Adaptive User Modeling for Query Expansion

Jean-Yves Delort[1],

[1] Montpellier II University – LIRMM
34392 Montpellier, France
delort@lirmm.fr

Abstract. Web users are often overwhelmed by the number of result pages retrieved by search engines. Query expansion is a common search strategy to narrow the search scope and to increase the relevance of the result lists. Existing query expansion systems proposed by search engines rely on techniques like term co-occurrence or synonymy which do not take into account the user's search behavior. In this article we present an approach that recommends refinement terms with respect to the user's needs and search strategies. The approach is unintrusive and only relies on implicit feedbacks extracted from the navigation trails. This article also presents Conqueries, an implementation of the proposed approach, describes the current version which has been used for almost a year and outlines the future improvements.

1 Introduction

Different kinds of interactive systems may assist users seeking for information with a search engine. The main goal of these systems is to improve the user queries which are supposed to represent their information needs. Word re-weighting, search for similar pages, spelling correction and query expansion are among current query reformulation techniques. Query-expansion consists in narrowing the scope of a search by adding new terms to a user's initial query.

Interactive query expansion tools are accessible from search engine interfaces[1]. However, insofar as they are on the server-side, they have a limited access to the user's interaction trails. Accordingly, they often neglect the user's behavior. For example, they may rely on term co-occurrence in a corpus of documents, on a thesaurus [1] or yet on a semantic net [2]. When they are tested on a wide-scale, recommendations provided by this approach (called global-analysis) are observed as being often ignored by users [3]. A likely reason is that the user's behavior is not taken into account. Indeed, Web user's profiles, intentions and strategies may greatly differ.

Agents supporting searching on search engines using the user's behavior have to cope with two main difficulties: 1) with a mean query size of 2.6 words [4], the representations of the user's need is often imperfect and incomplete and, 2) users are often

[1] See for example Yahoo!, Excite, or Kartoo.

reluctant to send their feedbacks, because of privacy concerns or because it is time-consuming.

In this article we present Conqueries, an adaptive agent which assists users to expand their queries with recommended terms and modifiers. Conqueries unobtrusively learn the user's current interests from the navigation trails in order to suggest personalized lists of keywords: Two users making the same query but having different search strategies will be proposed different terms. Conqueries does not depend on the search engine interface. It has been included on several search engines as is in use for almost a year to assist users[2].

The paper is organized as follows. Section two puts forward the main features of Conqueries. Section three explains how Conqueries deals with the user interactions and describes the recommendation algorithm. Section four describes the search engine templates. They are used by Conqueries to easily adapt to different search engine interfaces. Finally, we survey other query expansion systems, discuss how they differ from Conqueries and conclude with the future improvements.

2 System Overview

We have developed a system called Conqueries that helps users to expand their queries during their searches on search engines. The system suggests personalized lists of keywords depending on the user's behavior. It is based on an unintrusive approach to learn the user's interests from their interaction trails.

Conqueries considers two kinds of user interactions:

1. An access to a *result list*, i.e. a page that contains links towards *result pages* retrieved by a search engine.
2. An access to a result page.

Subsection one presents an example of Web search with a search engine where the user is assisted by Conqueries to expand her queries[3]. Subsection two summarizes the main goals of query expansion systems. Subsection three puts forward the hypotheses underlying our system and relates them to these goals. Subsection four describes how Conqueries cope with the user's shifts of focus and how it takes into account her disinterest for previously recommended terms. Finally, subsection five presents the main features of the user interface of Conqueries.

2.1 Example

Conqueries assists a user who is trying to expand her previous query because the results retrieved by the search engine did not satisfy her. The following scenario emphasizes the important steps in the search process and the time of Conqueries action.

[2] http://www.conqueries.com

[3] We will use the feminine pronoun (she, or) when referring to users of either gender.

Let us denote by "SE" a search engine and by "Anna" a user looking for information about the major cities in Sweden[4]:

1. Anna formulates her initial query which is "Sweden cities" and she submits it to SE.
2. The result list page reports that 5,180,000 documents are relevant for the SE. Anna clicks on the first one in the list.
3. Anna looks at the content of the page which deals with tourism in Sweden. Her need is not satisfied and she gets back to the result list.
4. At that time, Conqueries recommends her the following terms, "cityguide", "Stockholm", "Malmö", "sightseeing" and "tours".
5. Anna chooses to insert "Stockholm" and "Malmö" (two of the major Swedish cities) in her query and she submits it to SE.
6. The result list reports now that about 140.000 are relevant. Among the top-ten results, Anna sees and clicks on a link to a document that looks really relevant to her.
7. Indeed, the document contains the list of the major cities in Sweden as well as their number of inhabitants. Anna's information need is satisfied.

This example shows how a query expansion system, like Conqueries, can support searching on a search engine. The next subsection reviews the different kinds of support query expansion systems should provide to the users.

2.2 Supporting Searching on a Search Engine

Vakkari suggests several criteria that an agent supporting searching should fulfill [5]. In this subsection we summarize them and bring out how they relate to query expansion:

C-1 The user should be helped to expand and differentiate their conceptual model of the topic. For instance, the system could help the user:
  a. To find more appropriate or alternative vocabulary,
  b. to narrow the scope of her search (e.g. appending an " AND " operator before a term),
  c. to cut down on the result list (e.g. appending an " AND NOT " operator before a term).

C-2 The user should be helped to formulate the query with support for choosing appropriate boolean modifiers. For instance, the system could help the user:
  a. To choose appropriate boolean modifier "OR", "AND", "NEAR"...

C-3 Recommended terms should be presented to the user within a conceptual structure resembling her mental model.

In the next section, we introduce the hypotheses underlying Conqueries and relate them to these criteria.

[4] This scenario was tested on Google.com, Feb. 9th 2005.

2.3 Our Hypotheses

Our first hypothesis stems from the following observation: Web experts often expand their queries with terms picked up in the content of previously accessed result pages. Let us take an example to emphasize this idea. Let us suppose that a Web expert needs information about health insurance in France and that her initial query is “insurance in France”. Typically, existing search engines would retrieve millions of relevant pages. However, after browsing a few result pages, her next queries can contain refinement terms such as “health”, “medical” or “coverage”. We assume that often, these terms are picked up in the content of the accessed result pages.

An important feature on the search behavior is that intermediary accessed pages often make the user’s needs evolve. Indeed, as a consequence of her viewing of these pages, features she has in mind of her current needs may be changed, removed or added [6]. Thus again, we assume that the user is likely to pick up terms in the content of the previously accessed result pages.

Our approach is based on the two previous hypotheses. Thus, the proposed terms are intended to support the user in the way described by criterion C-1 in section 2.2.

2.4 Adapting to the user’s current interests

The user confidence in the recommender system would probably decrease quickly if the system suggested irrelevant keywords. It can happen between the times when the user starts a new search and before she accesses a first result page. Then, the recommender system keeps recommending terms related to her previous interest because it is based on the last accessed result pages. We use a shift detection heuristic in order to avoid such a situation. When a shift occurs, Conqueries resets all the information about the user’s needs it has saved so far.

Sometimes a term has been recommended several times in a row but the user has never used it. We consider that the user has no interest in it. Conqueries takes into account the previous recommendations in order not to recommend again terms that have been suggested more often than a given threshold.

2.5 Presenting Recommended Terms to the User

The interface of Conqueries is a toolbar in the browser window. Recommended terms are displayed in *menu-words*. The purpose of a menu-word is to show the user the list of modifiers that can be used together with the word in the query. When a user clicks on an item in a menu-word, the content of the query field in the browser window is updated. An update consists either in appending a boolean modifier followed by the word to the query or in removing the chosen word from the query. Boolean modifiers can be “AND”, “AND NOT”, “NEAR”, etc. They are search engine dependent. Figure 1 shows an example of a menu-word associated with the keyword “directories”.

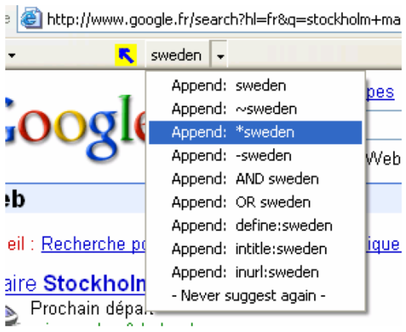

Fig. 1. A query-word

The query field is modified thanks to dynamically inserted Javascript into the content of the page when the page loads. Then, if the user clicks on a menu-item, a Javascript function is called.

The toolbar is made up with three areas [Fig. 2]:

1. "Settings": This single menu opens up a window where the user can tune the recommendation algorithm.
2. "History": This single menu contains the list of the user's previous queries. By clicking on a query, the user submits it again.
3. "Recommended terms": The fourth area displays the menu-words corresponding to the recommended terms.

Note that the interface of Conqueries supports the user in the way described by criterion C-2 in section 2.2.

3 Recommending personalized keywords

In this section we explain how Conqueries deals with the user's interactions and describe its recommendation algorithm.

3.1 Dealing with the user's interactions

Conqueries is a Browser Helper Object (BHO) for Microsoft Internet Explorer. Accordingly, it receives the browser's events triggered by the user's interactions.

When the user wants to access a page, Conqueries recognizes that the URL corresponds to a result list if it contains characteristic substrings like, the search engine domain name, the CGI name, an attribute, etc.

Conqueries finds out that a URL corresponds to a result page if the two following conditions hold:

1. the previously accessed page was a result list.
2. the URL is exactly contained in the enumeration of the previous result list. Indeed, the result list can contain other links (e.g. banners) which do not point towards result pages.

The second condition is checked as follow: First, two markers have two be looked for in the HTML content of the result list. Markers are strings saying precisely where the result enumeration starts and finishes. The URL is compared with all the URLs contained between the markers. If one matches, the URL corresponds to a result page. If the user accesses a result list, Conqueries carries out the following actions:

1. the user query *q* is extracted (directly from the URL in the case of a GET or in the sent data in the case of a POST),
2. *q* is sent to the recommender system,
3. the new list of recommended terms is received from the recommender system,
4. the list is displayed in menu-words in Conqueries.

In the case of result page, Conqueries proceeds as follow:

1. it waits for the content of the result page to be completely displayed in the browser window,
2. the content of the page is sent to the recommender system

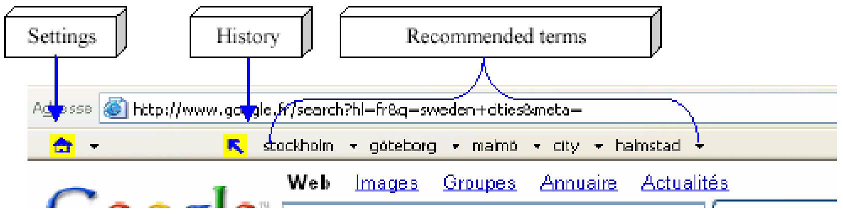

Fig. 2. Conqueries toolbar

3.2 Recommendation Algorithm

User's current interests are stored in three containers of elements represented as term vector: DOCS, QUERIES and RECS. DOCS contains the N last accessed result pages, QUERIES contains all the previous query and DOCS gathers the previous recommendations. Let us denote by CURRENT_REC the current list of recommended terms.

The recommendation algorithm is called when the user accesses a result list or a result page. It is made up with two modules, M1 and M2. M1 builds a list of terms to recommend. M2 removes peculiar terms in CURRENT_REC.

When Conqueries calls the recommender system with the content of the last accessed page as a parameter, the document is pushed on RECS (if RECS size is N, then the oldest document is removed) and M1 and M2 are called successively.
When the user accesses a result list, Conqueries extracts the query (from the URL or from the data sent) and sends it to the recommender system which pushes it on QUERIES and calls M2.

Building the list of terms (M1). Before being added to DOCS, the HTML content of a result page is filtered as follow:

1. Tags and Javascript are removed.
2. The language of the page is predicted by counting the numbers of French and English most frequent terms and choosing the language with the highest number of frequent terms (currently only French and English documents are supported).
3. Given the language of the document, a stemmer is used to remove terms the stems of which belong to a list of frequent stems to stop.
4. Remaining words are counted and are represent in the vector space model [7].

Using the first hypothesis described in section 2.3, the terms to be recommended come from DOCS. Given a term *t* in a document of DOCS, the following formula is computed:

$$w(t) = N(t) \times S(t) \qquad (1)$$

where *N(t)* is the number of documents in DOCS that contain the term *t* and *S(t)* is the sum of occurrences of the term *t* in all the elements of DOCS.
This formula models the second hypothesis presented in section 2.3. It takes into account both the overall frequency of a term in the last accessed results and the number of them containing it.

We preferred (1) to the formula proposed in [6]: $w(t) = \frac{S(t)}{|DOCS|}$. Indeed, this function does not take into account the word frequency in the content of the documents. Accordingly, too many terms have the same weights.

After that, CURRENT_REC is update with the top-K most relevant terms and CURRENT_REC is also pushed on RECS.

Removing peculiar terms (M2). The module M2 is in charge of detecting the user's shifts of focus and of filtering CURRENT_REC. Conqueries heuristic to detect the user's shifts of focus is the following one: as long as two consecutive queries have at least one word in common, the user is considered to be still searching for the same information. Indeed, it is only seldom that user replace all the terms of the previous query. Usually, at least one term representing the main idea remains in the query throughout the search process. When a shift is detected, QUERIES, DOCS and RECS are cleared as explained in section 2.4.

M2 pops CURRENT_REC and removes the terms that have been used in previous queries (QUERIES) or that have been recommended (RECS) more frequently than a given threshold F. The updated list of recommendations is pushed on RECS.

The recommendations worked out by our algorithm are presented to the user when she is on a page where she can reformulate her query. The next section shows how to make Conqueries supporting searching on different search engines.

4 Search Engine Templates

Conqueries can support searching on a great number of search engines provided that it knows useful information about the search engine:

1. the markers used to check if a URL belong to the result enumeration in a result list (let us denote them by, *startMarker* and *endMarker*)
2. the substrings used to check if a URL is a result list (*resultListCondition*).
3. the boolean modifiers and wildcards understood by the search engine (*modifier* in *modifierList*)
4. The form name (*formName*) and the name of the field where the query is entered (*formAttribute*).

These pieces of information are specific to each search engine. They are stored in a search engine template. Figure 3 shows an example of a template for Google.com.

```
<sengine>
<name>Google [COM]</name>
< resultListCondition >
http://www.google.com/search? NOT_q=cache NOT_q=related
</resultListCondition>
<queryAttributeName>q</queryAttributeName>
<startMarker>class=g></startMarker>
<endMarker>clear=all></endMarker>
<modifierList>
<modifier> </ modifier >
< modifier > -</ modifier >
</modifierList>
<formName>gs</ formName >
<formAttribute>q</ formAttribute >
</sengine>
```

Fig. 3. The Google.com template

Every time the user opens a browser instance, Conqueries connects to a server where it downloads the latest templates. If a search engine interface is changed then one only needs to update the template on the server-side so that every user of Conqueries can get the latest template.

5 Related Work

There is a great deal of research on query expansion and agent supporting searching on a search engine. This section summarizes several related approaches and describes how they differ from our approach.

To our knowledge, [8] describes the closest system to Conqueries. It describes a meta-search system that recommends query expansion terms from the content of annotated relevant documents. Howevers the system differs from Conqueries in the following reasons: First, the query expansion system is requires the user explicit feedback. Second, it requires a specific architecture. Third, the user's shifts of focus and the behavior with respect to previous recommendations are not taken into account.

Many other query reformulation systems are based on the user explicit feedback. In [9] a system is described where the user marks an area of text relevant with respect to her current needs. Then, the system uses the surrounding content (called the context) to automatically generate queries. Terms are represented in a semantic space and clustered. Queries are built using the most important terms of each cluster. In [10] a collaborative-filtering technique is used to discover semantically similar queries with a clustering algorithm. Similarity between two queries is computed using their representations as bags of words. The representation of a query contains all the terms included in the result pages annotated relevant by others who have made the same query.

In [11], implicit feedback models are applied to query expansion. For these models, several representations of the same document are available (title, abstract, summary, etc.) and a specific information retrieval system must be used. The user's interest in a document depends on the path she follows between its different representations. Each model has its own method of working out the current interests according to the followed paths. However, the model are not designed to cope with the user's shifts and the behavior with respect to previous recommendations is not taken into account

[12] describes a link recommender system that automatically generates synthesized queries from terms contained in the previously accessed documents. The system is based on an algorithm that predicts whether a term is likely to occur in the last accessed relevant document or not. The approach is based on the idea that the last accessed document is the only relevant one. Unlike Conqueries, this approach does not take into account the facts that user's needs can evolve during the search process and that they can require the access to several relevant pages to be satisfied.

Conclusion and Future Works

In this article we have presented Conqueries, an agent that helps users to refine their queries with suggested query expansion terms. Unlike many other systems, Conqueries does not depend on the user explicit feedback. Conqueries unobtrusively learns the user's interests from her search behavior.

The proposed query expansion method looks for interesting terms in the content of the previously accessed result pages. Basically, the more frequent a term occurs in the content of these pages, the more likely it is supposed to be related to the user's current interests and is worth recommending. In order to provide more accurate recommendations, Conqueries adapts to the user's behavior: First, it detects the user's interests and second, it discards the previous recommendations that have not been used (thus, the previous outputs are one of the input of the recommendation algorithm).

There are a couple of interesting issues that are addressed by Conqueries, for instance, the heuristics 1) to detect the user's shifts of focus and 2) to filter the recommendations using the previous recommendations. The usability of the current interface of Conqueries could also be compared with other kinds of query expansion tools.

The recommendation algorithm presented in this article relies on two hypotheses that need to be evaluated: First, the content of the previously accessed result pages contains useful refinement terms. Second, the more often a term occurs in a result page, the more likely it is to be connected to the user's current interests. In [13], this latter hypothesis was successfully used to detect the user's shifts of focus.

References

1. Voorhees, E.: Query expansion using lexical-semantic relations. Proceedings of the ACM Conference on Research and Development in Information Retrieval (1994) 61—69
2. Stenmark, D.: Query expansion using an intranet-based semantic net. Proceedings of IRIS-26 (2003)
3. Anick, Peter: Using terminological feedback for web search refinement - a log-based study. Proceedings of the ACM Conference on Research and Development in Information Retrieval (2003) 88—95
4. Spink, A., Jansen, B. J., Wolfram, D., Saracevic, T.: From E-Sex to E-Commerce: Web Search Changes. IEEE Computer, Vol. 35 (3). (2002) 107—109
5. Vakkari, P.: Cognition and changes of search terms and tactics during task performance. Proceedings of the RIAO'2000 Conference (2000) 894—907
6. Bates, Marcia J.: The design of browsing and berrypicking techniques for the online search interface. Online Review, Vol. 13(5). (1989) 407—431
7. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM, Vol. 18. (1975) 613—620
8. Smeaton, A. F., Crimmins, F.: Relevance feedback and query expansion for searching the web: A model for searching a digital library. Proceedings of The European Conference on Digital Librairies (1997)
9. Finkelstein, L., Gabrilovich, E., Matias, Y., Rivlin, E., Solan, Z., Wolfman, G., Ruppin, E.: Placing Search in Context: The Concept Revisited. Proceedings of the 10th International WWW Conference (2001)
10. Hust, A., Klink, S., Junker, M., Dengel, A.: Query Expansion for Web Information Retrieval. Proceedings of the 32nd Annual Conference of the German Informatics Society, Web Information Retrieval Workshop (2002)

11. White, R.W., Jose, J.M., van Rijsbergen, C.J., Ruthven, I.: A simulated study of implicit feedback models, Proceedings of the International European Conference on Information Retrieval (2004)
12. Zhu, T., Greiner, R., Haubl, G.: Learning a Model of a Web User's Interests. Proceedings of the Ninth International Conference on User Modeling (2003)
13. Delort, J-Y: Seeking for Clues About Users' Information Needs in Their Navigation. Proceedings of IADIS International Conference WWW/Internet (2004)