Disambiguating Search by Leveraging a Social Context Based on the Stream of User's Activity

Tomáš Kramár, Michal Barla, and Mária Bieliková

Faculty of Informatics and Information Technology
Slovak University of Technology
Bratislava, Slovakia
kramar.tomas@gmail.com, {barla,bielik}@fiit.stuba.sk

Abstract. Older studies have proved that when searching information on the Web, users tend to write short queries, unconsciously trying to minimize the cognitive load. However, as these short queries are very ambiguous, search engines tend to find the most popular meaning – someone who does not know anything about cascading stylesheets might search for a music band called `css` and be very surprised about the results. In this paper we propose a method which can infer additional keywords for a search query by leveraging a social network context and a method to build this network from the stream of user's activity on the Web.

1 Introduction

Finding a relevant document based on few keywords is often difficult. Many keywords are ambiguous, their meaning varies from context to context and from person to person. Some words are ambiguous by nature, e.g., a *coach* might be a bus or a person, other words became ambiguous only after being adopted for a particular purpose, not to mention English nouns, which, apart from their natural meaning, also name a software, music band or any other entity. There are also words whose meaning depends on the person who is using them; clearly, architecture means different things to a processor designer than to an architect. Based on the previous observations, we might conclude that using short queries is not a good idea. Unfortunately, this is how we search [1].

The search engines work like databases: they crawl and index documents and respond to queries with a list of results. The order of documents depends on the adopted relevance function; the most widely used search engine today – Google – uses a PageRank relevance function: the more links to a document, the more likely it is to appear at the top positions. This ordering is however not always compatible with user's information needs: a programmer searching for *cucumber* probably does not want to make a salad.

We tackle the problem by implicitly inferring the context and modifying the user's query to include it. The original query is enriched with additional keywords which capture the user's focus. In case of the said programmer, the resulting query might be *cucumber testing* which provides much more valuable and relevant documents than the original query. We select additional keywords

P. De Bra, A. Kobsa, and D. Chin (Eds.): UMAP 2010, LNCS 6075, pp. 387–392, 2010.

following the social network or rather the virtual community the user belongs to in this network. The search thus becomes personalized – the same query for another user from another community might be *cucumber salad*.

The paper is structured as follows. In next section we talk about related work, Sect. 3 gives an overview of how the social network is built, how the communities are extracted and how they are used in the process of keyword inference. In Sect. 4 we talk about preliminary experiments and give conclusions.

2 Related Works

The concept of search disambiguation is certainly not novel. Haveliwala proposed an alternative method of document ranking – a topic-sensitive PageRank [2]. For each document, multiple rank values are calculated, each biased in the context of one root topic from Open Directory Project, ODP[1]. The search results are then biased towards the current topic determined from the words of the document which the user started the search from.

The disambiguation and personalization is often achieved by leveraging some kind of social connections. In [3], the authors proposed a method for personalising the search results by leveraging communities. First a network of users' sessions is constructed (offline) from available access logs. Then, this network is used as a basis for detection of user communities. Subsequently, for each available document an interest of each community is calculated. When a user starts searching, her session is matched to the communities using a cosine similarity and the matching documents are ranked using a Bayesian network computed from the degree of interest of the matched communities to the document. Bender et al. [4] exploit existing social networks for ranking documents with a model called UserRank. The documents tagged, bookmarked or rated by user's friends get higher ratings. This approach, however, does not solve the problem of actually getting the document into the list of results. It can only reorder this list once it was retrieved by other means.

Personalization of the document retrieval itself can be done by automatic query refinement (also called query expansion), which has been recognized long ago as an effective technique to improve search results. Many approaches exist, ranging from an analysis of the lexical affinities [5] to thesaurus based techniques [6]. These methods are however based on a static information which does not always accurately capture user's interest. We believe that query refinement could achieve deeper level of personalization and disambiguation by also analyzing the documents and behavior of similar users as was already shown in [7].

Our method extends and combines the social networks and query refinement methods. We link the users in a social network not only by analyzing URLs of the visited pages, but also by analyzing the content features of these pages. We later use these features to capture user's current interest when she is searching, and also to provide the basis for our query refinement methods.

[1] ODP, `http://dmoz.org`

3 Social-Context Driven Query Expansion

In order to be able to expand the user's query with additional keywords, we need to capture its context and find the most appropriate keywords representing this context. The process is driven by an underlying automatically constructed social network and communities found within this network. The network is constructed from the simple user model based on the stream of user's activity. It is created from content features extracted from pages the user visited and an implicitly acquired user rating of the page.

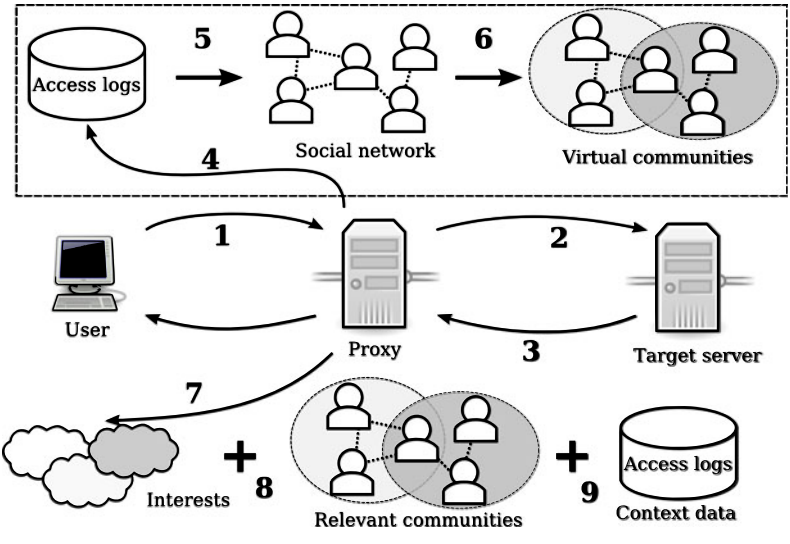

Fig. 1. Overview of the query expansion process

The overview of the process is depicted in Fig. 1. The user requests a page via proxy (step 1) configured in her browser. Proxy requests the page from the target server (step 2) and extracts the *characteristic document features* (step 4) – a vector of

- document keywords (using various keyword extraction algorithms and services such as tagthe.net or OpenCalais),
- tags from *delicious.com* (if available) and
- ODP category.

To capture user's *implicit rating* of the served webpage, a JavaScript code is inserted into every page, which detects user's scrolling and mouse movements and periodically updates the server's record about time the user has spent on the page. The implicit rating is subsequently calculated as a ratio between the time spent on page and the page size: $rating = 1 - \frac{1}{1+X}$, where $X = \frac{time\_on\_page}{document\_size}$.

To improve the accuracy of rating calculation and keyword extraction we extract and use the *cleartext*[2] version of the page, which holds only the core content, stripped of the markup and the navigational components.

Based on user's activity and the extracted features a *social network* is built (step 5), where a weight of an edge denotes a similarity of two users connected by this edge. The network is built in a sequence of steps:

1. New network is created, all users are connected with an edge of weight 0.
2. All documents which have implicit rating lower than the predefined minimal value are discarded.
3. For all visited domains (as in the Internet DNS system), a weight between the users who visited the same domain is incremented by a parameter `d`.
4. For all visited documents, a weight between the users who visited the same document is incremented by a parameter `p` (where $\texttt{d} < \texttt{p}$).
5. For each pair of users, a weight between them is incremented by the size of the overlap of the features extracted from the documents they visited.

The resulting graph represents the users and relationships among them. The stronger is their relationship, the more similar interests they have and the higher is the weight of the edge connecting them.

Next, a *community detection algorithm* is run (step 6), to partition the network into clusters of similar users (based on the stream of their activity). The algorithm is designed to take advantage of the weighted relations in the graph and produces overlapping communities, i.e., a user may belong to multiple communities at one time. This is an important property as a user might have multiple interests, each represented by one community. The community is created in the following steps:

1. Select a random vertex, not yet assigned to any community.
2. Spread the activation energy from the selected vertex to the rest of the network considering weights of the edges.
3. Create new community by collecting all vertices activated via the spreading.
4. If there is an unassigned vertex continue with step 1, otherwise end.

These two stages, the social network creation and community detection are performed periodically and offline. Found virtual communities then provide context for the search – the content features extracted from documents visited by the community or the actions carried on by its members.

In order to identify the *search context*, we capture user's current interest (step 7), which in our case is a set of documents features the user is currently interested in. We construct it dynamically – for every requested document, we search for an overlap of document's content features with the content features from the user's current interest. If an overlap is found, current interest is enriched by document's features. Otherwise, we consider that a new session (and thus a switch of interests) just got started. When we detect that a search has been

[2] We used custom implementation of the publicly available *readability* service, `http://lab.arc90.com/experiments/readability/`

initiated, the current interest helps us to determine all relevant (i.e., sharing at least one feature) communities (step 8). The top `n` communities are then considered as the *search context* and passed to the final stage of query expansion.

We use two approaches to infer new keywords, each using the data provided by the members of the communities (step 9) – *query stream analysis* and *keyword co-occurrence analysis*.

Query stream analysis follows a simple observation of how we do our searches. When a search query does not return relevant documents, it is redefined. The redefinition continues unless the user finds the information or gives up. A query stream represents one searching session, an uninterrupted succession of queries issued by the same user which have some common parts. A sample query stream is: `jaguar`, `jaguar speed`, `jaguar car speed`. We take all queries issued by users from the *search context* and search for query streams where at least one query matches the user's query. Query streams which did not lead to a successful retrieval (the documents visited from a search results page have low implicit rating) are discarded. The last query is extracted from each successful query stream and used to enrich the original query.

A *keyword co-occurrence analysis* is based on analyzing which additional keywords frequently occur with the words from the query in the documents viewed by the users from the current *search context*. The original query is enriched with the top `n` co-occurring keywords.

4 Preliminary Experiments and Conclusion

We evaluate the method of query refinement within a platform of an enhanced proxy server [8]. The proxy plays a crucial role in the experiment setup as it allows us to log each user request and further process and modify the response from a webserver. This way we modify the page before it is displayed and include the search results provided by the expanded query. We also use the modification features to insert scripts into pages to monitor user's activity. We do not need to cope with the user identification as this is all handled by the proxy itself.

The goal of the preliminary experiments was to verify that, given the context and the access logs, the proposed query expansion methods would give satisfactory results, generating queries which achieve subjectively better search results.

To evaluate the query stream analysis, we used AOL search engine data[3]. This dataset contains roughly 20M of queries from 650k users. Table 1 summarizes some queries and how they would be reformulated using this approach.

For the evaluation of the keyword co-occurrence, we used the data collected by the proxy server during its development, that is four users and 310 visited documents. Results provided by this method are also summarized in table 1.

The results of preliminary experiments are promising, as both approaches are capable of redefining the queries to rule out the ambiguity and to provide (subjectively) more relevant search results.

[3] AOL search engine logs, `http://www.gregsadetsky.com/aol-data/`

Table 1. Some examples of query reformulation using different approaches

Method used			
Query stream analysis		**Keyword co-occurrence**	
Original query	**Expanded query**	**Original query**	**Expanded query**
java history	history of java indonesia	passenger	passenger apache
jaguar	jaguar animal	branch	branch git
sphinx	sphinx cats	apache	apache server

The key parts of the method are based on the keywords and on the keyword overlap. We work on improvement of the keyword extraction process by extracting the parent keywords (hypernyms) as proposed in [9]. That should improve the chance for a match between two related documents. For example, documents with keywords `ruby` and `python` do not match, but when extended with their parent category `programming` they generate a match on `programming`.

The proposed method combines social networks and query expansion approaches based on the characteristic content features of the documents. User's current interest is mapped to the interests of the communities and the semantics of the query is inferred from the behaviour of these communities.

Acknowledgments. This work was partially supported by the Scientic Grant Agency of Slovak Republic, grant No. VG1/0508/09 and it is the partial result of the Research & Development Operational Programme for the project Support of Center of Excellence for Smart Technologies, Systems and Services II, ITMS 26240120029, co-funded by the ERDF.

References

1. Jansen, J., Spink, A., Saracevic, T.: Real Life, Real Users, and Real Needs: a Study and Analysis of User Queries on the Web. Information Processing & Management 36(2), 207–227 (2000)
2. Haveliwala, T.H.: Topic-sensitive Pagerank. In: WWW 2002, pp. 517–526. ACM, New York (2002)
3. Almeida, R.B., Almeida, V.A.F.: A Community-aware Search Engine. In: WWW 2004 (2004)
4. Bender, M., et al.: Exploiting Social Relations for Query Expansion and Result Ranking. In: ICDEW 2008, pp. 501–506. IEEE, Los Alamitos (2008)
5. Carmel, D., et al.: Automatic Query Refinement using Lexical Affinities with Maximal Information Gain. In: SIGIR 2002, pp. 283–290. ACM, New York (2002)
6. Liu, S., et al.: An Effective Approach to Document Retrieval via Utilizing WordNet and Recognizing Phrases. In: SIGIR 2004, pp. 266–272. ACM, New York (2004)
7. Kajaba, M., Návrat, P., Chudá, D.: A simple personalization layer improving relevancy of web search. Computing and Information Systems Journal 13, 29–35 (2009)
8. Barla, M., Bieliková, M.: "Wild" Web Personalization: Adaptive Proxy Server. In: Workshop on Intelligent and Knowledge Oriented Tech., WIKT 2009, pp. 48–51 (2009)
9. Barla, M., Bieliková, M.: On Deriving Tagsonomies: Keyword Relations Coming from Crowd. In: Nguyen, N.T., Kowalczyk, R., Chen, S.-M. (eds.) ICCCI 2009. LNCS (LNAI), vol. 5796, pp. 309–320. Springer, Heidelberg (2009)