2010.
13. C. Callison-Burch. Fast, cheap, and creative: Evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 286–295. Association for Computational Linguistics, 2009.
14. M. Chau, X. Fang, and O. R. L. Sheng. Analysis of the Query Logs of a Web Site Search Engine. *Journal of the American Society for Information Science and Technology (JASIST)*, 56(13):1363–1376, November 2005.
15. S. Dignum, U. Kruschwitz, M. Fasli, Y. Kim, D. Song, U. Cervino, and A. De Roeck. Incorporating Seasonality into Search Suggestions Derived from Intranet Query Logs. In *Proceed-*

ings of the IEEE/WIC/ACM International Conferences on Web Intelligence (WI'10), pages 425–430, Toronto, 2010.
16. B. M. Fonseca, P. B. Golgher, E. S. de Moura, B. Pôssas, and N. Ziviani. Discovering search engine related queries using association rules. *Journal of Web Engineering*, 2(4):215–227, 2004.
17. B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani. Using association rules to discover search engines related queries. In *Proceedings of the First Latin American Web Congress*, pages 66–71, 2003.
18. F. C. Gey, M. Buckland, A. Chen, and R. R. Larson. Entry vocabulary – a technology to enhance digital search. In *Proceedings of the First International Conference on Human Language Technology*, 2001.
19. M. R. Ghorab, J. Leveling, D. Zhou, G. J. F. Jones, and V. Wade. Identifying common user behaviour in multilingual search logs. In *CLEF 2009 Workshop, Part I*, volume 6241 of *Lecture Notes in Computer Science (LNCS)*, pages 518–525. Springer, 2010.
20. A. Göker and D. He. Analysing web search logs to determine session boundaries for user-oriented learning. In *AH '00: Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 319–322. Springer, 2000.
21. D. Hawking. Enterprise Search. In R. Baeza-Yates and B. Ribeiro-Neto, editors, *Modern Information Retrieval*, pages 645–686. Addison-Wesley, 2nd edition, 2010.
22. B. J. Jansen, A. Spink, C. Blakely, and S. Koshman. Defining a session on Web search engines. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(6):862–871, April 2007.
23. B. J. Jansen, A. Spink, and S. Koshman. Web Server Interaction with the Dogpile.com Metasearch Engine. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(5):744–755, March 2007.
24. J. Jansen, A. Spink, and I. Taksa, editors. *Handbook of Research on Web Log Analysis*. IGI, 2008.
25. T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting click-through data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161, Salvador, Brazil, 2005.
26. T. Joachims and F. Radlinski. Search engines that learn from implicit feedback. *IEEE Computer*, 40(8):34–40, 2007.
27. R. Jones and K. L. Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management (CIKM'08)*, pages 699–708, 2008.
28. R. Jones, B. Rey, O. Madani, and W. Greiner. Generating Query Substitutions. In *Proceedings of the 15th International World Wide Web Conference (WWW'06)*, pages 387–396, Edinburgh, 2006.
29. D. Kelly, K. Gyllstrom, and E. W. Bailey. A comparison of query and term suggestion features for interactive searching. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 371–378, Boston, 2009.
30. U. Kruschwitz. An Adaptable Search System for Collections of Partially Structured Documents. *IEEE Intelligent Systems*, 18(4):44–52, July/August 2003.
31. U. Kruschwitz. *Intelligent Document Retrieval: Exploiting Markup Structure*, volume 17 of *The Information Retrieval Series*. Springer, 2005.
32. D. Lungley and U. Kruschwitz. Automatically maintained domain knowledge: Initial findings. In *Proceedings of the 31st European Conference on Information Retrieval (ECIR'09)*, pages 739–743, Toulouse, 2009.

33. C. Manning, R. Prabhakar, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008.
34. K. Markey. Twenty-five years of end-user searching, Part 1: Research findings. *Journal of the American Society for Information Science and Technology (JASIST)*, 58(8):1071–1081, June 2007.
35. M. Mat-Hassan and M. Levene. Associating Search and Navigation Behavior Through Log Analysis. *Journal of the American Society for Information Science and Technology (JASIST)*, 56(9):913–934, 2005.
36. N. Nanas and A. Roeck. Autopoiesis, the immune system, and adaptive information filtering. *Natural Computing: an international journal*, 8(2):387–427, 2009.
37. B. Poblete and R. Baeza-Yates. Query-Sets: Using Implicit Feedback and Query Patterns to Organize Web Documents. In *Proceedings of the 17th International World Wide Web Conference (WWW'08)*, pages 41–50, Beijing, 2008.
38. M. Sanderson and B. Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–213, Berkeley, CA, 1999.
39. C. Sherman. Why Enterprise Search will never be Google-y. *Enterprise Search Sourcebook*, pages 12–13, 2008.
40. F. Silvestri. *Mining Query Logs: Turning Search Usage Data into Knowledge*, volume 4 of *Foundations and Trends in Information Retrieval*. Now Publisher, 2010.
41. B. Smyth, P. Briggs, M. Coyle, and M. O'Mahony. Google Shared. A Case-Study in Social Search. In *Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization (UMAP)*, pages 283–294. Springer, 2009.
42. R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng. Cheap and Fast - But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics, 2008.
43. J. F. Sowa. Semantic networks. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 1493–1511. John Wiley & Sons, New York, NY, USA, 1992.
44. A. Spink and B. J. Jansen. *Web Search: Public Searching of the Web*, volume 6 of *The Information Science and Knowledge Management Series*. Kluwer, 2004.
45. J. Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.
46. J. Teevan, E. Adar, R. Jones, and M. A. S. Potts. Information Re-Retrieval: Repeat Queries in Yahoo's Logs. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 151–158, Amsterdam, 2007.
47. P. Wang, M. W. Berry, and Y. Yang. Mining Longitudinal Web Queries: Trends and Patterns. *Journal of the American Society for Information Science and Technology (JASIST)*, 54(8):743–758, June 2003.
48. M. White. *Making Search Work: Implementing Web, Intranet and Enterprise Search*. Facet Publishing, 2007.
49. R. W. White, M. Bilenko, and S. Cucerzan. Studying the Use of Popular Destinations to Enhance Web Search Interaction. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 159–166, Amsterdam, 2007.
50. R. W. White and I. Ruthven. A Study of Interface Support Mechanisms for Interactive Information Retrieval. *Journal of the American Society for Information Science and Technology (JASIST)*, 57(7):933–948, 2006.
51. D. Widdows and B. Dorow. A Graph Model for Unsupervised Lexical Acquisition and Automatic Word-Sense Disambiguation. In *Proceedings of the 19th Conference on Computational Linguistics (COLING)*, pages 1093–1099, Taipei, Taiwan, 2002.