LAICOS: An Open Source Platform for Personalized Social Web Search

Mohamed Reda
Bouadjenek
PRISM Laboratory
Versailles University
mrb@prism.uvsq.fr

Hakim Hacid*
Sidetrade
114 Rue Gallieni, 92100
Boulogne-Billancourt, France
hhacid@sidetrade.com

Mokrane Bouzeghoub
PRISM Laboratory
Versailles University
mokrane.bouzeghoub@prism.uvsq.fr

ABSTRACT

In this paper, we introduce LAICOS, a social Web search engine as a contribution to the growing area of Social Information Retrieval (SIR). Social information and personalization are at the heart of LAICOS. On the one hand, the social context of documents is added as a layer to their textual content traditionally used for indexing to provide Personalized Social Document Representations. On the other hand, the social context of users is used for the query expansion process using the Personalized Social Query Expansion framework (PSQE) proposed in our earlier works. We describe the different components of the system while relying on social bookmarking systems as a source of social information for personalizing and enhancing the IR process. We show how the internal structure of indexes as well as the query expansion process operated using social information.

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


Keywords: Information Retrieval, Social networks.

1. INTRODUCTION

The Web 2.0 has strengthened end-users position in the Web through their integration in the heart of the ecosystem of content generation. This has been made possible mainly through the availability of tools such as social networks, social bookmarking systems, social news sites, etc. impacting the way information is produced, processed, and consumed by both humans and machines. As a result, on the one hand, the user is no longer able to digest the large quantity of information he has access to, and is generally overwhelmed by it. On the other hand, this abundance of information captures in general an explicit feedback of users and represents an interesting opportunity for enhancing existing data management systems or proposing new ones.

Information Retrieval (IR) is performed every day in an obvious way over the Web [1], typically under a search engine. However, finding relevant information on the Web still becomes harder for end-users as: (i) usually, the user doesn’t necessarily know what he is looking for until he reaches it, and (ii) even if the user knows what he is looking for, he doesn’t always know how to formulate the right query to find it (except if in cases of navigational queries [2]). In existing IR systems, queries are usually interpreted and processed using document indexes and/or ontologies, which are hidden for users. The resulting document results are not necessarily relevant from an end-user perspective, in spite of the ranking performed by the Web search engine.

To improve the IR process and reduce the amount of irrelevant documents, there are mainly three possible improvement tracks: (i) query reformulation using extra knowledge, i.e. expansion or refinement of the user query, (ii) post filtering or re-ranking of the retrieved documents (based on the user profile or context), and (iii) improvement of the IR model, i.e. the way documents and queries are represented and matched to quantify their similarities.

In this demonstration, we introduce LAICOS, an open source Personalized Social Web Search Engine prototype, in which social information and personalization are at the heart of the IR process. Actually, this prototype is relying on social bookmarking systems as a source of social information for personalizing and enhancing the IR process but can be extended to use any source of social metadata, i.e. tweets, comments, etc.

2. ANATOMY OF LAICOS

Figure 1 illustrates the different components of LAICOS, which are: (i) a set of connectors, crawlers, and a database storing the data, (ii) data indexing engines, and (iii) a query processing engine. In the following, we briefly describe some of these components.

2.1 Crawlers in LAICOS

LAICOS possesses two crawlers: (i) a crawler for web pages based on Heritrix [3], which was specifically designed for web archiving and crawling. For each crawled web page, (ii) a folksonomy crawler engine will download all the annotations

*This work has been mainly done when the authors was at Bell Labs France, Centre de Villarceaux.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD ’13, August 11–14, 2013, Chicago, Illinois, USA.
Copyright 2013 ACM 978-1-4503-2174-7/13/08 ...$15.00.

1We also refer to documents as web pages or resources.
2http://crawler.archive.org/index.html
tions that are associated to it through API\[4\]. Annotations are recovered from social bookmarking systems especially, the delicious website\[4\].

### 2.2 Social Indexes in LAICOS

Crawled web pages and their social annotations are stored into a repository. Two indexing engines are responsible for indexing and keeping up to date the following index structures: (1) A *textual content based index* structure, which is based on indexing the collection of crawled documents using the *inverted index* structure. The Apache Lucene search engine leverages this part\[4\]. (2) A *social based index* structure, which is based on the crawled annotations assigned by users to web pages in social bookmarking websites. We implement our own indexing engine and structure for this part. The *textual-content based index* structure is well described in \[12\]. Hence, we only describe the *social-based index* structure of LAICOS. This index consists of the following seven main data structures (See Table 1 for the internal format and compression used for each structure):

- **Docs**: it stores web pages’ ids (md5 hash of a web page name), the number of tags and users associated to the web page, as well as the offset in the Docs_Users posting list.
- **Tags**: it stores tags’ ids (md5 hash of the tag text), the number of web pages and users associated to the tag, and the offset in the Tags_Docs posting list.
- **Users**: it stores users’ ids (md5 hash of the user name), the amount of web pages and tags associated to the user, and the offset in the Users_Tags posting list.
- **Docs_Users**: it stores the posting list of users for web pages. In particular, for each web page, this structure stores: the id of the user who tags this web page, the amount of tags he has used to annotate this web page, and the offset in the Bookmarks posting list.
- **Tags_Docs**: it stores the posting list of web pages for tags. In particular, for each tag, this structure stores: the id of the web page which is tagged with this tag and the amount of users who have used this tag to annotate this web page.
- **Users_Tags**: it stores the posting list of tags for users. In particular, for each user, this structure stores: the id of the tag used by this user and the amount of web pages tagged by this user with this considered tag.
- **Bookmarks**: it stores the posting list of tags for a document and a user. In particular, for each unique pair of web page and a user, this structure stores: the ids of tags used by this user to annotate this web page.

![Figure 1: Architecture of LAICOS.](image)

### Table 1: Details on the format and compression used for each index data structure. The numbers in parenthesis are the size of each entry in bytes.

<table>
<thead>
<tr>
<th>Structure</th>
<th>Contents</th>
<th>Structure</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Docs</td>
<td>id (32)</td>
<td>Tags</td>
<td>id (32)</td>
</tr>
<tr>
<td>(44)</td>
<td>Number of users (4)</td>
<td>(44)</td>
<td>Number of users (4)</td>
</tr>
<tr>
<td></td>
<td>Number of tags (4)</td>
<td></td>
<td>Number of web pages (4)</td>
</tr>
<tr>
<td></td>
<td>Byte offset in</td>
<td></td>
<td>Byte offset in Tags_Docs (4)</td>
</tr>
<tr>
<td>Docs_Users</td>
<td>id (32)</td>
<td>Users</td>
<td>id (32)</td>
</tr>
<tr>
<td>(44)</td>
<td>Number of web pages (4)</td>
<td>(44)</td>
<td>Number of tags (4)</td>
</tr>
<tr>
<td></td>
<td>Number of tags (4)</td>
<td></td>
<td>Number of bookmarks (4)</td>
</tr>
<tr>
<td></td>
<td>Byte offset in</td>
<td></td>
<td>Byte offset in Users_Tags (4)</td>
</tr>
<tr>
<td>Tags_Docs</td>
<td>docid (32)</td>
<td>User_Tags</td>
<td>tagid (32)</td>
</tr>
<tr>
<td>(36)</td>
<td>Number of users (4)</td>
<td>(36)</td>
<td>Number of web pages (4)</td>
</tr>
<tr>
<td>Bookmarks</td>
<td>(36)</td>
<td>(36)</td>
<td></td>
</tr>
</tbody>
</table>

### 2.3 Query pre-processing engine in LAICOS

In IR systems, queries are usually pre-processed by being reformulated. This process includes either: (i) the reduction of queries, which is a technique to reduce long queries to more effective ones \[11\], or (ii) expansion of queries, which consists of enriching the user’s initial query with additional information \[10\].

In LAICOS, queries are interpreted and processed using the Personalized Social Query Expansion (PSQE) framework \[9\]. In order to achieve a social and a personalized expansions of a query, PSQE considers: (i) the semantic similarity between candidate terms and the query, and (ii) the extent to which the candidate terms are likely to be interesting to the user. The experiments performed on the PSQE framework in \[9\] show that it enhances web search significantly.

### 2.4 IR model in LAICOS

Modeling in IR consists of two main tasks \[1\]: (i) the definition of a conceptual model to represent documents and queries and (ii) the definition of a ranking function to quantify the similarities among documents and queries.

To model the textual content of documents, LAICOS is based on the Apache Lucene search engine. However, to model the social annotations associated to the corpus of

---

\[1\]http://delicious.com/developers
\[2\]http://delicious.com/
\[3\]http://lucene.apache.org/core/
documents. LAICOS uses the Personalized Social Document Representation (PSDR) framework described in [6].

Basically, PSDR is a framework that uses social information to enhance, improve and provide a personalized social representation of documents to each user. Briefly, when a user submits a query, the framework constructs, on the fly, a PSDR of all documents that potentially match the query based on other users’ experience (while considering both users that are socially close to the query issuer and relevant to documents). The experiments performed in [6] show that the PSDR framework enhances web search significantly.

2.5 Ranking model in LAICOS

In the classical non-personalized search engines, the relevance between a query and a document is assumed to be only based on the textual content of the document. However, relevance actually depends on each user [14]. Thus, only query terms matching with the textual content of documents is not enough to generate satisfactory search results for various users.

In LAICOS, the ranking score for a document $d$ that appears in the results list obtained when a user $u$ issues a query $q$ is computed using the $SoPRA$ ranking function as follows:

$$Rank(d, q, u) = \gamma \times Sim(q, Sd,u) + (1 - \gamma) \times \left[ \beta \times Sim(p_u, Sd,u) + (1 - \beta) \times Sim(q, d) \right]$$

(1)

where, $\gamma$ and $\beta$ are weights that satisfy $0 \leq (\gamma, \beta) \leq 1$. $SES(d)$ is the Search Engine Score (SES) given to the document $d$, $Sd,u$ is the PSDR of the document $d$ according to the user $u$, and $p_u$ is the profile of $u$.

Inspired by the Vectorial Space Model, queries, profiles, documents and their PSDR are modeled as vectors. Hence, the similarities between these vectors are computed using the cosine measure. At the end of this process we obtain a list of ranked documents according to: (i) a textual content matching score of documents and the query, (ii) a social matching score of documents and the query, and (iii) the social interest score of the user to documents.

3. LIFECYCLE OF A USER QUERY

The on-line IR sub-process, which is illustrated in the right part of Figure [1] takes in charge the user query in LAICOS. The query is sent in the form of keywords using the main interface, which is similar to classical search engines. Then, the PSQE framework handles the query. The user is also able to activate and tune the parameters of the expansion process of PSQE through an interface as illustrated in Figure [2]. As an output of this first step, PSQE returns an adapted user query as a vector of weighted terms, which is expected to be as close as possible to the user’s information needs.

Next, the new query is processed by a retrieval engine, which retrieves all documents that contain the query terms in their textual content. This process is based on the Apache Lucene search engine. Then, a ranking score is computed for each retrieved document as described in Section [2.5]. The result list of documents is sorted based on this final ranking score from the most relevant to the less relevant one. Finally, the top ranked documents are formatted for presentation to the user as illustrated in Figure [3].

4. PERFORMANCE AND SCALABILITY

4.1 Performance of LAICOS

LAICOS has already been evaluated in an off-line study following the evaluation methodology proposed and used in [3, 5, 17] for assessing personalized search algorithms. Particularly, LAICOS has been compared to many social ranking algorithms including [2, 3, 4, 10, 11, 15, 17]. As illustrated in Figure [3], the results show significant benefits of the LAICOS social search engine compared to the closest state of the art approaches for ranking algorithms. Especially, our approach outperforms the LDA-P approach and the Xu08 approach, which we consider as the closest works to the LAICOS ranking function (LDA-P is described in [6]).

4.2 Scalability of LAICOS

Currently, the main complexity of LAICOS is in its ability to compute a PSDR for documents while processing a query. Actually, this complexity scales linearly with the number of retrieved documents, which indicates that this approach can be applied to a very large datasets. By using parallel computation, we can easily and considerably reduce the execution
time. This is part of our future work to improve LAICOS. As an illustration, Figure 5 shows the execution time needed for processing queries according to the number of documents that they match w.r.t. several parameters. The queries and the users were randomly selected 10 times independently, and we report the average results each time. As depicted in Figure 5 none of these parameters have an impact on the execution time. This latter still scales linearly with the number of documents. The results are obtained on a MacBook Pro with a 2.8GHz Intel Core i7 CPU and 4GB 1333MHz DDR3 of RAM, running MacOS X Lion v10.7.4.

Figure 5: Execution time of LAICOS.

5. WHAT WILL BE DEMONSTRATED?

This demonstration is illustrated using documents indexed from a delicious dataset. This dataset is public, described and analyzed in [16]. The initial list of urls submitted to the LAICOS crawlers was the tagged web pages of this dataset. We setup the crawlers in such a way to remove all the non-english web pages, an operation performed using the Apache Tika toolkit. Note that all the web pages that return an http error code was considered as unavailable. Table 2 gives a description of the resulted corpus of data after our cleansing and pre-processing:

<table>
<thead>
<tr>
<th>Bookmarks</th>
<th>Users</th>
<th>Tags</th>
<th>Web pages</th>
<th>Unique terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 675 294</td>
<td>318 769</td>
<td>425 183</td>
<td>1 321 039</td>
<td>12 015 123</td>
</tr>
</tbody>
</table>

In this demonstration, we show mainly the retrieval performances of LAICOS by comparing it to many retrieval algorithms. Hence, we illustrate mainly the following search scenario:

1. A user is asked to register his delicious account, which is instantly crawled (or to use an existing one);
2. The user is then asked to issue a query. The results are presented as illustrated in Figure 3 where a summary of the social annotations of each document is given;
3. The user can then process the same query over different retrieval algorithms such as [2] [9] [13] [15] [17] to compare the obtained results;
4. Finally, the user can judge whether or not a document is relevant. This will serve to perform an end-user evaluation for LAICOS and the existing algorithms and to build a tested benchmark for evaluating SIR approaches.

6. CONCLUSION

This demo paper introduces LAICOS, a personalized social web search engine that includes the social context of both users and documents in the IR process. On the one hand, the social context of documents is added as a layer to the textual content traditionally used to index a collection of documents to provide a personalized social representation of documents using the PSDR framework. On the other hand, the social context of users is used for the query expansion process using the Personalized Social Query Expansion framework (PSQE) and for ranking purpose. The prototype of LAICOS is implemented using the Apache Lucene IR platform and will be soon available for researcher under a GNU General Public License. A strong characteristic of LAICOS is the fact that it is a modular system that allows easily developing, integrating and evaluating SIR approaches. In particular, eLAICOS is the module that is responsible for this ease integration and evaluation of SIR approaches, which is out of the scope of this paper. LAICOS is already available at http://159.217.144.88:8080/laicos-prototype/

7. REFERENCES


http://data.dai-labor.de/corpus/delicious/