CONQUERIES: An Agent that Supports Query Expansion

Jean-Yves Delort[1]

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

Abstract. This article presents Conqueries, an agent that assists users during their searches on search engines. The system recommends terms and modifiers to users so that they can reformulate their queries. The system unobtrusively learns the user's current needs in order to propose personalized lists of keywords. The article presents the design and the architecture of Conqueries which has been implemented and describes the current version which has been used for almost a year.

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 users' behaviors. For example, they may rely on term co-occurrence in a corpus of documents, on a thesaurus or yet on a semantic net [1]. When they are tested on a wide-scale, recommendations provided by this approach (called global-analysis) are observed as being often ignored by users [2]. A likely reason is that the users' behaviors are not taken into account. Indeed, Web users' profiles, intentions and strategies may greatly differ.

Agents supporting searching on search engines using the users' behaviors have to cope with two main difficulties: 1) with a mean query size of 2.6 words [3], the representations of the users' needs are often imperfect and incomplete and, 2) users are often reluctant to send their feedbacks, because of privacy concerns or because it is time-consuming.

This article tackles with the issue of designing an agent which assists web searching through interactive query expansion. It presents Conqueries, an agent which helps

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

users to reformulate their queries with recommended terms and modifiers. The query expansion algorithm is based on the content of the documents accessed by the user, a shift of focus detection algorithm and a heuristic to take into account the user's behavior with the suggestions themselves. The article reports feedbacks and design experience of Conqueries which has been used for almost a year.

The paper is organized as follows. Section two outlines the general architecture of Conqueries. Section three explains the main features of the current algorithm query expansion of Conqueries. Section four describes how Conqueries can easily adapt to different search engine interfaces thanks to search engine templates. 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[2]. Subsection two describes how Conqueries deals with the users' interactions. Subsection three addresses the issue of presenting a list of suggested terms to a user so that she can easily reformulate her previous query.

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. Let us denote by "SE" a search engine and by "Anna" a user looking for information about the major cities in Sweden[3]:

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.

[2] We will use the feminine pronoun (she, or) when referring to users of either gender.
[3] This scenario was tested on Google.com, Feb. 9th 2005.

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 Dealing with the User’s Interactions

Conqueries is a Browser Helper Object (BHO) for Microsoft Internet Explorer. Accordingly, it can receive 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.

2.3 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 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 depend on the query language of the search engine. Figure 1 shows an example of a menu-word associated with the keyword "sweden" during a search on Google.

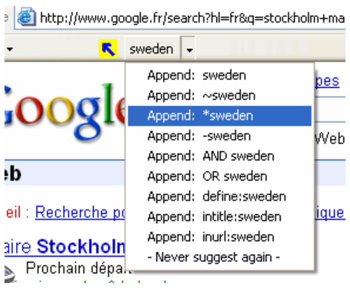

Fig. 1. A query-word

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.

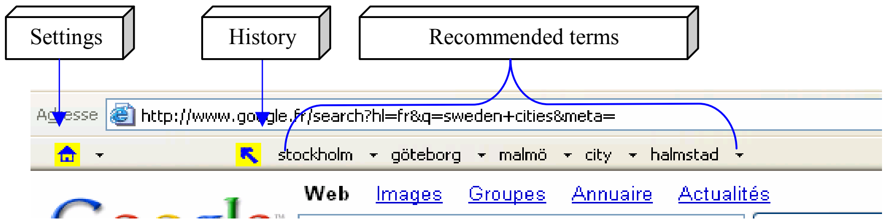

Fig. 2. Conqueries toolbar

3 Query Expansion

In this section we briefly review the main features of the current algorithm used by Conqueries. Details of the algorithm can be found in [4]. The algorithm is based on the assumption that Web searchers often expand their queries with terms picked up in the content of previously accessed result pages [5]. Let us take an example to emphasize this idea. Let us suppose that a user 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 "security". We assume that often, these terms are picked up in the content of the accessed result pages.

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.

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.

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 and Conclusion

To our knowledge, [6] describes the closest system to Conqueries. It describes a meta-search system that recommends query expansion terms from the content of annotated relevant documents. However the system differs from Conqueries in the following reasons: First, the query expansion system 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. [7] 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 the query expansion algorithm used by Conqueries, this approach does not take into account the facts that user's needs can evolve during the search process and that the user may need to access several relevant pages before her information need is satisfied.

This article introduced Conqueries, an adaptive agent that assists users to reformulate their queries on search engines. Conqueries proposes a novel concept of interface for interactive query expansion. The interface has the ability to make the user reformulate her queries in only two clicks. Most users dislike systems that require explicit feedback that is why our query expansion algorithm is unintrusive. It has been improved thanks to the users' feedbacks which tend to emphasize the need of taking into account the shifts of focus and of filtering the unused recommendations.

The effectiveness of the algorithm implemented in Conqueries was studied in [5]. Future works will address the evaluation of the effectiveness of the interface we propose and the improvement of our current query expansion approach using a hybrid approach.

References

1. Stenmark, D.: Query expansion using an intranet-based semantic net. Proceedings of IRIS-26 (2003)
2. 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
3. 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
4. Delort, J.-Y.: Adaptive User Modeling for Query Expansion. Proceedings of the International Conference of Internet Technologies (2005)
5. Delort, J.-Y.: A User-Centered Approach for Evaluating Query Expansion Methods. (Under submission*)*
6. 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)
7. 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)