Social Semantic Query Expansion

CLAUDIO BIANCALANA, Roma Tre University
FABIO GASPARETTI, Roma Tre University
ALESSANDRO MICARELLI, Roma Tre University
GIUSEPPE SANSONETTI, Roma Tre University

Weak semantic techniques rely on the integration of Semantic Web techniques with social annotations, and aim to embrace the strengths of both of them. In this paper, we propose a novel weak semantic technique for query expansion. Traditional query expansion techniques are based on the computation of two-dimensional co-occurrence matrices. Our approach proposes the use of three-dimensional 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 and StumbleUpon. The results of an in-depth experimental evaluation performed on both artificial datasets and real users show that our approach outperforms traditional techniques, such as relevance feedback and personalized PageRank, so confirming the validity and usefulness of the categorization of the user needs and preferences in semantic classes. We also present the results of a questionnaire aimed to know the users opinion regarding the system. As one drawback of several query expansion techniques is their high computational costs, we also provide a complexity analysis of our system, in order to show its capability to operate in real time.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Query formulation

General Terms: Algorithms, Experimentation, Human factors

ACM Reference Format:

1. INTRODUCTION

The amount of information published on the World Wide Web is growing at an astonishing rate reaching the size of billions of published pages. However, this unbridled information growth has not been accompanied by a corresponding evolution of techniques for managing and exploiting such information [Mikroyannidis 2007]. Most of the content published on the Web is not structured in a way that makes it easy to answer simple queries. One possible solution is, therefore, to assign semantic information to Web resources in order to facilitate their retrieval and sharing. The techniques for doing this can be divided into strong semantic techniques and weak semantic techniques [Torre 2009]. The former rely on the principles of the Semantic Web [Berners-Lee 1998], which provides a common structure to allow data to be shared and reused.
across applications, using ontologies (i.e., formal descriptions of concepts and their relationships), and machine-processable metadata. Following this approach, resources are annotated with concepts extracted from ontologies that describe a certain domain. Nevertheless, due to their complexity, the development of ontologies is rather costly and laborious.

On the other hand, the weak semantic techniques often rely on the principles of the Web 2.0, which leverages the Web as a collaborative and social platform, where users can play an active role in authoring contents and annotating resources through tags that collectively compose the folksonomy [Merholz 2004] of a knowledge domain. The word “folksonomy”, a combination of “folk” and “taxonomy”, was first used by Thomas Vander Wal in a mailing list [Smith 2004; Quintarelli 2005]. It provides metadata created by users rather than professionals or authors [Mathes 2004]. The great success of the Web 2.0 is mainly due to (i) the widespread appeal of blogging and tagging services that allow users to publish and share content, and (ii) the fact that it does not require collaboration and communication techniques so sophisticated as those required by the Semantic Web.

The system we present is a social extension of the traditional QE techniques, which are based on a coarse syntactic analysis to extract co-occurrences to build two-dimensional matrices [Biancalana et al. 2008]. These matrices basically represent the distribution of co-occurring terms in a given collection of documents. For example, if the terms classic and music appear together in one document, they are considered to co-occur once.

The limit of a relatively simple and easily accessible structure such as this one is the latent ambiguity of the collected information. If the terms chosen by the user are polysemous (e.g., window that means either a glass-filled frame, or an opening in a wall), the query expansion risks misinterpreting the interests, thus leading to an erroneous result. Therefore, how is it possible to expand this structure by focusing on the semantic characteristics of the collected terms? In order to justify the solution, this text suggests it is worthwhile introducing the concept of semantic class [Biancalana and Micarelli 2009]. A semantic class is a category comprising all the terms that share a semantic property (e.g, terms woman, girl belong to the same semantic class identified by the key word female). Our system makes use of three-dimensional co-occurrence matrices, where the added dimension is represented by semantic classes related to the folksonomy extracted from social bookmarking services such as delicious ¹ and StumbleUpon ². Therefore each co-occurrence is associated with a specific semantic class.

In our approach, the expansion process takes place by analyzing multiple occurrences divided into categories related to semantic classes, which are analyzed in the folksonomy. The whole procedure of adaptation 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 pages. The generation of the user profile occurs through the creation of a model that is dynamically updated using the information from the searches (visited pages and corresponding search queries). The input queries are analyzed according to collected data, and if the comparison yields a positive result (i.e., if the queries actually reflect the interests already shown by the user in previous searches), then the system returns different QEs, before carrying out the search. All of these QEs are related to the terms of the user query, but each of them involves a different semantic field. The output of the system is a page where results are displayed in different blocks, each one categorized through keywords, thus helping the user decide which result is most relevant to him. This way our system

¹ delicious.com
² stumbleupon.com
provides the contextualization and categorization of the information by analyzing and extracting the semantic domain of the user interests. The paper describes an in-depth experimental evaluation using real users. A comparative analysis of our findings with those obtained by using well-known techniques, such as relevance feedback, shows that our approach is able to achieve better results. This means that our system provides a better correlation with the actual user interests, which confirms the validity and usefulness of their categorization. One drawback of several QE techniques is their high computational costs [Imran and Sharan 2010]. In order to show the capability of our system to operate in real time, an analysis of its computational complexity is also provided.

This paper is organized as follows. Section 2 describes in detail the system architecture, also providing an analysis of the computational complexity. The experimental setup and the results of an in-depth evaluation on real users are presented in Section 3. Section 4 discusses some works related to our approach, and Section 5 reports our concluding remarks and plans for future work.

2. NEREAU

*Nereau*, Master of Spiders, is the name of a divinity worshipped in the Nauru islands, in Micronesia. It is a foremost figure in many myths, some of which give it a specific role, that of endowing the mad with rationality and the mute with speech, thus making them complete human beings. In the first stages of the ideation, the unusual parallelism between spiders (*spidering* is the activity through which search engines index-link the huge amount of documents on the Web, “jumping” from one link to another) and language (semantic classes, containers of terms characterized by common semantic properties, are driving elements for the concepts set forth in this chapter) has offered a peculiar opportunity to launch the system being developed.

The Nereau search engine memorizes and interprets users’ behavior in order to provide tailor-made results that match the user’s interests. To conduct a search, the user interacts with the system by means of an interface: the entire customization procedure is completely transparent to him, since it occurs implicitly, on the basis of the choices he made when submitting the query to the system and on the pages he visited. The delineation of the user profile occurs through the creation of a model regularly updated with the information obtained from the performed searches (i.e., visited pages and corresponding search queries). Following the definition of a sufficiently representative user model, the system is able to proceed with the “tailoring”: the entered queries are analyzed, considering the collected data, and if the comparison yields a positive outcome (namely, if the queries match the interests the user previously manifested in other searches), the system proceeds with several query expansions, each one referring to a different semantic field pertaining to the words the user entered, before conducting the search. The final outcome is a page in which the results are grouped in several blocks (see Fig. 1), each one classified with keywords. This helps the user to sift through results according to his current interests.

2.1. Folksonomies and Personalized Query Expansion

The proposed approach roots its origin in the semantic analysis of the information stored in social bookmarking services such as *delicious* and *StumbleUpon* and user modeling techniques able to improve the ranking according to the preferences and needs of users. The use of information with a social content, that is, data based on the active participation of all involved users, is the subject of recent studies which underline both its positive and negative sides. Data reliability is sometimes spoiled by the
introduction of erroneous or personal information, or by spamming phenomena [Yanbe et al. 2007].

Al-Khalifa and Davis [2007] address several aspects of the folksonomy, namely the classification freely made by the users, without necessarily having to resort to pre-established hierarchies. The authors suggest that tags can be subdivided into three main categories:

**Personal tags.** Users employ them to organize their own resources, and they are not associated with the actual content. For example, terms such as myblog, tostudy, todo;

**Subjective tags.** They contain individual evaluations on bookmarks and go beyond the information content;

**Factual tags.** These are the ones that are closest to the original content, since they include concepts, names, places or other features that are strictly linked to the resource they refer to.

An evaluation conducted on a wide-ranging set of documents from delicious, obtained the results described in Table I. As we can see, a large percentage of the extracted data falls within the factual tags category. Most of annotations turn out to be often consistent with the categories of the associated bookmarks [Al-Khalifa and Davis 2007]. The presence of a vast number of users, who agree in assigning a tag to a resource, has been shown to be a very reliable criterion [Halpin et al. 2007]. These are indubitable incentives for the experimentation of Information Retrieval techniques entirely or partially based on folksonomies.

<table>
<thead>
<tr>
<th>Table I. Extracted Tags Percentages</th>
</tr>
</thead>
<tbody>
<tr>
<td>tag category</td>
</tr>
<tr>
<td>personal tags</td>
</tr>
<tr>
<td>subjective tags</td>
</tr>
<tr>
<td>factual tags</td>
</tr>
</tbody>
</table>

Recently, the literature has offered several examples of interaction with the data retrieved by social bookmarking services [Biancalana 2009; Biancalana et al. 2008]. In particular, we report an example [Bao et al. 2007] that has represented one of the most relevant input for the conception of our system: the Social Similarity Ranking (SSR),
used to estimate the similarities between a search query and a Web page candidate for result; and the Social Page Ranking (SPR), which determines the rank of each page, based exclusively on the amount of input data associated with the page itself.

Before going into details of the proposed approach, we briefly introduce the query expansion approach.

### 2.2. Automatic Query Expansion

Query expansion is the process of altering a user query to improve retrieval performance. In particular, given a query composed of \( n \) terms \( Q = \{q_1, q_2, \ldots, q_l, q_{l+1}, \ldots, q_n\} \) the alteration might both add new terms \( Q' = \{q'_1, q'_2, \ldots, q'_m\} \) chosen from a dictionary \( E \), and remove terms from the original query \( Q'' = \{q_{l+1}, \ldots, q_n\} \) obtaining:

\[
Q_{\text{exp}} = (Q - Q'') \cup Q' = \{q_1, q_2, \ldots, q_l, q'_1, q'_2, \ldots, q'_m\}
\]  

(1)

The purpose of the query expansion is enhancing the expressivity and reducing the ambiguity of the original query, in order to ensure the subsequent retrieval of information that matches the information needs. Considering definition 1, there are two key issues when it comes to devising an optimal query expansion:

1. how to select the best terms for sets \( Q' \) and \( Q'' \);
2. which terms belong to set \( E \), and (in case) how to sort them;

One answer to the first issue is based on distinction between the several expansion modalities [Efthimiadis 1996], organized as follows:

- **Manual expansion.** It is based on the progressive redefinition of the query entered directly by the user, for example by entering new terms, or, if possible, Boolean operators;
- **Interactive expansion.** It is the joint interaction between the system (which shows the user some terms to modify the original query) and the user (whom, after entering the query, assesses the possibility of expanding it, using the system’s suggestions);
- **Automatic expansion.** It is automatically carried out by the system, in such a way for it to be clear to the user. In this case, similarly to what happens in interactive expansion, it is crucial to distinguish between the several available choices for the source of \( E \) terms:
  - source based on the search results, thus depending on the user’s past choices, or
  - source not depending on the results; it may depend on the entire search corpus (for example the documents indexed by the system) or by external resources.

**Manual expansion** is entirely done by the user. The interpolation of powerful instruments, such as Boolean operators, is a rare feature, ignored or not comprised by most tools, which means that the only manual expansion technique available to everyone is the progressive addition of terms, which gradually become more and more specific and exclusive, in order to limit the noise caused by the huge amount of resources available on the Web.

As for interactive expansion, some examples are to be found in the most advanced search engines. It is the case of Google and Yahoo! which suggest corrections based on potential misspellings; the former actually features other interesting techniques, such as the suggestion of terms associated with the query or the automatic correction of errors. **Automatic expansion** is the most ambitious technique proposed, as well as being the most investigated one in the Information Retrieval field. With this method, the user virtually knows nothing about the process the system follows to expand the
original query: the goal is to find the information needs the user didn’t manage to express completely or correctly.

2.3. Co-occurrence Matrices and Query Expansion

In this section we present our approach to computing the co-occurrence matrices and exploiting them in the query expansion process.

Let us suppose that given a large collection of text documents it is possible to estimate statistical properties of terms. Many statistical measures have been developed to the best term relationship levels [Leydesdorff and Vaughan 2006], either analyzing entire documents, lexical affinity relationship (i.e., pairs of closely related words contain exactly one of the initial query terms), etc. The generic term \( t_i \) is related to 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 this paper, we propose a variant of the Hyperspace Analogue to Language (HAL) approach [Burgess and Lund 1995] for the calculation of matrices of co-occurrence between several terms. Such variant is based on the removal of the sliding window used in the HAL approach, following the simple calculation of co-occurrences between all terms on a page. This simplification has been introduced in order to limit the computational complexity of the original approach, without entirely removing the most worthy intuitions (namely, the ones regarding the ratio between concepts and contexts of association).

The co-occurrence matrix is obtained as follows:

1. for each document taken into consideration, the textual information is extracted as a set of terms \( T = \{t_1, t_2, \ldots, t_n \} \) (where \( |T| = n \));
2. for each term \( t_i \in T \), the number of occurrences \( w_i \) inside \( T \) is calculated, thus obtaining vector \( \overrightarrow{occ}_{t_i} = (t_1, w_1), (t_2, w_2), \ldots, (t_n, w_n) \);
3. starting from \( \overrightarrow{occ}_{t_i} \), we obtain the normalized vector \( \overrightarrow{occ}_{t_i}^{\prime} \), whose elements are calculated as follows:
   \[
   (t_i, w_i^{\prime}) = (t_i, w_i \cdot k_{norm}) \quad 1 \leq i \leq n
   \]
   where \( k_{norm} \) is a normalization coefficient, which may be defined beforehand or vary dynamically (for example, we could define \( k_{norm} = 1/w_{max} \), where \( w_{max} \) is the peak occurrence value contained in vector \( \overrightarrow{occ}_{t_i} \)). The values \( \overrightarrow{occ}_{t_i}^{\prime} \) allow us to compose an intermediate term co-occurrence matrix, to be called \( M_{int} \) (dimensions: \( n \times n \)), whose lines are all equal and corresponding to the same vector;
4. starting from \( M_{int} \), the co-occurrence matrix \( M \) is updated:
   \[
   m_{ij} = M_{int}[t_i][t_j] \cdot M_{int}[t_j][t_i] \quad \forall \ t_i, t_j \in T
   \]
   \[
   M[t_i][t_j] = M[t_i][t_j] + m_{ij}
   \]
   \[
   M[t_j][t_i] = M[t_j][t_i] + m_{ij}
   \]

Note that \( M \) contains a large quantity of duplicated data, since the equations 3, 4, and 5 actually convert it into a symmetrical matrix: this becomes an added value of the system when the matrix is used to expand the query.

Starting from the created co-occurrence matrix, the query expansion process is as follows:

1. given the original query \( Q \) as set of terms \( \{q_1, q_2, \ldots, q_n\} \), the stemming of terms \( q_i \) is carried out, so obtaining the new query \( Q' \);
2. for each term \( q_i' \in Q' \), the corresponding vector \( \overrightarrow{cv}_{q_i} \) is extracted from the co-occurrence matrix \( M \); the vector refers to the \( m \) terms found during the training documents analysis;
3. for each term \( q_i' \in Q' \), the corresponding vector \( \overrightarrow{cv}_{Q'} \) is extracted from the co-occurrence matrix \( M \) by summing the weights referring to the terms themselves;
(4) the vector $\vec{cv}_{Q'}$ is sorted according to the contained values, and the terms with the highest values are extracted from matrix $M$;
(5) the extracted terms are added to query $Q'$, from which the expanded query $Q_e$ is obtained by reconvertting the stemmed roots into real terms.

What follows is a quick example to make the adopted procedure more comprehensible. In order to make things easier, let us suppose that the only training document is the following sentence:

*I love black cats and white dogs, but white cats and white dogs do not like me.*

Following the parsing and stemming of the document we obtain a set of terms $d$ similar to the following one:

\[ d = \{ \text{cat, i, dog, not, lik, lov, and, whit, dog, but, do, me, black, whit, whit, cat, and} \} \]

in which, as you can see, only the root of the found terms is kept, while no trace of the terms’ position in the document is kept. Starting from $d$, we obtain the set $d'$

\[ d' = \{ \text{cat, dog, whit, dog, black, whit, whit, cat} \} \]

A part-of-speech tagger allow us to filter out terms that are not nouns, proper nouns, and adjectives. After these preliminary steps it is possible to calculate the vector $\vec{occ}_d$ containing the occurrence values of the term inside the document:

\[ \vec{occ}_d = \langle (\text{cat}, 2), (\text{whit}, 3), (\text{dog}, 2), (\text{black}, 1) \rangle \]  

(6)

Suppose that the number of acceptable keywords is $k = 3$: according to the calculated values, the new vector of occurrences referring to the first $k$ keywords is the following:

\[ \vec{occ}_k = \langle (\text{cat}, 2), (\text{whit}, 3), (\text{dog}, 2) \rangle \]  

(7)

Starting from here, we obtain the normalized vector $\vec{occ}_k'$

\[ \vec{occ}_k' = \langle (\text{cat}, 0.66), (\text{whit}, 1), (\text{dog}, 0.66) \rangle \]  

(8)

using the value $k_{\text{norm}} = 1/w_{\text{max}} = 1/3$ to implement the normalization; the obtained weights are used to create the temporary matrix $M'$ (Table II).

**Table II. Intermediate Co-occurrence Matrix $M'$**

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>whit</th>
<th>dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0.66</td>
<td>1</td>
<td>0.66</td>
</tr>
<tr>
<td>whit</td>
<td>0.66</td>
<td>1</td>
<td>0.66</td>
</tr>
<tr>
<td>dog</td>
<td>0.66</td>
<td>1</td>
<td>0.66</td>
</tr>
</tbody>
</table>

The last step consists in the calculation of the corresponding co-occurrence matrix $M$, shown in Table III, in which each element is included following the modalities described.

**Table III. Co-occurrence Matrix $M$**

<table>
<thead>
<tr>
<th></th>
<th>cat</th>
<th>whit</th>
<th>dog</th>
</tr>
</thead>
<tbody>
<tr>
<td>cat</td>
<td>0.4356</td>
<td>0.66</td>
<td>0.4356</td>
</tr>
<tr>
<td>whit</td>
<td>0.66</td>
<td>1</td>
<td>0.66</td>
</tr>
<tr>
<td>dog</td>
<td>0.4356</td>
<td>0.66</td>
<td>0.4356</td>
</tr>
</tbody>
</table>

Let us see how the query expansion takes place. Suppose the user types in the following search query:

*dogs and cats*
The corresponding representation as a set is 
\[ Q = \{ \text{cats, dogs, and} \} \]
while the corresponding stemmed query \( Q' \) is the following:
\[ Q' = \{ \text{cat, dog, and} \} \]
For each term contained therein, the relevant vectors contained in matrix \( M \) (if present) are extracted. Hence we obtain
\[
\vec{v}_{\text{cat}} = (0.4356, 0.66, 0.4356) \\
\vec{v}_{\text{dog}} = (0.4356, 0.66, 0.4356)
\]
which we can use to calculate the vector corresponding to the entire query \( Q' \):
\[
\vec{v}_{Q'} = (0.8712, 1.32, 0.8712) \tag{9}
\]
What immediately stands out is the fact that the peak value in formula 9 is the one referring to the stemmed term \text{whit}. Therefore, it is added to query \( Q' \), and the following step is the expanded query \( Q_e \), which takes place by inputting the original terms again, instead of the stemmed ones:
\[ Q_e = \{ \text{cats, dogs, and, white} \} \]

2.4. System Conception
In this section we describes the actual implementation of the system and provides an explicatory example.

Our goal is to combine co-occurrence matrices with the use of tags strictly intended for the semantic aspect. In order to clarify the novelties our system introduces, let us consider a simple example. Suppose that a user, interested in both the polysemous meanings of the term \text{amazon}, has made several searches, visiting one of the links shown by the search engine (Table IV). Later on, the user enters the query \text{amazon} on Google, obtaining some results (Table V).

The query the user entered is undoubtedly incomplete and inexpressive, as far as his interests are concerned, and these limits inevitably affect the search engine results. For example, the first result for \text{amazon}, intended as the \text{river} (Amazon river) comes seventh, after several results linked to Amazon. Despite the incompleteness of the example in point, it is obvious that the term’s polysemy, combined with the inexpressive research query, leads to results that sometimes don’t match the user’s original interests.

Now let us imagine using the query expansion mechanism by creating and using a term co-occurrence matrix. As thoroughly explained in the first chapters, this methodology allows us to construct an information base starting from the documents the user

<table>
<thead>
<tr>
<th>query</th>
<th>url</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy books online</td>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
</tr>
<tr>
<td>amazon faq</td>
<td><a href="http://www.amazon.com/gp/help/customer/display.html?nodeId=10197041">www.amazon.com/gp/help/customer/display.html?nodeId=10197041</a></td>
</tr>
<tr>
<td>amazon river</td>
<td>en.wikipedia.org/wiki/Amazon</td>
</tr>
<tr>
<td>visit amazon river</td>
<td><a href="http://www.discoveramazonia.com/peru/cruises.htm">www.discoveramazonia.com/peru/cruises.htm</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>query</th>
<th>url</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.amazon.com">www.amazon.com</a></td>
<td>AWS-home-page-Money-b/105-2795646-2783816?ie=UTF8&amp;node=343581</td>
</tr>
<tr>
<td>en.wikipedia.org/wiki/Amazon</td>
<td>en.wikipedia.org/wiki/Amazon</td>
</tr>
<tr>
<td>en.wikipedia.org/wiki/Amazon</td>
<td>en.wikipedia.org/wiki/Amazon</td>
</tr>
<tr>
<td>en.wikipedia.org/wiki/Amazon</td>
<td>en.wikipedia.org/wiki/Amazon</td>
</tr>
<tr>
<td>en.wikipedia.org/wiki/Amazon</td>
<td>en.wikipedia.org/wiki/Amazon</td>
</tr>
</tbody>
</table>

Table IV. Search Session on Term Amazon
Table V. Search Session on Term Amazon in Google
visited, searching for the most frequent terms within them. Carrying on with the suggested example, suppose that the user model following the analysis of the URLs becomes similar to the one shown in Table VI (deliberately incomplete, to act as an example).

By looking at the table it is possible to deduce the reciprocal distance from the pairs of terms (*river*, *buy*) and (*river*, *books*) owing to the relatively low co-occurrence values; vice-versa, the term *amazon* is closely related to all other terms.

Suppose submitting the same query, *amazon*, to query expansion, using the previous table. The expansion process is likely to lead to a result not very different to the one shown in Table VII.

The result is obviously far from the expected on: the expansion of the query, based exclusively on the co-occurrence values, entailed an overlapping of very distant concepts, since they refer to an objectively ambiguous query, with no real meaning, such as *amazon books river*.

The main issue is always the same: the polysemy of the term used in the query thwarts the results, or actually makes them counterproductive, of the query expansion based on co-occurrence matrices, because such a method doesn’t take into account the semantic value of the analyzed terms. And this is exactly where the system described hereafter shows how useful it can be.

Table VIII shows a schematic representation of the answer obtained with our system, in which the results are broken down into classified groups through tags. The following paragraph gives a detailed description of the redefinition of the co-occurrence matrix used by the system. See Section 2 for a more in-depth analysis of the three key aspects lying behind this result: the search for tags, the construction of the user model and the multiple expansion of the original query.

### 2.4.1. Three-dimensional co-occurrence matrices

The matrices based on the co-occurrence of terms take the schematic form illustrated in Table IX. The generic term $t_x$ is associated with all other $n$ terms $t_i$ (with $i = 1, \ldots, n$) according to one coefficient $c_{xi}$, which measures the co-occurrence between the two terms. Being an $n \times n$ matrix, each coefficient (aside from the ones on the diagonal) repeats itself twice: this way, despite having to memorize a double amount of data, the data referring to a single term are all available on one matrix line, making access easier during the query expansion process.

In order to limit the effects of polysemous terms during query expansion, we introduce the concept of *semantic class*, a category comprising all the terms that share a semantic property. The terms *woman*, *girl* belong to the same semantic class identified

<table>
<thead>
<tr>
<th>expanded query</th>
<th>results</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>amazon books river</em></td>
<td><a href="www.amazon.com/River-Fire-Water-Taitetsu-Unno/dp/0385485115">Link</a></td>
</tr>
</tbody>
</table>
Table VIII. Results with Nereau

<table>
<thead>
<tr>
<th>nature, south-america:</th>
</tr>
</thead>
<tbody>
<tr>
<td>en.wikipedia.org/wiki/Amazon_River_Dolphin</td>
</tr>
<tr>
<td><a href="http://www.amazon.com/Rivers-Edge-Crispin-Glover/dp/B000053VAX">www.amazon.com/Rivers-Edge-Crispin-Glover/dp/B000053VAX</a></td>
</tr>
<tr>
<td>en.wikipedia.org/wiki/Amazon_River</td>
</tr>
<tr>
<td><a href="http://www.worldwildlife.org/wildplaces/amazon/index.cfm">www.worldwildlife.org/wildplaces/amazon/index.cfm</a></td>
</tr>
<tr>
<td><a href="http://www.nationalgeographic.com/wildworld/amazonriver/">www.nationalgeographic.com/wildworld/amazonriver/</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>outdoors:</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.eduweb.com/amazon.html">www.eduweb.com/amazon.html</a></td>
</tr>
<tr>
<td><a href="http://www.pbs.org/journeyintoamazonia">www.pbs.org/journeyintoamazonia</a></td>
</tr>
<tr>
<td><a href="http://www.britannica.com/eb/article-9109565/Amazon-River">www.britannica.com/eb/article-9109565/Amazon-River</a></td>
</tr>
<tr>
<td><a href="http://www.fishinginamazon.com/">www.fishinginamazon.com/</a></td>
</tr>
<tr>
<td><a href="http://www.ecoadventures.com/SABrochure/peruamazoncruise.html">www.ecoadventures.com/SABrochure/peruamazoncruise.html</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>books, ecommerce, shopping:</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.amazon.com/">www.amazon.com/</a></td>
</tr>
<tr>
<td><a href="http://www.amazon.com/books-used-books-textbooks/b/?ie=UTF8&amp;node=283155">www.amazon.com/books-used-books-textbooks/b/?ie=UTF8&amp;node=283155</a></td>
</tr>
<tr>
<td><a href="http://www.amazon.co.uk/">www.amazon.co.uk/</a></td>
</tr>
<tr>
<td><a href="http://www.amazon.ca/">www.amazon.ca/</a></td>
</tr>
<tr>
<td><a href="http://www.amazon.co.uk/books-used-books-textbooks/b/?ie=UTF8&amp;node=266239">www.amazon.co.uk/books-used-books-textbooks/b/?ie=UTF8&amp;node=266239</a></td>
</tr>
<tr>
<td><a href="http://www.amazon.ca/books-used-books-textbooks/b/?ie=UTF8&amp;node=916520">www.amazon.ca/books-used-books-textbooks/b/?ie=UTF8&amp;node=916520</a></td>
</tr>
</tbody>
</table>

Table IX. Scheme of Co-occurrence Matrix

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>...</th>
<th>tn</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>c11</td>
<td>c12</td>
<td>...</td>
<td>c1n</td>
</tr>
<tr>
<td>t2</td>
<td>c21</td>
<td>c22</td>
<td>...</td>
<td>c2n</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>tn</td>
<td>ctn</td>
<td>ctn</td>
<td>...</td>
<td>ctn</td>
</tr>
</tbody>
</table>

Fig. 2. Examples of semantic classes.

by the key word female; the same goes for guitar, sound with the keyword music, and so on (see Fig. 2).

Just as a semantic class hosts several terms, a single term may belong to several classes, depending on its polysemy, namely, the feature of expressing more than one meaning, referring to different contexts (an example is shown in Figure 3).
Let us go back to the example described in the previous paragraph. If we were to graphically represent the semantic properties of the most relevant terms (amazon, river, buy, ...) we would obtain a result similar to the one shown in Figure 4. Some terms clearly belong to one single class (at least as far as the user's interests are concerned); the term amazon is the exception, since it concurrently belongs to different, clashing classes. The semantic information implied in the user's interests is completely ignored in the pertinent term co-occurrence matrix, and that is why the query expansion formed by polysemous terms such as amazon gives results far from the hoped-for ones. In order to solve such problems, our system introduces the tridimensional co-occurrence matrix.

In order to better explain the novelties the system brings about, let us go back once again to the suggested example, giving a general overview of the developed system's learning phase. We shall refer to the only two results shown in Table X.

Let us analyze the two results in sequence. The system, owing to methods linked to social bookmarking (which the following section studies in depth) knows that

<table>
<thead>
<tr>
<th>query</th>
<th>visited url</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy books online</td>
<td><a href="http://www.amazon.com/">www.amazon.com/</a></td>
</tr>
<tr>
<td>amazon river</td>
<td>en.wikipedia.org/wiki/Amazon_River</td>
</tr>
</tbody>
</table>
amazon.com belongs to semantic classes identified by terms, such as shopping and e-commerce; en.wikipedia.org/wiki/Amazon_River on the other hand, is associated with nature. If we consider expanding the concept of semantic classes to entire co-occurrence matrices, rather than considering only the single terms, the partial result may be summarized as is shown in Figure 5 (the values shown are merely illustrative).

Let us imagine organizing the collected information in a new data structure. The priority is always that of being able to quickly consult the values referring to each single word during the query expansion, therefore, the first level to be accessible from the outside must contain, yet again, the set of terms found in the several training documents. This is when this new system breaks off from the old model (Table XI). Instead of aiming directly at the co-occurrence values, each term of the matrix is linked to an intermediate level that contain the pertinent semantic classes, each one having its own relevance index. In this way, each word is somehow contextualized and redirected to specific categories identified by key words or tags, actually before being linked, through co-occurrence values, to all other terms in the matrix. Only by this stage co-occurrences between terms gain a primary interest again: indeed, for each word, the matrix contains a number of co-occurrence vectors equal to the number of tags with which the same word is associated. That is to say, as we can see in Figure 6, that starting from a single term it is possible to achieve several co-occurrence vectors, each one referring to one single semantic class.

The benefits are obvious: the calculation of the co-occurrence values is combined with the classification by means of tags, in such a way to keep the values referring to semantically distant concepts separated. As Figure 6 shows, the term amazon, if referring to the semantic class nature, presents high co-occurrence values with the term river; vice-versa, if referring to the category shopping, it is strongly linked to the terms books and buy.

The query expansion, considering the introduced novelties, is now able to produce results similar to the illustrative one presented at the end of the previous paragraph. The expansion of a basic query such as amazon, within the context of the tridimensional matrix shown in Figure 6, will follow three directions corresponding to the three tags associated with the term. In such a way we achieve a real multiple expansion of the query, such as the one shown in Table XII.
Table XI. Matrix with References to Semantic Classes

<table>
<thead>
<tr>
<th>term + tag</th>
<th>ecommerce</th>
<th>shopping</th>
<th>nature</th>
</tr>
</thead>
<tbody>
<tr>
<td>buy</td>
<td>1.3</td>
<td>1.3</td>
<td>/</td>
</tr>
<tr>
<td>amazon</td>
<td>1.3</td>
<td>1.3</td>
<td>1.2</td>
</tr>
<tr>
<td>books</td>
<td>1.2</td>
<td>1.2</td>
<td>/</td>
</tr>
<tr>
<td>river</td>
<td>/</td>
<td>/</td>
<td>1.2</td>
</tr>
<tr>
<td>water</td>
<td>/</td>
<td>/</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Fig. 6. Detail of tridimensional matrix.

Table XII. Multiple Expansion of Query Amazon

<table>
<thead>
<tr>
<th>tags</th>
<th>expanded query</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecommerce, shopping</td>
<td>buy AND (books OR book) AND amazon (rivers OR river) AND amazon</td>
</tr>
<tr>
<td>nature</td>
<td></td>
</tr>
</tbody>
</table>

2.5. Social Query Expansion

To explore the potential of personalized query expansion by a social approach, we have developed an innovative search engine that can record and interpret the user behavior in order to provide personalized search results, according to his interests. The whole procedure of personalization is completely transparent to the user, because it occurs in an implicit way based on his choices, related to the terms in the submitted queries, and to the corresponding visited pages. The generation of a user profile occurs through the creation of a model, updated dynamically with the information derived from the searches. The input queries are analyzed according to collected data, and if the comparison has a positive outcome (i.e., if the queries reflect the interests that the user has already shown in previous searches), then the system makes different query expansions, each one to a different semantic field, before carrying out the search. The final result is a page in which results are grouped in different blocks, each of them categorized through keywords, in such a way to facilitate the user in the choice of the result that is most coherent with his interests.

The system takes advantages of some resources freely available on the Web. Results obtained by Google in each search session are shown to the user in such a way to underline the different categories of each group of results. The search for the tags associated with the pages visited by the user, is carried out by analyzing the information provided by two of the main sites of social bookmarking, namely, StumbleUpon and delicious. In this case, the data collection occurs directly by parsing the HTML pages containing the necessary information.
In order to model the user visits, the system employs co-occurrence matrices. The creation of such matrices and their use in the query expansion process have been described in Section 2.3.

As aforementioned, the limit of this approach 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, leading to erroneous results. In order to overcome the above problem, in our system the classical model of co-occurrence matrix has been extended. The user model consists of a three-dimensional correlation matrix (see an example in Fig. 6). Each term of the matrix is linked to an intermediate level, containing the relative belonging classes, each accompanied by a relevance index. In 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, identified by tags.

In the example in Figure 6, 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, is in strong relation with terms such a books and buy.

2.5.1. User Profiling and Query Expansion. Nereau is based on two different algorithms: the first refers to the creation and update of the user model (discussed in Section 2.5.2), the second to the query expansion (discussed in Section 2.5.4). 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 represent relationships between the subjects; and occurrences, which connect the subjects to pertinent information resources. By using a map of maps, it is possible to insert the co-occurrence values between the pairs of co-occurring terms in the training documents when such pairs are present.

2.5.2. Creation and Update of the User Model. 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. Nereau performs the analysis of the visited URLs in an incremental way, according to the following algorithm (see Algorithm 1, where capital deltas (Δ) denote comments):

1. a temporary map \( M \) is initialized, where it is possible to record the collected data, before updating the pre-existent model (empty at first execution). The map keys are the tags extracted from bookmarking services, the values are the relative two-dimensional co-occurrence matrices; (line 3)

2. for each visited URL one obtains the corresponding HTML page, from which the textual information is extracted through a parser, in form of a list of terms; (line 7)

3. the list of terms is filtered in order to eliminate stopwords (i.e., all those terms that are very frequent in all documents, so irrelevant to the creation of the user model); (line 7)

4. the list of terms undergoes a stemming by means of the Porter’s algorithm [Porter 1980]. At the same time, the system records the relations between stemmed terms and original terms; (line 7)

5. the co-occurrence matrix corresponding to the most relevant \( k_{term} \) keywords is evaluated. The relevance is measured by counting the occurrences within the
begin
  // co-occurrence global matrix initialization, represented by a map of maps
  \( M \leftarrow \text{Map}([]) \);
  // training documents analysis
  for \((\text{doc}, \text{query})\) in \(D\) do
    // term occurrence map initialization (stemming and stopword removing)
    \( \text{doc} = \text{parse}(\text{doc}) \);
    // term frequency calculation
    \( \text{terms} \leftarrow \text{Map}([]) \);
    // term frequency calculation for every terms in the document
    \( \text{terms} = \text{frequency}\_\text{occurrences}(\text{doc}) \);
    // co-occurrence matrix initialization
    \( \text{co}\_\text{occ} \leftarrow \text{Map}([]) \);
    // co-occurrence document matrix initialization
    \( \text{co}\_\text{occ} = \text{co}\_\text{occurrences}(\text{terms}) \);
    // get sites list of social bookmarking for tag search
    \( \text{sites} = \text{get}\_\text{social}\_\text{bookmarking}\_\text{sites}() \);
    // initialization of URL list tags
    \( \text{tags} \leftarrow \text{Set}([]) \);
    // retrieve tags by URL
    for \(i = 0; i < \text{sites}.\text{size}() \& \& \text{tags}.\text{size}() = 0; i++\) do
      \( \text{tags} = \text{retrieve}\_\text{tags}(\text{url}, \text{sites}[i]) \);
    end
    // update intermediate matrix \( M \)
    \( \text{update}(M, \text{tags}, \text{terms}) \);
  end
  // get unique terms set
  \( \text{all}\_\text{terms} = \text{get}\_\text{term}\_\text{set}(M) \);
  // get the subset of user model
  \( \text{user}\_\text{matrix} \leftarrow \text{get}\_\text{user}\_\text{matrix}(\text{all}\_\text{terms}) \);
  // update user model by the intermediate matrix
  \( \text{update}(\text{user}\_\text{matrix}, M, \text{all}\_\text{terms}) \);
  // saving the updated user model
  \( \text{save}(\text{user}\_\text{matrix}) \);
end

ALGORITHM 1: User Model Creation & Update

document itself, with the exception of terms used in the query (recorded by the system together with the corresponding URL), to which is assigned the maximum weight; (lines 9-15)

(6) 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., on the number of users which agree to associate that tag with the visited URL); (lines 17-23)

(7) 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 \( \text{tag}_i \) the system updates the values of the co-occurrence just calculated, according to the tag relevance weight. After that, the vectors \( M_{\text{tag}_i,t_j} \), relative to each term \( t_j \) are updated by inserting the new (or summing to the previous) values; (line 25)
(8) the set all\textit{terms} is calculated, containing all terms encountered during the update of the temporary map $M$; (line 28)

(9) from the persistence layer one obtains a subset $UM_{\text{terms}}$ of the user model in form of a three-dimensional matrix of co-occurrence, corresponding only to the terms contained in \textit{terms}; (line 30)

(10) the matrix $UM_{\text{terms}}$ is updated with the values of $M$. For each $t_i$ belonging to \textit{terms}, the set of keys (\textit{tags}) is extracted from $M$, which points to values corresponding to $t_i$. For each tag $i$ belonging to \textit{tags}, the vector $M_{\text{tag},t_i}$ is added to the pre-existent vector $UM_{t_i,\text{tag}}$, updating the values for the terms already present, and inserting new values for the terms never encountered. (line 32)

2.5.3. The search for tags. The introduction of tags is the most innovative aspect of the system being examined, hence the method adopted to search for tags associated with the documents visited by the user deserves a more in-depth analysis. The classification of each one of the resources the user has visited occurs through a variable number of attempts. During each one of these attempts, useful information is potentially extracted from the previously mentioned Web sites StumbleUpon and delicious. In order to make this procedure more clear, take the following URL as an example:

roman\textperiodcenteredia.amazon.com/about/index.htm

The first step consists in the creation of the reference URL for the tag research. The procedure varies according to the chosen Web site; with StumbleUpon the result is as follows:

www.stumbleupon.com/url/roman\textperiodcenteredia.amazon.com/about/index.htm

Yet the analysis of the originated URL yielded a negative outcome: there is no useful information on the URL chosen as an example. A similar, but more complex procedure follows, to search for information on delicious, and the outcome is negative once again. As from now, we proceed with the simplification of the original URL. In a first phase, the path of the original URL is gradually reduced, until its root is reached. We thus have two new results to be submitted to the above mentioned Web sites:

roman\textperiodcenteredia.amazon.com/about roman\textperiodcenteredia.amazon.com

In neither case do we obtain relevant information, and therefore we move on to a second phase in which the obtained URL is further simplified, by gradually removing all possible subdomains, until the most generic result possible is achieved, namely, the domain’s name. In the case in point only one simplification is possible:

amazon.com

Finally, with this result it is possible to find the associated tags. For example, by gaining access to the URL

www.stumbleupon.com/url/amazon.com

we find out that amazon.com is associated with tags such as shopping, books, e-commerce (not by chance these are the same ones presented in the example of the previous section). The search for tags goes with the definition of a rank for each found tag, in order to measure its actual relevance and consistency before it is used to update the user model. Ranks are assigned according to the following observations:
— the maximum value is 1. In Web sites such as delicious, in which it is possible to know the relevance of each tag (i.e., the number of users who agree in associating that tag to the classified URL), the most relevant tag will have the highest value, while the others will be assigned lower values, depending on their relevance;
— the gradual simplification of the URLs to be analyzed go with the definition of a relevance coefficient whose value decreases as we get further away from the original URL. In this way we look for a compromise between the need to classify each resource and the risk of finding inaccurate information following the simplifications.

The decision to consult only StumbleUpon and delicious (the most developed and reliable among social bookmarking services) is due to need to avoid burdening this user model-definition phase from a computational point of view. Anyway, in this regard, the system is totally scalable: in order to exploit other sources of information it is sufficient to define the pertinent tag-extraction rules. Tags are differently weighted on the basis of the corresponding URL according to an algorithm whose a simplified version (without taking into account the diverse access to the several sources of information) may be outlined as follows:

(1) the considered URL is preliminarily formatted by removing disturbance elements such as the starting protocol and anchors;
(2) for each visited social bookmarking service and until no relevant information has been found, the address corresponding to the considered URL is generated and analyzed. The initial value of the relevance coefficient is set to 1;
(3) if the result is negative, the previous step is repeated simplifying further the considered URL and modifying the value of the relevance coefficient;
(4) if the result is positive (i.e., if the analysis of one of the considered URLs leads to a positive outcome) the corresponding tags are extracted and ranked based on their occurrences in the visited source of information.

The value of the relevance coefficient is multiplied by 0.8 for each URL simplification.

2.5.4. Query Expansion. Query expansion is performed starting from the original submitted terms, by accessing the information collected in the user model. The result is a set of expanded queries, each of them associated with one semantic class. In such a way, the user can be presented with different subgroups of results divided in categories. Exploiting the possibilities of submitting queries containing boolean logic of low level offered by Google, every expansion assumes the following form:

\[(t_{11} \text{ OR } t_{12} \text{ OR } \ldots \text{ OR } t_{1x}) \text{ AND } (t_{21} \text{ OR } \ldots \text{ OR } t_{2y}) \text{ AND } \ldots \text{ AND } (t_{y1} \text{ OR } \ldots \text{ OR } t_{yx})\]

where \(t_{yx}\) represents the generic term \(x\) corresponding to the stemmed root \(y\), and the different terms coming from the same root undergo OR operation amongst them, because it is necessary that the result contains at least one of them. See examples in Table XIII.

Hereafter, we describe the algorithm of multiple expansion. It consists of the following steps:

(1) let us suppose to have a query \(Q\) with \(n\) terms \(q_i\) (\(i = 1, \ldots, n\)) (see Fig. 7). For each

\[Q \quad q_1, q_2 \ldots q_n\]

Fig. 7. Original query \(Q\) with \(n\) terms \(q_i\).
Table XIII. Example of Multiple Expansions

<table>
<thead>
<tr>
<th>original query</th>
<th>categorization tags</th>
<th>expansions</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon</td>
<td>e-commerce, shopping</td>
<td>buy AND (books OR book)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AND amazon</td>
</tr>
<tr>
<td>amazon</td>
<td>nature:</td>
<td>(rivers OR river) AND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>amazon</td>
</tr>
</tbody>
</table>

of these terms $q_i$, the system evaluates the corresponding stemmed term $q'_i$, so providing a new query $Q'$ (see Fig. 8);

$$Q' = q'_1 q'_2 \ldots q'_n$$

Fig. 8. Corresponding stemmed query $Q'$ with $n$ terms $q'_i$.

(2) for each term $q'_i$ belonging to $Q'$, the corresponding bidimensional vector is extracted from the three-dimensional co-occurrence matrix. Each of the extracted vectors may be thought as a map, whose keys are the tags $tag_j$ (with $j = 1, \ldots, m$) associated with the term $q'_i$ and the values are themselves co-occurrence vectors between $q'_i$ and all the other encountered terms (see Fig. 9). Each tag $tag_j$ has a corresponding relevance value $valtag_j$ computed as follows:

$$valtag_j = \sum_{k=1}^{s} cooc(q'_i, t_k)$$

that is, the relevance value $valtag_j$ associated with the $tag_j$ is given by the sum of the co-occurrence values $cooc()$ between the analyzed stemmed query $q'_i$ and the $s$ terms $t_k$ associated with $tag_j$;

![Bidimensional vector related to the stemmed query term $q'_i$.](image)

(3) the $m$ tags $tag_j$ associated with the term $q'_i$ can belong to different semantic classes $C_1, C_2, \ldots, C_K$, each of them categorized by one or more tags; for each semantic class $C_k$ a tag vector is produced by sorting its related tags according to their relevance value $valtag_j$ (see Fig. 10);
(4) amongst all the tags belonging to the same semantic class $C_k$, only the $K_{tag}$ tags with the higher relevance values $val_{tag_j}$ are selected, on which the multiple expansion is performed; for each semantic class $C_k$, a sum vector is evaluated by summing the co-occurrence values of terms $t_i$ with $q_i'$ related to $K_{tag}$ tags belonging to $C_k$ (see Fig. 11);

(5) for each sum vector computed in the previous step, the most relevant terms $K_{qe}$ (corresponding to higher values of co-occurrence with the query term $q_i'$) are selected. Combining the extracted terms with those of the stemmed query $Q'$, a new query $EQ'$ is produced (see Fig. 12);

$$EQ' = Q' + \{t_1, t_2, ..., t_{Kqe}\}$$

Fig. 12. Expanded stemmed query.

(6) for each expanded query $EQ'$, the corresponding query $EQ$ is calculated through the substitution of stemmed terms with the original terms recorded by the system; obviously, we have a number of expanded queries equal to the number $K$ of semantic class $C_k$ (see Fig. 13). Google employs an ad-hoc word variation or automatic stemming feature. In order to avoid to perform two different stemming processes for the retrieval, the original terms substitute the related stemmed terms used by Nereau during the query expansion obtaining the expanded original queries $EQ$.

$$EQ'_1 \rightarrow EQ_1$$
$$EQ'_2 \rightarrow EQ_2$$
$$...$$
$$EQ'_K \rightarrow EQ_K$$

Fig. 13. Expanded original queries.
(7) the queries $EQ$ obtained above, with the related tags, are entered into the map $M_{EQ}$, in which the keys are expanded queries and the values are sets of tags. If $M_{EQ}$ already contains an expanded query identical to the input one, its tags are added to the corresponding set of tags (see Fig. 14).

![Fig. 14. Map with expanded queries and associated tags.](image)

The following simplified example illustrates the query expansion process.

1. Let us suppose to have the query $Q = \{\text{drafts}\}$ with a single term (i.e., $n = 1$) (see Fig. 15).

![Fig. 15. Original query considered in the example.](image)

After the stemming process, we have the query $Q' = \{\text{draft}\}$ illustrated in Figure 16;

![Fig. 16. Corresponding stemmed query.](image)

2. then the co-occurrence analysis follows, in which a bidimensional vector corresponding to the stemmed query item $q' = \text{draft}$ is extracted from the three-dimensional co-occurrence matrix (see Fig. 17); we obtain the following $K = 3$ semantic classes with the related tags: $C_1 = \{\text{beer, pub, food}\}, C_2 = \{\text{basket, nba}\}, C_3 = \{\text{design, resource}\}$;

3. for each semantic class $C_k$, we have a relevance sorted vector obtained by considering the relevance values $valtag_j$ (see Fig. 18);

4. for each semantic class $C_k$, we select the $K_{tag} = 2$ tags with the higher relevance values $valtag_j$ in order to compute the related sum vector (see Fig. 19);

5. for each sum vector, the most relevant terms $K_{qe}$ ($K_{qe} = 1$ in the example) are combined with the terms of the stemmed query $Q'$ in order to produce the expanded query $EQ'$ (see Fig. 20);
Fig. 17. Bidimensional vector related to the stemmed query term *draft*.

Fig. 18. Relevance sorted vectors related to the $K = 3$ semantic classes $C_k$.

Fig. 19. Computation of the sum vectors related to three semantic classes $C_k$. 
Fig. 20. Expanded stemmed queries related to the three semantic classes $C_k$.

(6) for each expanded query $EQ'_i$, the corresponding query $EQ$ is calculated, through the substitution of stemmed terms with the original terms recorded by the system (see Fig. 21);

(7) the expanded original queries and the related tags are entered into the map $M_{EQ}$.

2.6. Complexity Analysis

The following section presents an in-depth analysis of the computational complexity of the implemented algorithms. The operations regarding access to external resources, such as persistence or the social bookmarking services, conventionally have a unit cost (this approximation, albeit imaginary, has the purpose of evaluating only the complexity linked to the calculation phases of algorithms).

2.6.1. Complexity of the user model update. In order to calculate the computational complexity of the algorithm referring to the update of the co-occurrence tridimensional matrix, we will use the following values:

- $n$: Number of analyzed training documents;
- $c$: Constant that indicates the number of operations to be conducted during the document parsing phase (stemming, stopword removal, term extraction);
- $c_u$: Average complexity index of an URL (namely, the number of simplifications to be made to go back to the root);
- $T$: Average number of terms each document contains;
- $K$: Number of keywords extracted from each document;
- $t_{term}$: Average dimension of a co-occurrence vector referring to a tag and a term in the temporary co-occurrence matrix;
- $n_{term}$: Number of different keywords selected in all training documents;
- $m_{tag}$: Average number of tags found for each document;
- $n_{tag}$: Number of different tags found for all documents;
- $t_{tag}$: Average number of tags referring to a single term in the temporary co-occurrence matrix.

The calculation of complexity entails the following steps:

Parsing. Parsing and its associated operations (stemming and stopword removal) have a cost that consistently increases with the number of terms contained in...
the considered document, hence the complexity of the entire set of documents is $O(n c T)$;

**Calculation of normalized occurrences.** It is subdivided into two phases: the first one for the selection of the $K$ keywords following the calculation of occurrences (cost equal to $O(n T \log(T))$, due to the sorting of the set of $T$ terms to select the keywords, and the second one for the normalization of the values referring to the extracted terms (cost equal to $O(nK)$). The most relevant component is obviously the former, therefore the overall cost of this operation is $O(n T \log(T))$;

**Calculation of co-occurrences.** The two embedded cycles necessary to calculate the co-occurrences entail a quadratic complexity with reference to the extracted $K$ terms $O(nK^2)$;

**Search for tags.** Ignoring the cost of the initial formatting of the studied URL (cost = $O(1)$ for each document), the complexity of the recursive algorithm to search for tags referring to a URL can be schematized as follows:

$$C(x) = \begin{cases} O(1) & \text{se } x = 0 \\ C(x - 1) + O(1) & \text{se } x > 0 \end{cases}$$

where $x$ is the number of attempts to be made before the outcome of the search for tags is positive. In the **worst-case scenario** (assuming no useful results are found despite the subsequent simplifications), the number of attempts is equal to the complexity index of URL $c_u$, hence the complexity of the algorithm implemented in all documents is $O(nc_u)$;

**Update of the intermediate matrix.** The cost of the operation depends on the number of tags associated with the document ($m_{tag}$) and on the square of the extracted keywords ($K$), therefore $O(nK^2m_{tag})$;

**Calculation of the set terms.** This operation costs $O(n_{tag})$ (since it depends on the overall number of tags present in the temporary matrix);

**Update of the user model.** For each considered term there are two main phases: the search for tags associated with the term in the intermediate matrix (cost being $O(n_{term}n_{tag})$) and the actual update of the matrix representing the user model (cost depending on the values $t_{tag}$ and $t_{term}$, thus equal to $O(n_{term}t_{tag}t_{term})$).

After analyzing the main steps of the algorithm it is possible to draw some generic conclusions (shown in Table XIV).

<table>
<thead>
<tr>
<th>Algorithm phase</th>
<th>Dominant values</th>
</tr>
</thead>
<tbody>
<tr>
<td>operations on all training documents</td>
<td>$O(ncT), O(n T \log(T)), O(nK^2m_{tag})$</td>
</tr>
<tr>
<td>update of the pre-existing user model</td>
<td>$O(n_{term}n_{tag}), O(n_{term}t_{tag}t_{term})$</td>
</tr>
</tbody>
</table>

The most relevant components for computing the complexity of the first phase of the algorithm (namely, all the operations cyclically carried out on all documents) have costs equal to $O(ncT)$ (parsing), $O(n T \log(T))$ (calculation of occurrences) and $O(nK^2m_{tag})$ (update of the temporary matrix). As far as the second phase is concerned (focusing on the update of the pre-existing user model), the two most relevant phases have complexity $O(n_{term}n_{tag})$ and $O(n_{term}t_{tag}t_{term})$, respectively. In order to correctly interpret such results, it is necessary to assess the best, worst, and average cases as follows:
Worst case. The terms extracted from each document are different from all the other extracted terms and all the documents are assigned different tags;

\[
\begin{align*}
\text{worst case:} & \quad \begin{cases} 
 n_{\text{term}} &= nK \\
 t_{\text{term}} &= K \\
 n_{\text{tag}} &= nm_{\text{tag}} \\
 t_{\text{tag}} &= m_{\text{tag}}
\end{cases} \\
\end{align*}
\tag{11}
\]

Best case. The same keywords are always extracted from each document, and each document is assigned only one tag;

\[
\begin{align*}
\text{best case:} & \quad \begin{cases} 
 n_{\text{term}} &= K \\
 t_{\text{term}} &= K \\
 n_{\text{tag}} &= 1 \\
 t_{\text{tag}} &= 1
\end{cases} \\
\end{align*}
\tag{12}
\]

Average case. We have the following conditions:

(1) the terms extracted from each document are mostly, but not completely, different from all the other terms found in the other documents;
(2) the terms that co-occur on average with each term are basically the same as the average number of terms extracted from each document;
(3) most tags associated with each document are different from all the other tags assigned to other documents;
(4) the number of tags referring to each term is equal to the average number of tags assigned to each document.

\[
\begin{align*}
\text{average case:} & \quad \begin{cases} 
 n_{\text{term}} &\simeq nK \\
 t_{\text{term}} &\simeq K \\
 n_{\text{tag}} &\simeq nm_{\text{tag}} \\
 t_{\text{tag}} &\simeq m_{\text{tag}}
\end{cases} \\
\end{align*}
\tag{13}
\]

The average case therefore coincides almost entirely with the worst case, unless there are significant variations (irrelevant in terms of asymptotic complexity). The complexities of the second phase of the algorithm, with reference to the three cases studied, assume the forms shown in Table XV. In the user model update we can immediately notice how, in the average case, one of the two cost components is equal to that of the temporary matrix update.

<table>
<thead>
<tr>
<th></th>
<th>Worst case</th>
<th>Best case</th>
<th>Average case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>(O(n^2Km_{\text{tag}}))</td>
<td>(O(K))</td>
<td>(O(n^2Km_{\text{tag}}))</td>
</tr>
<tr>
<td>Phase 2</td>
<td>(O(nK^2m_{\text{tag}}))</td>
<td>(O(K^2))</td>
<td>(O(nK^2m_{\text{tag}}))</td>
</tr>
</tbody>
</table>

Based on the above analysis, the computational complexity of the algorithm in the average case is as follows:

\[
C_1 = O(ncT) + O(nT\log(T)) + O(nK^2m_{\text{tag}}) + O(n^2Km_{\text{tag}}) \tag{14}
\]

2.6.2. Complexity of the multiple query expansion. After analyzing the complexity of the algorithm for the user model update, let us focus now on the multiple query expansion. Even in this case, we use some predefined values:

\(q\). Number of terms of the original query;
\(t_{\text{tag}}\). Average number of tags referring to one single term in the user model;
\(n_{\text{tag}}\). Overall number of tags referring to the terms of the query;
\(k_{\text{tags}}\). Maximum number of tags selected for the multiple query expansion;
Social Semantic Query Expansion

Average dimension of a co-occurrence vector referring to a term and a tag in the user model;

Let us examine the main steps required to calculate the complexity:

**Stemming of the query.** The initial stemming operation of the query has a cost of $O(q)$;

**Search for tags.** For the expansion process, tag research is parametric compared to the number of the query terms and to the average number of tags referring to each term: hence $O(q_{tag})$. The second phase of the algorithm has a cost of $O(n_{tag}\log(n_{tag}))$;

**Creation of the expanded query.** The operation is repeated for a number of times equal to the number of selected tags ($k_{tags}$); the cost of each expansion is $O(q_{term}\log(t_{term}))$ (the most costly operation is sorting the set of terms that may be used for the expansion process), therefore the complexity of all the tags is $O(qk_{tags}t_{term}\log(t_{term}))$.

The most relevant component is the one referring to the creation of expanded queries, so the complexity of the multiple query expansion algorithm is as follows:

$$C_2 = O(qk_{tags}t_{term}\log(t_{term}))$$  \hspace{1cm} (15)

### 2.6.3. Evaluations

On the basis of such analyses, we can draw some conclusions on the developed system, also with reference to the system based on co-occurrence on a page level. Table 2.6.3 shows the computational complexities of the two systems.

<table>
<thead>
<tr>
<th>System</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>co-occurrence page + keywords</td>
<td>$O(ncT) + O(nT\log(T)) + O(nK^2)$</td>
</tr>
<tr>
<td>Nereau</td>
<td>$O(ncT) + O(nT\log(T)) + O(nK^2m_{tag}) + O(n^2Km_{tag})$</td>
</tr>
</tbody>
</table>

Our system presents a complexity that is comparable with that of the system already present in literature. The introduction of the classification by means of semantic classes obviously entails a worsening of the computational time needed for the execution of the algorithm. Compared to the first system, the phase of the actual update of the co-occurrence matrix is burdened by the introduction of the new term $m_{tag}$.

To imagine a real case, let us momentarily leave out the components shared by both the calculated complexities (i.e., $O(ncT) + O(nT\log(T))$). Unless documents with an irrational number of terms are found, the most burdened phases of the two algorithms are indeed the ones focusing on the update of the respective user models with the $K$ terms extracted from each document. The choice of the most relevant between the two remaining terms that contribute to the complexity of Nereau depends on the ratio between the number of analyzed documents ($n$) and the number of keywords extracted from each document ($K$). In the most likely event that the second term is greater than the first one, we can finally assert that the difference between the complexities of the two compared systems is represented only by the term $m_{tag}$, which is usually present in only a few units.

Something must be stated about the analysis of the real performance of Nereau. The phase in which the system reveals its major impact in terms of execution time, is undoubtedly the search for tags to be associated with each document. This is due to the lack of specific libraries for a quick access to data collected from social bookmarking services. As a result, the only (and extremely burdensome) way to retrieve information is accessing the pages available on the Web and then parsing them. This requires
additional time for downloading and parsing Web pages. The problem is mostly evident when URLs are involved, and as well there is no information associated with them (or only after a radical simplification of the original URL).

3. EVALUATION

In this section, we present a comparative performance analysis among Nereau, the proposed social-based search engine, and other query expansion and personalized search approaches.

A number of different aspects must be evaluated in order to assess the real effectiveness of search engines, such as index coverage, search capabilities, presentation, and user effort in seeking tasks. In this evaluation, we are particularly interested in the standard relevance measures to evaluate the effectiveness of the retrieval of Web documents and the quality of the results. Several relevant factors make this comparative analysis somewhat difficult. Personalized search aims at enhancing user interaction by understanding the user needs, the context, and the applications and information being used, typically across a wide set of user goals. Usage data that might be of potential interest for recognizing and assessing information consumption patterns of each user and the various information foraging strategies must be accurately collected. Moreover, personalization is influenced by the selection of particular topics on which the evaluation is to be performed. It can create an authoring bias where the topics selected by a group of peers influence the relative results of one approach when compared with others. For example, one approach might exploit a topic characterized by a wealth of documents and references, while a different one is critically affected by the presence of several polysemous words in the query set. In spite of these issues, implementing an experimental evaluation of personalized approaches in a real setting is still the most significant method to measure the scalability and the overall quality of search effectiveness, in terms of both coverage and accuracy of the produced search results. While coverage measures the ability of engines to produce all the references that are likely to be visited by the user, accuracy is essential in evaluating the quality of such references.

Five different search engines have been included in the comparative analysis: Google (denoted simply as Google in the following figures and tables), the personalized version of Google (PersGoogle), a query expansion search engine based on co-occurrence data (CoOcc), a traditional search engine with Relevance Feedback (RF), and our system (Nereau). In the first personalized version of Google back in 2004, the search engine showed a directory like category drop-down menu, where users could select the categories that matched their interests. During the search process, the search engine adapted the results according to each user needs, assigning a higher score to the resources related to what the user had seen in the past. A slider in the graphic user interface allowed the user to control the level of personalization in the results. For example, if the user had earlier chosen the category of Computers as one of his interests, results such as Apple, Acer or HP would have ranked among the first positions. Unfortunately, no details or evaluations are presently available for the algorithms exploited for the re-ranking process, except the ones contained in the patent application filed in 2004 [Zamir et al. 2004]. Our comparative evaluation takes into account the current version of personalized Google. It basically reorders the search results based on gathered usage data, such as previous queries, Web navigation behavior and, possibly, visited sites that serve Google ads, computers with Google Applications installed, such as Desktop Search and personal information, which may be implicitly or explicitly provided by the user.

Relevance feedback aims at modifying the initial query using words extracted from top-ranked or identified relevant documents. If both documents and queries are rep-
resented in a Vector Space Model [Salton and Buckley 1997], the Rocchio feedback approach alters the initial query by combining the vectors of the relevant documents increasing the recall of the search engine, and possibly its precision as well [Manning et al. 2008].

Query expansion based on co-occurrences is a well-known approach that collects the correlations between pairs of terms in a given corpus. It is a straightforward approach that limits the computational complexity through the idea of associating contexts to the current user needs, as described in Section 2.1. The two fundamental problems of information retrieval, namely, synonymy and polysemy, are addressed during the construction of the query vector. Ambiguous words have only one lemma for all their meanings. If one meaning is mentioned in a query, the documents in which the term appears with the other meanings are also retrieved and estimated as closer to the query. In case of polysemy there will be terms associated with more than one meaning, but if the query is composed by a number of keywords, the intended meaning is more likely to be referenced. These terms and their related ones will form a cluster, which is associated with the intended meaning and outweighs the unintended meanings. Several studies in the literature have proven the effectiveness of this approach, but have also raised some doubts on its real improvements in the performance of document retrieval systems, because of the following potential issues:

— Weighting terms that occur more frequently in the whole dataset, so favoring the more popular (see, for example, Peat and Willett [1991]);
— Expanding each single term in the query in isolation, ignoring the potential meaning of the all terms as a whole;
— Co-occurrences data extracted from small collections of documents;
— Collection of documents not including relevant concepts and information during the query expansion.

In order to play down those issues mainly related to the documents selected for the initial dataset, the co-occurrence matrix used for expansion is built on the corpus of documents retrieved during the learning process. In this way, it is certain that enough relevant documents for the expansion are included and there are less chances to see several common terms that cover several different topics of interests.

The comparative analysis consists in the following three evaluations:

— TREC corpus-based evaluation;
— ODP corpus-based evaluation;
— Web user-based evaluation.

Corpus-based evaluations have the advantage of showing a zero test-retest variability if the same closed corpus is employed in future experiments that include different approaches. Nevertheless, as stated previously, experimental settings to real scenarios provide undoubted insights into the performance of the retrieval engines.

We also include a specific disambiguation analysis in order to measure the efficacy of the search engines to tackle queries characterized by polysemic and ambiguous terms. A brief qualitative analysis summarizes the opinions collected by users that had been employing the Nereau engine for a four-week period of daily usage.

3.1. TREC corpus-based evaluation
In the first evaluation, we consider the TREC 3 2004 Robust Track on TREC disks 4 and 5. It contains over 500K documents, a subset of them marked as “relevant” or “irrelevant” according to a given topic. On average, each document consists of 467 terms.

3trec.nist.gov/data.html
All the 249 queries are included in the evaluations. The approaches considered in this evaluation are RF, CoOcc, Google, and Nereau. We cannot include PersGoogle too because most of the documents of the TREC collection are no longer available in the Google index, so they would not be included in the user history exploited for personalizing the results.

In order to express the performance of the retrieval we employ the Precision at 20 (P@20) and the Mean Average Precision (MAP): the former evaluates the fraction of the retrieved documents that are relevant to the user information needs, the latter is useful to average various precisions when there are sets of distinct queries to be submitted to the search engine. We do not consider the recall measure, which evaluates the proportion of relevant documents that are retrieved, as it is not computable in open corpus domains. In the Web, in fact, we cannot know the whole number of relevant documents available. For the same reason, we do not consider the F1 score (or F-measure) either, an other standard statistical parameter, which combines the precision and the recall of the test to compute the resulting score.

The average number of result pages viewed by a typical user for a query is 2.35 [Jansen et al. 2000], and a more recent study [Jansen et al. 2005] reports that about 85.92% of users view no more than two result pages. For these reasons, the precision is evaluated at a given cut-off rank, considering only the top 20 results returned by the system. Figure 22 and 23 show the P@20 and MAP scatter plots, respectively, after collecting a certain number of feedbacks. Figure 24 shows the box plots of the measures. Google approach shows the worst outcomes with a low average precision and MAP. This is an expected result because Google does not exploit the suggestions that feedbacks might provide. Better average outcomes are obtained by employing the relevance feedback, even though the slope of the linear model of data is negative. That is to say that the amount of information collected by means of the relevance feedback negatively affects the precision by including irrelevant keywords during the expansion of the queries. Better outcomes are obtained through both CoOcc and Nereau approaches.

It should be noted how several Web references included in the corpus do not find a correspondence in the delicious social service. For this reason, Nereau is put in an unfavorable position in comparison with CoOcc trained on the collection of documents related to the relevant topics. The same issue also affects the ODP corpus-based evaluation (see Sect. 3.2). In the performed evaluations the exact total proportion of resources for which Nereau could retrieve tags was 62%. In spite of that, Nereau is still able to obtain a better MAP and upper quartile on the obtained precision values.

3.2. ODP corpus-based evaluation

Our goal is to build profiles of users that show interests in some specific topics. Each topic must be associated with more than one document, whose content is extracted by personalized search engines and used to build a user profile representation.

Open Directory Project 4 (ODP) is a multi-language directory of links belonging to the Web. ODP has a hierarchic structure: the links are grouped into categories and subcategories, also known as topics. It is therefore possible to identify a level-based organization within the hierarchy. An example of topic is Top/Business/Forestry and Agriculture/Fencing; excluding the Top level common to all the topics, we have:

— Level I: Business;
— Level II: Forestry and Agriculture;
— Level III: Fencing.

4www.dmoz.org
Given the large quantity of links contained in ODP, we have decided to limit to the third level the links taken into consideration for the evaluation. The pages corresponding to such links are retrieved from the Web and indexed. The obtained index consists of 131,394 links belonging to 5,888 topics. Thereafter, ten topics are chosen at random, five of which corresponding to potential user information needs, and five whose function is exclusively that of representing the pages visited by the user whose content is not relevant, that is, transient needs. The links of each topic were then subdivided into a training set, corresponding to 25% of the links, and the remaining links for test sets. The ten topics are summarized in Table XVII. The same evaluation methodology has been employed in previous scenarios regarding personalization in information seeking tasks [Gasparetti and Micarelli 2007; Gasparetti et al. 2009].

It is clear now that this methodology allows us to build several different profiles of potential users. Once these profiles are built, it is possible to compare the precision of the search engines. In this evaluation, Google, RF and Nereau approaches are
compared in terms of F1 score. A query is built for each topic belonging to the user needs. The query is composed by the terms that form the topic name in ODP (e.g., query="shopping craft papers"). The evaluation aims at measuring the fraction of document retrieved by the search engine from the whole collection of indexed documents that are also included in the test set for each need. Table XVIII shows the variation of F1 score for the three engines. In this evaluation, RF engine does not take any sensible advantage of the content extracted from the training documents. Nereau outperforms the other approaches, even though several links in the training set do not have any reference in the delicious service. Part of the training documents are indeed very old or not very popular, therefore it is not likely that users attach metadata to these resources on delicious.
Fig. 24. Box plots of MAP and P@20 values.

Table XVII. Benchmark Statistics: ODP Topic, Number of Links for Test and Training, and if Topic is Part of User Needs

<table>
<thead>
<tr>
<th>Topic</th>
<th>Test links</th>
<th>Training links</th>
<th>Need</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sports/Cycling/Human Powered Vehicles</td>
<td>15</td>
<td>5</td>
<td>+</td>
</tr>
<tr>
<td>Computers/Home Automation/Products and Manufacturers</td>
<td>27</td>
<td>7</td>
<td>+</td>
</tr>
<tr>
<td>Business/Mining and Drilling/Consulting</td>
<td>74</td>
<td>18</td>
<td>+</td>
</tr>
<tr>
<td>Games/Roleplaying/Developers and Publishers</td>
<td>52</td>
<td>14</td>
<td>+</td>
</tr>
<tr>
<td>Business/Agriculture and Forestry/Fencing</td>
<td>100</td>
<td>27</td>
<td>+</td>
</tr>
<tr>
<td>Shopping/Crafts/Paper</td>
<td>35</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Arts/Performing Arts/Magic</td>
<td>25</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Science/Publications/Magazines and E-zines</td>
<td>26</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Science/Social Sciences/Linguistics</td>
<td>13</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Recreation/Guns/Reloading</td>
<td>15</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>382</td>
<td>101</td>
<td></td>
</tr>
</tbody>
</table>

Table XVIII. Comparison in terms of F1 Score

<table>
<thead>
<tr>
<th>Topic</th>
<th>PersGoogle</th>
<th>RF</th>
<th>Nereau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers/Home Automation/Products and Manufacturers</td>
<td>0.05</td>
<td>0.08</td>
<td>0.16</td>
</tr>
<tr>
<td>Sports/Cycling/Human Powered Vehicles</td>
<td>0.09</td>
<td>0.13</td>
<td>0.09</td>
</tr>
<tr>
<td>Games/Roleplaying/Developers and Publishers</td>
<td>0.10</td>
<td>0.18</td>
<td>0.18</td>
</tr>
<tr>
<td>Business/Mining and Drilling/Consulting</td>
<td>0.19</td>
<td>0.14</td>
<td>0.19</td>
</tr>
<tr>
<td>Business/Agriculture and Forestry/Fencing</td>
<td>0.05</td>
<td>0.14</td>
<td>0.57</td>
</tr>
<tr>
<td>Average F1</td>
<td>0.10</td>
<td>0.13</td>
<td>0.24</td>
</tr>
</tbody>
</table>

3.3. Web user-based evaluation

We have discussed a system evaluation through a test collection and the results of evaluation metrics to calculate the effectiveness score of the system.

Personalized search engines, such as Nereau, need to collect and analyze large amount of usage data related to the current and past user interests and needs in order to provide better recommendations in comparison with traditional approaches. For this reason, the evaluation also involves a group of people that have evaluated the effectiveness of the search engines in real scenarios. A total of 42 people have been
recruited to participate in the user evaluation, mostly students of Computer Science courses. All participants hold a bachelor’s degree. A vast majority of males (36) outnumbers females (6). All of them are aged below 30. This choice allowed us to have people deemed comfortable with using search engines in their activities. Some of the recruited people (8%) use search engines once a week on average, while the others use these tools at least once a day. A substantial number of people (70%) are to be considered experts, namely, they know the basic notions of boolean matching between words and page contents, and they are familiar with some advanced search techniques (e.g., boolean operators and phrase search).

Each user is asked to choose two general domains of interest with the recommendation that the awareness and familiarity of the topic is adequate for analyzing contents retrieved on the Web. For each of these topics, the user performs five search sessions, each one related to some specific sub-topic of the chosen domain. The prototype monitors the pages the user decides to visit in the top ten results page. There is no time limit to be observed during the evaluation.

After training, the user is asked to perform and evaluate a search session related to one information need in the chosen domains. In particular, the user has 40 results made up of the three lists of ten results obtained by four engines: Google, PersGoogle, CoOcc, and Nereau. The final lists are randomized. Google search engine is chosen for its popularity, high effectiveness, and the state-of-the-art of ranking algorithms in Web information retrieval. Moreover, by asking users to create a personal account, Google is able to provide personalized ranks based on the users Web history. Users with a Google account were asked to clear their Web history or otherwise create a new one. Google evaluation is performed by asking the users to log out from the search engine before retrieving any search result.

Users express a judgment for each result with a five-point Likert-type scale of values. The performance of the recommendation process was assessed by evaluating the normalized version of Discounted Cumulative Gain (nDCG) [Järvelin and Kekäläinen 2000; 2002]. It is a well-known measure for evaluating a graded relevance scale of documents in a search engine result set. Rather than MAP, nDCG is much more focused on the top of the ranked list.

nDCG is usually truncated at a particular rank level to emphasize the importance of the documents retrieved first. 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:

\[
\text{nDCG}@n = \frac{\text{DCG}@n}{\text{IDCG}@n}
\]

and the Discounted Cumulative Gain (DCG) is defined as:

\[
\text{DCG}@n = \text{rel}_1 + \sum_{i=2}^{n} \frac{\text{rel}_i}{\log_2 i}
\]

where \( \text{rel}_i \) is the graded relevance of the \( i \)-th result (i.e., from 0=non significant to 4=very significant), and the Ideal DCG (\( \text{IDCG} \)) 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 order a subset of items in such a way that highly relevant documents are placed on top of the list, while less important ones are moved further down. Basically, higher values of nDCG mean that the system output gets closer to the ideally ranked output.
In order to evaluate the reliability of such comparisons, all results were tested for statistical significance using the 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 outputs of two search engines are virtually equivalent) can be rejected.

Table XIX summarizes the evaluation results. In terms of best performance, Nereau wins on the ideal ranking of users, especially when the user sifts through five or more results. The worst performance is obtained by the non-personalized Google approach. More precisely, both CoOcc and Nereau obtain higher results. The contextual information that is included during the query expansion helps reduce ambiguity and makes the retrieval more accurate. CoOcc query expansion performs slightly 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 five or ten results in absolute terms. Figure 25 better explains the results with same means for nDCG@1, while for nDCG@5 and nDCG@10 Nereau behaves more accurately. The difference between the two approaches is also observable by the number of terms used during the expansion of the query. Nereau adds 2.96 terms to the original query on average, while CoOcc uses 2.57 terms. Basically, Nereau alters the query with more words than the co-occurrence based retrieval.

3.3.1. WordNet-based disambiguation analysis. Personalization has an important role when users submit ambiguous queries, that is, consisting of terms with multiple different meanings. Past and current contexts might help disambiguate polysemous words and improve result accuracy. For this reason, this evaluation aims at gathering ambiguous queries and performs a comparison of how the different approaches behave in correctly disambiguating their meanings.

A straightforward methodology that involves the analysis of the WordNet [Fellbaum 1998] lexical database has been defined. Briefly, WordNet is a collection of synsets, namely, groups of nouns, adjectives and adverbs all expressing a common concept (e.g., house, home, dwelling, habitation, etc.). Synsets are interlinked by means of semantic and lexical relations. In this way, it is very easy to find terms that have potentially several different meanings [Hirst and Budanitsky 2005].

A random choice of these ambiguous terms enables us to focus on the 10 keywords shown in Table XX. For each term, two synsets (or semantic contexts) are identified by the ones included in the database.

Formally, it is possible to define a triple \(< T, X_T, Y_T >\) with the following characteristics:

— \(T\) is a polysemic term, with different meanings depending on its context (for example, \(T = \text{mercury}\));
— \(X_T\) and \(Y_T\) are two sets of tags, each of which consists of five tags that briefly describe a semantic context (e.g., \(X_T = \{ \text{msn, java, chat, im, linux} \}\) and \(Y_T = \{ \text{planets, nasa, solar system, space, astronomy} \}\); of course, \(T\) gets a different meaning in each of the two semantic contexts \(X\) and \(Y\).

Table XX summarizes the set of triples used in this evaluation.

For every triple, we collected 400 documents from the Web, subdivided into:
— 100 documents for each of the two contexts X and Y. These two collections are divided into two parts of 50 documents each one, which we used for training and test; — 200 “noisy” documents, namely, that belong to both of the two semantic classes, so obtaining a collection of 4800 documents.

The documents of the three collections related to the two contexts X and Y and the noisy collection are retrieved by submitting to delicious the following queries, respectively:

— q: T (tag:x1 OR tag:x2 ... tag:x5) -tag:y1 -tag:y2 ... -tag:y5
— q: T -tag:x1 -tag:x2 ... -tag:x5 (tag:y1 OR tag:y2 ... tag:y5)
— q: T -tag:x1 ... -tag:x5 -tag:y1 ... -tag:y5

Each document retrieved by delicious is annotated with a set of tags. As might be expected, two profiles $U_X$ and $U_Y$ are built by analyzing the documents and tags of the context X and Y, respectively. The noisy collection too is included in both the profiles.
At the end of training, the initial terms $T$ are submitted to the search engines. For each term, the following measures are evaluated for both the profiles $U_X$ and $U_Y$: precision, recall, and $F_1$ F-measure. Tables XXI and XXII summarize the results.

Table XXI. Terms and Semantic Contexts

<table>
<thead>
<tr>
<th>Topic</th>
<th>C</th>
<th>RF R</th>
<th>RF P</th>
<th>RF F1</th>
<th>CoOcc R</th>
<th>CoOcc P</th>
<th>CoOcc F1</th>
<th>Nereau R</th>
<th>Nereau P</th>
<th>Nereau F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>amazon</td>
<td>a</td>
<td>0.65</td>
<td>0.60</td>
<td>0.62</td>
<td>0.70</td>
<td>0.62</td>
<td>0.66</td>
<td>0.80</td>
<td>0.62</td>
<td>0.70</td>
</tr>
<tr>
<td>cancer</td>
<td>a</td>
<td>0.40</td>
<td>0.34</td>
<td>0.37</td>
<td>0.50</td>
<td>0.43</td>
<td>0.46</td>
<td>0.50</td>
<td>0.54</td>
<td>0.52</td>
</tr>
<tr>
<td>capital</td>
<td>a</td>
<td>0</td>
<td>0.58</td>
<td>0</td>
<td>0</td>
<td>0.58</td>
<td>0</td>
<td>0.20</td>
<td>0.60</td>
<td>0.30</td>
</tr>
<tr>
<td>depression</td>
<td>a</td>
<td>0</td>
<td>0.20</td>
<td>0</td>
<td>0</td>
<td>0.10</td>
<td>0</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
</tr>
<tr>
<td>harrison</td>
<td>a</td>
<td>0.30</td>
<td>0.55</td>
<td>0.39</td>
<td>0.10</td>
<td>0.68</td>
<td>0.17</td>
<td>0.30</td>
<td>0.70</td>
<td>0.42</td>
</tr>
<tr>
<td>lee</td>
<td>a</td>
<td>0.30</td>
<td>0.65</td>
<td>0.41</td>
<td>0.20</td>
<td>0.64</td>
<td>0.30</td>
<td>0.40</td>
<td>0.66</td>
<td>0.50</td>
</tr>
<tr>
<td>mercury</td>
<td>a</td>
<td>0.40</td>
<td>0.70</td>
<td>0.51</td>
<td>0.50</td>
<td>0.66</td>
<td>0.57</td>
<td>0.50</td>
<td>0.66</td>
<td>0.57</td>
</tr>
<tr>
<td>oxford</td>
<td>a</td>
<td>0.75</td>
<td>0.60</td>
<td>0.67</td>
<td>0.80</td>
<td>0.58</td>
<td>0.67</td>
<td>0.80</td>
<td>0.60</td>
<td>0.69</td>
</tr>
<tr>
<td>porter</td>
<td>a</td>
<td>0.30</td>
<td>0.90</td>
<td>0.45</td>
<td>0.40</td>
<td>0.86</td>
<td>0.55</td>
<td>0.30</td>
<td>0.70</td>
<td>0.42</td>
</tr>
<tr>
<td>victoria</td>
<td>a</td>
<td>0.80</td>
<td>0.70</td>
<td>0.75</td>
<td>0.80</td>
<td>0.58</td>
<td>0.67</td>
<td>0.70</td>
<td>0.80</td>
<td>0.75</td>
</tr>
<tr>
<td>vancouver</td>
<td>a</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.80</td>
<td>0.86</td>
<td>0.83</td>
<td>0.80</td>
<td>0.86</td>
<td>0.83</td>
</tr>
<tr>
<td>york</td>
<td>a</td>
<td>0.70</td>
<td>0.50</td>
<td>0.58</td>
<td>0.80</td>
<td>0.70</td>
<td>0.75</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
</tr>
</tbody>
</table>

While the order of the topics that obtain better results are similar among the considered approaches, the average precision and recall measures differ significantly. Topics such as *mercury*, *lee*, and *amazon* are clearly easier to disambiguate while *cancer*, *capital*, and *depression* need more sophisticated approaches. The average precision favors *Nereau* and the approach based on co-occurrences. In terms of average recall and $F_1$ score *Nereau* outperforms both *RF* and *CoOcc*. In particular, the average of the two standard deviation measures of $F_1$ score over the contexts $a$ and $b$ shows that *Nereau*
is able to disambiguate the same term over both the considered contexts, while CoOcc obtains more dispersion from the average precision.

<table>
<thead>
<tr>
<th></th>
<th>RF</th>
<th>CoOcc</th>
<th>Nereau</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg P</td>
<td>0.5</td>
<td>0.58</td>
<td>0.60</td>
</tr>
<tr>
<td>Avg R</td>
<td>0.41</td>
<td>0.45</td>
<td>0.50</td>
</tr>
<tr>
<td>Avg F1</td>
<td>0.39</td>
<td>0.44</td>
<td>0.51</td>
</tr>
<tr>
<td>$\sigma_{F1}$</td>
<td>0.26</td>
<td>0.29</td>
<td>0.23</td>
</tr>
</tbody>
</table>

3.4. Qualitative Analysis

User Ratings

Collecting subjective measures is a valuable way of measuring participant feelings towards satisfaction. For this reason, Nereau was installed and configured on a Web server openly accessible through the Internet. We asked the 42 users to use the search engine in their daily activities for a four-week period. The system started monitoring the users actions (e.g., visited pages, submitted queries) and profiles. When Nereau was able to collect enough data about the user, it started proposing potential query expansions in a different frame of the search page. At this point, the user was able to submit an estimation of the value of the proposed expansion in terms of suitability of the terms for the purpose of the needs underlying the given query. The judgments for the proposed expansions were in a five-point Likert-type scale of values from 1 (*useless expansion*) to 5 (*very useful expansion*). Specifically, we asked each user to evaluate the following aspects:

1. the category relevance to his own interests;
2. the categorization accuracy.

A total of 313 votes submitted by users have been collected over a four-week period. The results are summarized in Figure 26. Most users found the suggested terms for expansion useful and related to the initial needs. Moreover, none of them found the expansions incorrect or inconsistent with his current needs.
4. RELATED WORK

In this section, we report a few works that are somehow related to the approach described in this paper.

The literature proposes several systems that do not perform query expansion but still take advantage of folksonomies to provide users with recommendations. In van Setten et al. [2006] the authors study the role of annotations in supporting users to discover relevant information, while in Xu et al. [2006] an approach to identify a set of tags in order to label a resource of a folksonomy is described. Carmagnola et al. [2007] discuss the contribution that the analysis of tagging activity can lead to the construction of user profiles. Jaschke et al. [2007] advance FolkRank, an approach to identify potentially relevant tags for a folksonomy user, and Zanardi and Capra [2008] put forward Social Ranking, a method to answer queries of folksonomy users through collaborative filtering technology.

Marinho et al. [2012], Marinho et al. [2011], and Milicevic et al. [2010] provide a comprehensive overview of the most recent works on social tagging recommender systems. In Marinho and Schmidt-Thieme [2007] the authors address the problem of tag recommendation from a collaborative filtering (CF) perspective. Through a simple and suitable protocol, they compared four tag recommenders, showing that a simple CF based on the user-tag profile matrix can provide significant improvements on the results of the baseline algorithms. Tso-Sutter et al. [2008] propose an approach to integrate tags into standard CF algorithms such as user- and item-based CF. They also advance a method to catch the three-dimensional correlations among users, items, and tags, by adopting a tag extension mechanism and a fusion technique adapted from a predicting rating task to a predicting item task. In Tatu et al. [2008] the authors describe their natural language understanding approach to provide tag recommendations for bookmarks and publications. They leverage the textual content associated with bookmarks, documents (publications and Web pages), and users, and produce patterns within the concept space or the existing tag space. Li et al. [2008] put forward a social interest discovery approach based on user-generated tags. They have realized an Internet Social Interest Discovery (ISID) system aimed to capture the common user interests and to cluster users and their saved URLs by different interest topics. Their results show that ISID can effectively cluster similar documents based on interest topics and discover user communities with similar interests regardless of whether or not they have online or offline social connections. Jáschke et al. [2007; 2008] present three different classes of algorithms for tag recommendation in folksonomies: simple adaptations of collaborative filtering based on user-tag and user-resource projections, adaptations of the well-known PageRank algorithm, and simple methods based on counting the most popular tags. The results of an evaluation performed on large-scale real-world datasets show that the leverage of the hypergraph structure in FolkRank provides a significant advantage and that simple methods based on tag counts can yield results almost as good as the best results with only minimal computational costs. Shepitsen et al. [2008] present an algorithm for customizing the recommendations in folksonomies, which relies on hierarchical tag clusters. Their basic recommendation framework is independent of the clustering algorithm, but the authors employ a context-dependent variant of hierarchical agglomerative clustering, which depends on the users current navigation context in the cluster selection. The evaluation results show that data sparsity can significantly influence the quality of the cluster selection process, thereby affecting the recommendation accuracy. A learning framework for automatic tag recommendation in real-time is proposed in Song et al. [2008]. Tagged training documents are represented as triplets (words, docs, tags), and organized in two bipartite graphs that are partitioned into clusters through spectral recursive embedding. Tags in each topical cluster...
are ranked by means of the algorithm advanced by the authors. At the same time, the document distribution is modeled into mixture components within each cluster and words are aggregated into word clusters by a two-way Poisson mixture model. The mixture model ranks a new document based on its posterior probabilities so that tags are recommended according to their ranks. In Sen et al. [2009] the term tagommender is used for the first time to indicate a recommendation algorithm that predicts users' preferences for items based on their inferred preferences for tags. A traditional recommender system yields predictions for movies on the basis of clicks and movie ratings. Differently, the movie tagommender proposed by the authors first infers users' preferences for tags, based on which then provides movie recommendations. In order to infer users' preferences for tags, the system takes advantage of signals of interest in tags (searches, tag applications) and signals of interest in items (clicks, movie ratings). To this aim, item signals need to be translated into tag signals. Illig et al. [2011] report an evaluation of different algorithms of content-based tag recommendation in a cold-start scenario on a large real-world dataset - a crawl of the delicious bookmarking system. The cold-start problem occurs when a bookmark is uploaded for the first time and therefore no information provided by other users can be used. Specifically, the authors found that an one-vs-one Support Vector Machine variant on length normalized document feature vector is the most effective algorithm among all the evaluated classifiers. The authors could thus prove that tag assignments can be learned by applying Machine Learning techniques to cope with the cold-start problem of collaborative recommender systems. Rendle et al. [2009] advance a tag recommender based on a tensor factorization model, which can thus take advantage of the ternary relationship in tagging data. Such approach, called Ranking with Tensor Factorization (RTF), allows the optimization of the factorization model for the best personalized ranking. RTF is able to deal with missing values and learns from pairwise ranking constraints. The optimization problem is solved through a gradient descent algorithm. The authors also provide a learning and prediction method with runtime complexity analysis for RTF. The prediction runtime of RTF does not depend on the number of observations, but only on the factorization dimensions. Symenonidis et al. [2008; 2010] propose an unified framework to model the three types of entities existing in a social tagging system: items, tags, and users. These three-dimensional data are represented by three-dimensional matrices, called three-order tensors, on which latent semantic analysis and dimensionality reduction are carried out through the higher order singular value decomposition method and the Kernel-SVD smoothing technique. The results obtained by the proposed approach show significant improvements in terms of effectiveness measured through standard parameters such as recall and precision, so revealing its capability to catch the users' multimodal perception of items, tags, and users.

As regards automatic query expansion (QE), such technique has been widely used in Information Retrieval. Among the various QE approaches proposed in literature, some of them take advantage of the implicit relevance feedback through pseudo-relevance feedback (PRF) [Baeza-Yates and Ribeiro-Neto 1999]. All these methods follow the basic assumption: documents classified higher by an initial search contain many useful terms that can help discriminate relevant documents from irrelevant ones. Despite the large number of studies, a crucial issue is that the expansion terms identified through traditional methodologies from the pseudo-relevant documents may not be all useful [Cao et al. 2008]. Bilotti et al. [2004] analyze the effect of some QE approaches on document retrieval in the context of question answering, mainly targeted to the so-called “factoid” questions, namely, fact-based, natural language questions that usually can be answered by a short noun phrase. More specifically, the authors describe a quantitative comparative analysis between two different strategies for tackling term
variation: i) employing a stemming algorithm at indexing time, or ii) carrying out a morphological query expansion at retrieval time. The findings show that, when compared to the baseline (no stemming nor expansion), stemming yields a lower recall, while morphological expansion results in higher recall. However, higher recall is paid at the cost of retrieving more irrelevant documents and ranking relevant documents at lower positions. One of the failure reasons of the query expansion has been identified in the lack of relevant documents in the local collection. Consequently, some works advance the use of an external resource for query expansion in order to improve the effectiveness of query expansion, such as thesaurus [Nanba 2007], Wikipedia [Xu et al. 2009], and search engine query logs [Cui et al. 2003]. Abouenour et al. [2010] point out that the adoption of a thesaurus, typically constructed through statistical techniques, poses several drawbacks. First of all, the construction of a thesaurus is time-consuming because of the great deal of data to process. Effective semantic QE techniques can also rely on ontologies instead of thesauri. Indeed, ontologies describe both semantic and concept relations, and enable semantic reasoning as well as cross-language information retrieval. The authors specifically deal with the enhancement of question answering in Arabic, a complex language for its peculiarities. They propose an approach that implements a semantic QE based on the WordNet ontology in Arabic. As a result, the described QE method bears the following semantic relations: synonymy, hyponymy (supertypes), hypernymy (subtypes), and the Super Upper Merged Ontology (SUMO) concept definition [Niles and Pease 2003]. SUMO is a top-level ontology that defines general terms and can be used as a foundation for middle-level and more specific domain ontologies. The documents retrieved through the previous process are then re-ranked using a structure-based approach based on the Distance Density n-gram model. The results of experiments performed on TREC and CLEF translated questions are evidence of a significant improvement of performance in terms of accuracy, Mean Reciprocal Rank, and number of answered questions. Recently, several authors have focused on social annotations as external resource, largely motivated by their increasing availability through many Web-based applications. Among these, Carman et al. [2009] explore how useful tag data may be to improve search results, but they focus primarily on data analysis rather than retrieval experiments. Our idea to take advantage of folksonomies arises from such considerations.

In literature we have found three approaches that, like our system, exploit the potential provided by social annotations [De Meo et al. 2010; Bouadjenek et al. 2011; Lin et al. 2011]. In the following, we discuss these systems in more detail, identifying for each of them similarities and differences with our approach. The first approach is proposed in De Meo et al. [2010]. 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 De Meo et al. [2010] 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 expansion of the user queries and the update of user profiles enable any content-based recommendation system exploiting a folksonomy to find and suggest resources corresponding to

---

5www.ontologyportal.org
6www.clef-initiative.eu/dataset/corpus
7The Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks over a set of queries. The reciprocal rank is the reciprocal of the rank at which the first relevant document is retrieved [Croft et al. 2010].
the user preferences and needs. Hence, this approach addresses the limitations of traditional content-based systems. Furthermore, enhanced user profiles may enable any collaborative filtering system to identify and recommend to a user resources that are likely to be relevant to him, even though he has not explicitly sought them. There are some similarities between our approach and the one proposed in De Meo et al. [2010]. In particular, both of them (i) maintain a tag-based profile for each user; (ii) exploit the \(<\text{user, resource, tag}>\) relationships to extract tags belonging to the same semantic class. As for the main differences, we observe that: (i) in De Meo et al. [2010] tags are ranked through PageRank, while in our approach they are ranked by evaluating co-occurrences in the Web documents visited by the user; (ii) in De Meo et al. [2010] the user has to add tags to resources and/or select the most relevant tags to them; in other terms, the user provides an explicit feedback. In our approach, we consider only implicit feedback, which is collected on the basis of user clicks on the result regarded closer to the submitted query. The evaluation of tag authoritativeness entails an off-line data analysis, while our system works on-line.

The second QE system that relies on social annotations to improve its performance is described in Bouadjenek et al. [2011]. 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) [Bao et al. 2007]. Based on these considerations, the reliability of an entity \(e\) is given by \(-\log(\text{SPR}(e))\). The system described in Bouadjenek et al. [2011] is the one that shares the largest number of similarities with our system. More specifically, both the approaches (i) maintain a tag-based profile for each user; (ii) provide a method to rank tags; (iii) exploit the \(<\text{user, resource, tag}>\) relationships to extract tags belonging to the same semantic class; (iv) are able to disambiguate tags. As for the main differences, we observe that in Bouadjenek et al. [2011] the authors evaluate the reliability of an entity through an algorithm derived from PageRank, which requires the knowledge of the network topology, which has to be calculated and maintained over time. Therefore, the approach is not dynamic because the computation of SPR indices is not incremental, so the folksonomy may grow and/or change without being considered by the system. On the contrary, our algorithm is incremental and takes into account possible temporal dynamics in the folksonomy.

The last QE system identified in literature, which exploits the potential of social annotations to enhance its effectiveness, is presented in Lin et al. [2011]. 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 [Cao and yao Liu 2007] 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 through 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 $Q$. As for the term-dependency method, the authors make two term-dependence assumptions of query terms: full independence and sequential dependence. The first variant, which underlies many IR models, assumes that query terms are independent to each other; the second variant assumes the dependence between neighboring query terms. There are some similarities between our approach and the one proposed in Lin et al. [2011]. Specifically, both of the approaches (i) employ a term ranking approach; (ii) use IR techniques to select the most relevant terms to be added to the original query; (iii) consider the statistical dependence between query terms, which provides significant benefits in terms of retrieval performance. As for the main differences, we observe that (i) the approach in Lin et al. [2011] does not consider a user profile, so not being able to provide personalized query expansions; (ii) our approach works on-line by querying social bookmarking services based on user requests at a given time. There is no off-line processing that allows us to make a selection of terms belonging to the re-ranked list, as in Lin et al. [2011]. Our approach offers the advantage of being able to increase over time the user model on folksonomy, so avoiding the cold-start problem.

5. CONCLUSIONS

In this paper we have proposed a new weak semantic technique for providing social and personalized query expansions. In particular, our query expansion approach 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. The expansion process takes place by analyzing multiple occurrences divided into categories related to semantic classes, which are analyzed in the folksonomy. We have presented the results of an in-depth 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. Moreover, we have also described a computational complexity analysis, whose results show the capability of our system to operate in real-time. 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 pages. 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 functions, 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


ACM Transactions on Intelligent Systems and Technology, Vol. 1, No. 1, Article 1, Publication date: January 2013.
Social Semantic Query Expansion


IMRAN, H. AND SHARAN, A. 2010. Selecting effective expansion terms for better information retrieval. IJCSA 7, 2, 52–64.


