Folksonomy-Based Adaptive Query Expansion

Claudio Biancalana, Fabio Gasparetti, Alessandro Micarelli, Alfonso Miola, and Giuseppe Sansonetti
Department of Computer Science and Automation,
Artificial Intelligence Laboratory, Roma Tre University,
Via della Vasca Navale, 79, 00146 Rome, Italy
claudio.biancalana,gaspare,micarel,miola,gsansone@dia.uniroma3.it

ABSTRACT

Adaptive query expansion (QE) allows users to better define their search domain by supplementing the original query with additional terms related to their preferences and information needs. The system we present is an extension of the traditional QE techniques, which rely on the computation of two-dimensional co-occurrence matrices. Our system makes use of three-dimensional co-occurrence matrices, where the added dimension is represented by semantic classes (i.e., categories comprising all the terms that share a semantic property) related to the folksonomy extracted from social bookmarking services such as *delicious*, *Digg*, and *StumbleUpon*. The generation of the user profile occurs through the creation of a model that is dynamically updated using the information gleaned from the searches (visited pages and corresponding search queries). The system analyzes the input queries and, if they actually reflect the interests already shown by the user in previous searches, it returns different QEs involving different semantic fields. The output of the system is structured in different blocks categorized through keywords, thus helping the user judge which result is most relevant to him. The results of an experimental evaluation involving real users are reported.

Author Keywords

Social bookmarking, Personalized search, Query expansion

ACM Classification Keywords

H.3.3 Information Search and Retrieval: Query formulation

INTRODUCTION

The amount of information published on the World Wide Web is growing at an astonishing rate, thus making it necessary to devise effective methods for helping users find what they are looking for. Query expansion (QE) allows users to expand their search domain by supplementing their original query with additional terms and phrases [1]. The system we present is a social extension of the traditional QE techniques, which are based on the computation of two-dimensional co-occurrence matrices [3]. Our system makes use of three-dimensional co-occurrence matrices, where the added dimension is represented by semantic classes (i.e., categories comprising all the terms that share a semantic property) related to the folksonomy extracted from social bookmarking services such as *delicious* [1], *Digg* [2], and *StumbleUpon* [3]. The whole procedure of adaptation is completely transparent to the user, as it takes place in an implicit way based on his profile. The user profile is created and dynamically updated using the information related to visited pages and corresponding search queries. The system analyzes the input queries and, if they actually reflect the interests already shown by the user in previous searches, it returns different QEs involving different semantic fields. The output of the system is structured in different blocks categorized through keywords, thus helping the user decide which result is most relevant to him.

SYSTEM ARCHITECTURE

The architecture of the system we propose is depicted in Figure 1.

The roles of modules and the modalities which they actively collaborate through, can be described as follows:

- **Interface** the system interface is the contact point with the user. It has the main role of readdressing external requests to the specialized modules and processing the results obtained in order to show them in a more understandable form;
- **Expansion** after the user has submitted his search query, this module is responsible of the QE process. To perform multiple expansions, this module has to access the user interests stored in the user model;
- **Search** it deals with the actual search, receiving (possibly expanded) queries in input and returning the corresponding results;
- **Persistence** it retains all the necessary information: login data, encountered terms (both before and after stemming), tags, co-occurrence values between terms, tag relevance, and URLs of documents visited by the user; it interacts

[1] delicious.com
[2] digg.com
[3] www.stumbleupon.com

Workshop SRS'12, July 17, 2012, Montreal, Canada.

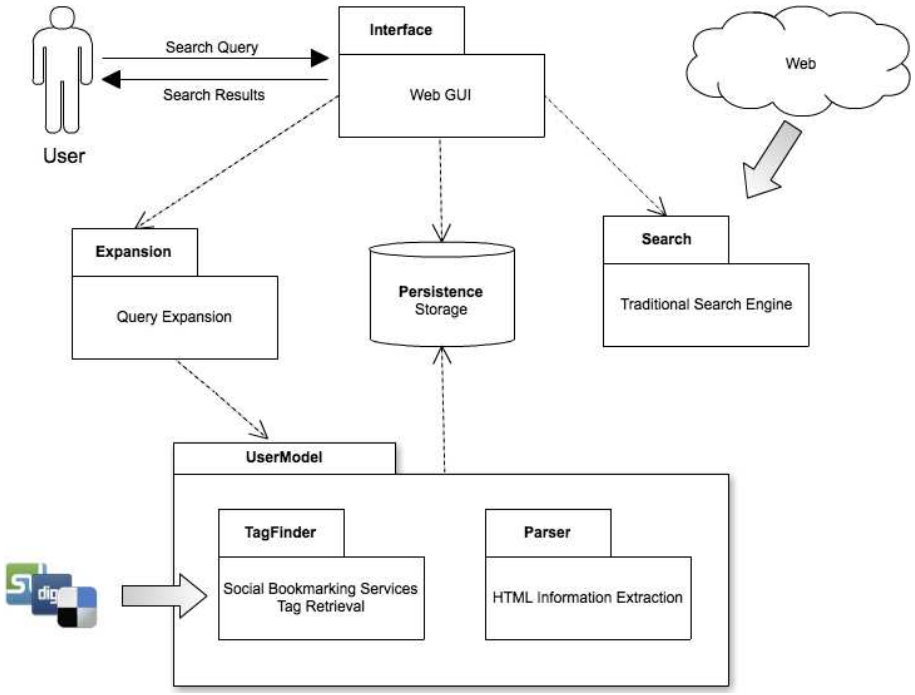

Figure 1. The system architecture

mainly with the interface (for user login and saving URLs) and the user model (for data needed for the construction and consultation of the user model);

- **UserModel** it is the largest module because it deals with updating the user profile realized as a three-dimensional co-occurrence matrix. The interaction with the persistence module is the first step in order to obtain data (visited URLs and corresponding queries) from which to extrapolate information for the model update. Before carrying out the necessary calculations, this module makes use of two other sub-modules: Parser and TagFinder;
  - **Parser** the main role of this sub-module is to filter out the unnecessary information concerning the user interests collected by the system, and to provide the user model with a sorted set of terms for computing the three-dimensional matrix. It includes parsing functionalities (i.e., the format filtering in the HTML pages visited by the user), stemming, and stopword removal;
  - **TagFinder** it is the module dedicated to the search of tags to be associated with the pages visited by the user. It interacts with external resources (social bookmarking services) to find complete tags of a relevance index, in order to provide them to the user model.

Results obtained in each search session are then shown to the user in such a way to underline the different categories of each group of results. The search of the tags associated with the pages visited by the user is carried out by analyzing the information provided by main sites of social bookmarking. In this case, data collection occurs directly by parsing the HTML pages containing the necessary information. In order to model the user visits, the system employes matrices based on co-occurrence at the page level: terms highly co-occurring with the issued keywords have been proven to increase precision when appended to the query [10]. The generic term t_x is in relation with all other n terms t_i (with $i = 1, \ldots, n$) according to a coefficient c_{xi} representing the co-occurrence measure between the two terms. In a classical way, we can construct the co-occurrence matrix using the Hyperspace Analogue to Language approach [5]: once a term is given, its co-occurrence is calculated with n terms to its right (or its left); in particular, given a term t and considered the window f_t of n terms w_i to its right $f_t = \{w_1, \ldots, w_n\}$, we have *co-oc*$(t, w_i) = \frac{w_i}{i}$, $i = 1 \ldots, n$. A pair (a, b) is equal to pair (b, a), that is, the co-occurrence matrix is symmetrical. For each one of the training documents a co-occurrence matrix is generated, whose lines are then normalized to the maximum value. The matrices of the single document are then summed up, so generating one single co-occurrence matrix representing the entire corpus. The limit of this structure consists in the latent ambiguity of collected information: in presence of polysemy of the terms adopted by the user, the result of the query expansion risks to misunderstand the interests, thus leading to erroneous results. In order to overcome this problem, in our system the classical model of co-occurrence matrix has been extended. The user model consists of a three-dimensional co-occurrence matrix (see an example in Fig. 2). Each term of the matrix is linked to an intermediate level containing the relative belonging classes, each accompanied by a relevance index. This way, each term is *contextualized* before being linked to all the other terms present in the matrix, and led to well determined semantic categories that are identi-

fied by tags. In the example illustrated in Figure 2, the term *amazon*, if referred to the semantic class *nature*, shows high values of co-occurrence with the term *river*. Vice-versa, if it is referred to the category *shopping*, it is in strong relation with terms such as *books* and *buy*.

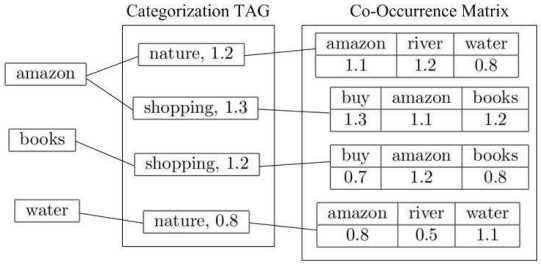

Figure 2. An example of three-dimensional correlation matrix

NEREAU

The system is based on two main algorithms: the first refers to the user model creation and update, the second to the query expansion. With reference to the pseudocode, we notice that the co-occurrence matrix is represented by a map of maps for encoding knowledge and connecting this encoded knowledge to relevant information resources. Maps of maps are organized around topics, which represent subjects of discourse; associations, which express relationships between the subjects; and occurrences, which connect the subjects to pertinent information resources.

User Model Creation and Update

The creation and update of the user model are based on the pages chosen by the user while searching. Starting with an empty model, every time the user clicks on a result after typing a search query, the system records the visited URL, together with the query originally used for the search. Our system performs the analysis of the visited URLs in incremental way, according to the following algorithm (see Algorithm 1, where capital deltas (Δ) denote comments):

- a temporary map M is initialized, where it is possible to record the extracted data, before updating the pre-existent model (empty at first execution). The map keys are the encountered tags, the values are the relative two-dimensional co-occurrence matrices;
- for each visited URL , the corresponding HTML page is obtained, from which the textual information is extracted through a parser, as a list of terms;
- the list of terms is filtered in order to eliminate the stopwords, namely, terms that are very frequent but irrelevant to the creation of the user model;
- the list of terms undergoes a stemming by means of the Porter's algorithm [14]. Meanwhile the system records the relations between stemmed terms and original terms;
- the co-occurrence matrix corresponding to the most relevant k_{term} keywords is evaluated. The relevance is measured by counting the occurrences within the document itself, with the exception of terms used in the query (retained by the system along with the corresponding URL), to which is assigned the maximum weight;
- tags concerning the visited URLs are obtained by accessing different sites of social bookmarking. Each extracted tag has a weight which depends on its relevance (i.e., the number of users which agree to associate that tag to the visited URL);
- the update of the temporary map M is performed by exploiting all information derived from the co-occurrence matrix and the extracted tags in a combined fashion. For each tag_i the system updates the co-occurrence values just calculated, according to the tag relevance weight. After that, the vectors M_{tag_i,t_i}, relative to each term t_i are updated by inserting the new (or summing to the previous) values;
- the set $terms$ is calculated, which contains all terms encountered during the update of the temporary map M;
- from the persistence module a subset UM_{terms} of the user model is obtained as a three-dimensional matrix of co-occurrences, corresponding only to the terms contained in $terms$;
- the matrix UM_{terms} is updated with the values of M. For each t_i belonging to $terms$, the set of keys ($tags$) is extracted from M, which points to values corresponding to t_i. For each tag_i belonging to $tags$, the vector M_{tag_i,t_i} is added to the pre-existent vector UM_{t_i,tag_i}, updating the

Algorithm 1: User Model Creation and Update

```
begin
    Δ Initialize the global co-occurrence matrix M (map of maps);
    M ← Map([]);
    Δ Analyze training documents;
    for (doc, query) in D do
        Δ Parse the document (stemming and stopword removal);
        doc = parse(doc);
        Δ Initialize the co-occurrence matrix of different terms;
        terms ← Map([]);
        Δ Compute the co-occurrence value of every term;
        terms = frequency_occurrences(doc);
        Δ Initialize the co-occurrence matrix of document;
        co_occ ← Map([]);
        Δ Compute the co-occurrence matrix of document;
        co_occ = co_occurrences(terms);
        Δ Get the site list of Social Bookmarking for tag search;
        sites = get_social_bookmarking_sites();
        Δ Initialize URL list tags;
        tags ← Set([]);
        Δ Retrieve tags by URL;
        for i = 0; i < sites.size() & tags.size() = 0; i++ do
            tags = retrieve_tags(url, sites[i]);
        Δ Update the matrix M;
        update(M, tags, terms);
    Δ Initialize all terms in documents;
    all_terms ← Set([]);
    Δ Get unique terms set;
    all_terms = get_term_set(M);
    Δ Get subset of user model;
    user_matrix ← get_user_matrix(all_terms);
    Δ Update user model by the intermediate matrix;
    update(user_matrix, M, all_terms);
    Δ Store updated user model;
    save(user_matrix);
```

Table 1. Example of multiple expansions

original query	*categorization tags*	*expansions*
amazon	**e-commerce, shopping:**	buy **AND** (books **OR** book) **AND** amazon
amazon	**nature:**	(rivers **OR** river) **AND** amazon

values for the terms already present and inserting new values for the terms never encountered.

Query Expansion

Query expansion is performed beginning from the original terms entered into the search engine by accessing the information collected in the user model. The result is a set of expanded queries, each of them associated with one or more tags. This way, it is possible to present the user with different subgroups of results grouped in categories. Using low level boolean logic, every expansion assumes the following form:

$$(t_{11} \text{ OR} \ldots \text{OR } t_{1x}) \text{ AND } (t_{21} \text{ OR} \ldots \text{OR } t_{2x}) \ldots \text{AND } (t_{y1} \text{ OR} \ldots \text{OR } t_{yx})$$

where t_{yx} represents the generic term x corresponding to the stemmed root y. The different terms coming from the same root undergo OR operation amongst them, since the result has to contain at least one of them (see examples in Table 1). The algorithm of multiple expansion is the following (see Algorithm 2):

- let us suppose that the query Q is given, which consists of n terms q_i (with $i = 1, \ldots, n$). For each of them the system evaluates the corresponding stemmed term $q_i^{'}$, so obtaining the new query $Q^{'}$ as a new result;
- for each term belonging to $Q^{'}$, the corresponding two-dimensional vector q_i is extracted from the three-dimensional co-occurrence matrix. Each of those vectors may be viewed as a map, whose keys are the tags associated with the terms $q_i^{'}$ (which have a relevance factor), and the values are themselves *co-occurrence vectors* between $q_i^{'}$ and all the other encountered terms;
- for each encountered tag the relevance factor is recalculated, adding up the single values of each occurrence of the same tag in all two-dimensional vectors. This way, the result is a vector T in which tags are sorted according to the new relevance factor;
- amongst all tags contained in T, only the higher k_{tag} are selected and considered for the multiple expansions;
- for each selected tag t_i the vector sum_{t_i} is computed, which represents the sum of the co-occurrence values of the three-dimensional matrix, corresponding to all terms $q_i^{'}$ of the query $Q^{'}$;
- for each vector sum_{t_i}, the most relevant terms k_{qe} (corresponding to higher values) are selected. Combining the extracted terms with those of the query Q, a new query $EQ^{'}$ (made up of stemmed terms) is initialized;
- for each expanded query $EQ^{'}$, the corresponding query EQ is computed by means of the substitution of stemmed terms with all the possible original terms stored into the system, exploiting the boolean logic according to the scheme previously shown;
- the query EQ and the original tag t_i are entered into the map M_{EQ}, whose keys are expanded queries and values are sets of tags. If M_{EQ} already contains an expanded query identical to the input one, the tag t_i is added to the corresponding set of tags.

Algorithm 2: Multiple Query Expansion

```
begin
    Δ Initialize the query to be expanded (a list of n terms);
    query ← [q_1, q_2, ..., q_n];
    Δ Stemming of query terms;
    query ← stemming(query);
    Δ Get the subset of the user model related to the query;
    user_matrix = get_user_matrix(query);
    Δ Initialize the tag map for multiple query expansion;
    expansion_tags ← Map([]);
    Δ Compute tags for multiple expansion;
    expansion_tags = find_expansion_tags(query, user_matrix);
    Δ Initialize the expanded query map related to tags;
    exp_queries ← Map([]);
    Δ Compute expanded queries for every tag;
    for (tag, ranking) in expansion_tags do
        Δ Compute the expanded query by choosing most relevant terms;
        exp_query = select_relevant_terms(query, user_matrix);
        Δ Enter the result in the expanded query map;
        insert_expanded_query(exp_query, tag, ranking, exp_queries);
    return exp_queries;
```

EVALUATION

In this section we present a comparative analysis of our system Nereau, a search engine that performs query expansion based on co-occurrence data, and the *Google* [4] search engine. The well-known co-occurrence approach for query expansion is based on the calculation of co-occurrences between all terms in a given corpus. It is a straightforward approach that limits the computational complexity keeping the idea of associating contexts to the current user needs. It is not clear the improvements in the performance of document retrieval systems that are possible to gain by the query expansion based on term co-occurrence because very frequent terms tend to discriminate poorly between relevant and non-relevant documents [13]. For this reason, the co-occurrence matrix is built on the corpus of documents retrieved during the learning phase, in order to increase the chance to expand the query with keywords that are likely to be effective discriminators. A total of 42 people were recruited to participate in the user evaluation, mostly students of computer science courses. All participants hold a bachelor's degree. There are a majority of males (36) over females (6). All of them are below 30 years old. This choice allowed us to have people deemed familiar with using search engines in their activities. Some of the recruited people (8%) use search engines once a week on average, while the others

[4] www.google.com

use these tools at least once per day. A substantial number of people (70%) are to be considered experts, that is, they are acquainted with basic notions of boolean matching between words and page contents and they know some advanced search techniques (e.g., boolean operators and phrase search). The performance of the system was assessed by evaluating the normalized version of Discounted Cumulative Gain (nDCG) [8, 9]. nDCG is usually truncated at a particular rank level to emphasize the importance of the first retrieved documents. To focus on the top-ranked items, we considered the DCG@n by analyzing the ranking of the top *n* items in the recommended list with $n \in \{1, 5, 10\}$. The measure is defined as follows:

$$nDCG@n = \frac{DCG@n}{IDCG@n} \quad (1)$$

and the Discounted Cumulative Gain (DCG) is defined as

$$DCG@n = rel_1 + \sum_{i=2}^{n} \frac{rel_i}{\log_2 i} \quad (2)$$

where rel_i is the graded relevance of the i-th result (i.e., from 0=*non significant* to 4=*very significant*), and the Ideal DCG ($IDCG$) for a query corresponds to the DCG measure where scores are re-sorted monotonically decreasing, that is, the maximum possible DCG value over that query. nDCG is often used to evaluate search engine algorithms and other techniques whose goal is to rank a subset of items so that highly relevant documents are placed on top of the list, while less important ones are moved lower. Basically, higher values of nDCG mean that the system output gets closer to the ideal ranked output.

Nereau is a personalized search engine, therefore it needs data related to the current and past user interests and needs. For this reason, the evaluation simulates the traditional interaction with search engines. Each user is asked to chose two general domains of interest with the recommendation that the awareness and familiarity with the topic is adequate for analyzing content retrieved on the Web. For each of these topics, the user performs five search sessions, each related to one specific sub-topic of the chosen domain. The prototype monitors the pages the user decides to visit in the top 10 result page. During the evaluation, there was not any time limit to observe. After training, the user is asked to perform and evaluate a search session related to one information need in the selected domains. In particular, the user has 30 results made up of the three lists of 10 results obtained by three engines: *Google* (i.e., no query expansion), *Co-occurrences* (i.e., query expansion through co-occurrences matrices), and *Nereau* (i.e., query expansion through co-occurrences matrices and folksonomy). The final lists are randomized. Google search engine is chosen for its popularity and high effectiveness. Users express a judgment for each result through a five-point Likert-type scale of values. Now we are able to compute the nDCG measures for the three systems. In order to evaluate the reliability of such comparisons, all results were tested for statistical significance using t-test. In each case we obtained a p-value < 0.05. Therefore, the null hypothesis that values are drawn from the same population (i.e., the output of two search engines are virtually equivalent) can be rejected. Table 2 summarizes the evaluation results. In terms of best performance, Nereau gains on the ideal ranking of users. More precisely, both the query expansion techniques obtain higher results compared with Google. The contextual information that is included during query expansion helps reduce ambiguity and makes the retrieval more accurate. Co-occurrences engine performs better if the task is to recommend only one document (i.e., the more relevant), while Nereau outperforms the other approaches if the task is to retrieve 5 or 10 results. If we look at the number of terms used during query expansion, Nereau has 2.96 terms added to the original query on average, while Co-occurrences engine uses 2.57 terms. In other terms, Nereau alters the query with more words than the Co-occurrences engine.

Table 2. Comparison of search engines in terms of nDCG@n measures

	nDCG@1	nDCG@5	nDCG@10
Google	0.13	0.28	0.44
Co-occurrences	0.44	0.51	0.68
Nereau	0.33	0.55	0.71

RELATED WORK

In literature we found at least four approaches to query expansion that, like our system, exploit the potential provided by social annotations [11, 7, 4, 12]. The method described in [11] relies on creating a social network by analyzing the user activity on the Web. This data are related to URLs and content features of the visited documents. Additionally, the authors consider an implicitly captured user rating. Once the social network has been built, the authors employ an algorithm for detecting the virtual communities within it. This way, the network is partitioned in clusters of similar users. The creation of the social network and the detection of the virtual communities are carried out periodically and offline. When the user issues his search query, the system is able to determine the user current interest as a set of features of the documents he is presently interested in. Subsequently, the system verifies if the user current interest can be mapped to the interests of one or more amongst the virtual communities previously found. If it happens, the system exploits the information of those communities in order to infer the semantics of the original query and select the most appropriate keywords to be added to it.

The second approach is proposed in [7]. It builds and maintains a profile for each folksonomy user, and a knowledge base composed of two graphs that the authors call Tag Resource Graph (TRG) and Tag User Graph (TUG). These graphs store the tags used in the folksonomy and the way they label the resources (TRG) or the way they are retained in the user profiles (TUG). When a user issues a query consisting of a set of tags, the approach proposed in [7] identifies further tags, defined "authoritative", which show a high PageRank in TRG and/or TUG. Such tags are proposed to the user, who may select them to refine his query. The selected tags and the ones directly entered by the user are retained in his profile so to enhance it.

The third QE system that relies on social annotations to improve its performance is described in [4]. In order to achieve social and personalized expansions of a query term t with term t_j, the authors propose the use of two entities: the similarity between t and t_j, which expresses the semantic strength between the two terms; and the similarity between t_j and the user profile, which represents how relevant to the user u a tag t_j is likely to be. The user profile proposed by the authors is a weighted vector, where the generic term is the *user term frequency, inverse user frequency (wtf-iuf)*, that expresses how relevant a term is to a user, given a set of users. After evaluating the similarities above, the system performs a merge operation to provide a final ranking value representing the similarity between t and t_j for the user u. To this aim, the authors advance the use of the Weighted Borda Fuse. As specific constraint of their approach, they also put forward a similarity measure expressing the reliability of an entity e (i.e., user, resource, or tag) in a folksonomy based on its popularity captured by computing the SocialPageRank (SPR) [2]. Based on these considerations, the reliability of an entity e is given by $-log(SPR(e))$.

The last QE system identified in literature, which exploits the potential of social annotations to enhance its effectiveness, is presented in [12]. The advanced approach consists of two phases: (1) a term-dependency method to select the candidate expansion terms is executed, (2) the system employs a machine learning technique for term ranking based on their potential impact on retrieval effectiveness. This is accomplished through the ListNet [6] of learning to rank approaches. ListNet is a feature-based learning to rank method that minimizes a listwise loss function based on the probability distribution on permutations. Neural Network and Gradient Descent are then employed as model and algorithm in the learning method. When a query Q is submitted, the system extracts a set of M possible expansion terms from social annotation sample by means of the term-dependency method. Based on their potential impact on retrieval performance, the ranking model re-ranks the M terms, so producing a new term ranking list. Then, the system selects the top ranked terms in the new list and uses them to expand the original query Q.

CONCLUSIONS

In this paper we have proposed a personalized query expansion approach that relies on the definition of semantic classes (i.e., categories comprising all the terms that share a semantic property) related to the folksonomy extracted from social bookmarking services such as *delicious* and *StumbleUpon*. We have presented the results of an experimental evaluation and a comparative analysis, which confirm the correlation with user interests and the effective coherence and utility of their categorization in semantic classes. A further strength of our system is that the whole procedure is completely transparent to the user, as it takes place in an implicit way based on his choices related to the terms of the submitted queries and the corresponding visited documents. The generation of the user profile occurs through the creation of a model that is dynamically updated by using the information from the searches (visited pages and corresponding search queries).

There are several research thrusts that we intend to pursue in the future. First of all, we intend to study ways of integrating natural language processing knowledge and procedures in our approach. Moreover, we want to introduce the temporal component in order to interpret the user information needs as his searches change over time. A further research challenge is to consider alternative ways of tag categorization to be added to tag search through social bookmarking sites, for example, those based on automatic document categorization. Finally, we would like to enhance our system with new capabilities, such as (i) to make tag suggestions, thus encouraging the discovery of potentially related topics, (ii) to fully use social aspects by considering friend networks, and (iii) to take into account contextual factors related to the user environment on mobile platforms (e.g., smartphones and tablets).

REFERENCES

1. J. Bai, D. Song, P. Bruza, J.-Y. Nie, and G. Cao. Query expansion using term relationships in language models for information retrieval. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 688–695, 2005.
2. S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su. Optimizing web search using social annotations. In *Proceedings of the 16th International Conference on World Wide Web*, pages 501–510, New York, NY, USA, 2007. ACM.
3. C. Biancalana, A. Lapolla, and A. Micarelli. Personalized web search using correlation matrix for query expansion. In *WEBIST (Selected Papers)*, pages 186–198, 2008.
4. M. R. Bouadjenek, H. Hacid, M. Bouzeghoub, and J. Daigremont. Personalized social query expansion using social bookmarking systems. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information*, pages 1113–1114, New York, NY, USA, 2011. ACM.
5. C. Burgess and K. Lund. Hyperspace analogue to language (HAL): a general model of semantic representation. In *Proceedings of the Annual Meeting of the Psychonomic Society*, 1995.
6. Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, New York, NY, USA, 2007. ACM.
7. P. De Meo, G. Quattrone, and D. Ursino. A query expansion and user profile enrichment approach to improve the performance of recommender systems operating on a folksonomy. *User Model. User-Adapt. Interact.*, 20(1):41–86, 2010.
8. K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, New York, NY, USA, 2000. ACM Press.

9. K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

10. M.-C. Kim and K.-S. Choi. A comparison of collocation-based similarity measures in query expansion. *Information Processing and Management*, 35:19–30, 1999.

11. T. Kramár, M. Barla, and M. Bielikov. Disambiguating search by leveraging a social context based on the stream of user's activity. In P. De Bra, A. Kobsa, and D. N. Chin, editors, *UMAP*, volume 6075 of *LNCS*, pages 387–392, Heidelberg, 2010. Springer.

12. Y. Lin, H. Lin, S. Jin, and Z. Ye. Social annotation in query expansion: a machine learning approach. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information*, pages 405–414, New York, NY, USA, 2011. ACM.

13. H. J. Peat and P. Willett. The limitations of term co-occurrence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42:378–383, 1991.

14. M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.