Lookup, Explore, Discover: how DBpedia can improve your Web search

Roberto Mirizzi\textsuperscript{a}, Azzurra Ragone\textsuperscript{a} Tommaso Di Noia\textsuperscript{a} and Eugenio Di Sciascio\textsuperscript{a}
\textsuperscript{a}Politecnico di Bari, via Re David 200, 7015, Bari – Italy
E-mail: mirizzi@deemail.poliba.it, a.ragone@poliba.it, t.dinoia@poliba.it, disciascio@poliba.it

Abstract. The power of search is with no doubt one of the main aspects for the success of the Web. Differently from traditional searches in a library, search engines for digital libraries (as the Web is) allow to return results that are considerably effective. Nevertheless, current search engines are no more able to really help the user in all those tasks that go under the umbrella of exploratory search, where the user needs not only to perform a lookup operation but also to discover, understand and learn novel contents on complex topics while searching. Similar problems arise also with search exploiting classical keyword-based tagging techniques. In the paper we present LED, a web based system that aims to improve Web search by enabling users to properly explore knowledge associated to a query/keyword. We rely on DBpedia to explore the semantics of keywords and to guide the user during the exploration of search results.

1. Introduction

Thanks to the social paradigm of the Web 2.0 new tools and technologies emerged to allow the user to interact with on-line resources. By now, we are very familiar with the notion of annotating a web page as well as of tagging system, tag cloud, tag browsing just to cite a few. Surfing the wave of the social Web, several web search tools have been developed to help users to navigate through search results in a fast, simpler and efficient way combining traditional search facilities with the tag cloud exploration. Such tools are usually available as plug-ins (add-ons) for the most common browsers and the most relevant are currently: DeeperWeb\textsuperscript{1}, Search Cloudlet\textsuperscript{2}, SenseBot - Search Results Summarizer\textsuperscript{3}. The main aim of these plug-ins is to guide users in refining queries with terms which are somehow related with the ones they are looking thus exploring the results space. This usually happens when the user has not a clear idea of what she is looking for and (possibly) she needs some suggestions in the exploration process. The above mentioned tools enrich the page of search results from classical search engines, e.g., Google, with a tag cloud suggesting relevant keywords, so that the user may expand her query by adding new terms just clicking on tags in the cloud.

From the users perspective, a very interesting aspect related to the generation of the tag cloud in the page of search results is the combination of lookup search with exploratory browsing. As pointed out by Marchionini in [8] there are mainly two different ways of search strategies: lookup and exploratory search. The presentation of contextualized tag clouds combined with search results makes possible to combine the two search strategies. As a matter of fact, after a lookup in a search engine the user may explore the results space.

Basically, all the above cited tools exploit text retrieval techniques to suggest new tags looking for relevant keywords in the snippets of search engines results. Terms in the tag cloud are the most frequent ones in the result snippets. None of these

\textsuperscript{1}http://www.deeperweb.com
\textsuperscript{2}http://www.getcloudlet.com
\textsuperscript{3}http://www.sensebot.net
tools take into account the meaning/semantics associated to tags. In this paper we present LED (acronym of Lookup Explore Discover), a tool for exploratory search exploiting the knowledge encoded in the Web of data for the tag cloud generation. Although LED shares with the above mentioned tools all the main functionalities such as add/remove tags in order to refine the query, the way it generates the tag cloud is completely different:

- tags are semantically related to the query and not to the (partial) content of search results;
- each generated tag is associated to an RDF resource coming from the Web of Data (in our case we mainly rely on DBpedia [2]) with its associate semantics;
- tags are semantically related with each other;

Therefore, while for the other tools the main aim is to give to the user a “visual summary” of the search results, in LED the terms included in the tag cloud are the most relevant w.r.t. the typed keyword(s), not the most relevant w.r.t. the retrieved documents. Also for LED, the main purpose is to suggest keywords similar to the one(s) representing the query in order to allow users to refine it and narrow down the search results adding/removing new tags from the cloud. In LED the different dimension of tags in the cloud does not reflect the number of occurrences of the keywords in the search results but the semantic similarity of each tags w.r.t. the searched keyword(s). LED wants to favour a serendipitous approach [6], that is the fortuitous discovery of something which is not already included in the search results and that the user did not know to exist, but that could be as well of interest for her.

The innovative aspects of LED are:

- a novel approach to exploratory browsing that exploits the Semantic Web: in this first version of the tool we focus on DBpedia and present an intuitive way to explore this knowledge base;
- a web application that help users to refine queries for search engines with the aid of semantically generated suggestions;
- a meta-search engine and news collector that integrates results coming from different external sources in a single platform;
- a hybrid approach to rank resources in a RDF dataset (such as DBpedia): our system exploits both the graph nature of a RDF dataset and text-based techniques used in IR;
- a RESTful JSON API that can be accessed from any environment that allows HTTP request, useful both for building innovative web applications and for comparison with different algorithms.

The remainder of this paper is structured as follows: in Section 2 we present in detail the LED interface. Section 3 is focused on the back-end of the system. Section 4 shows how LED backend can be used as a semantic tagging system. In Section 5 we present the LED RESTful API. A discussion and a comparison of related work is done in Section 6. Conclusion and future work close the paper.

2. LED: Lookup Explore Discover

In this section we introduce LED, a web-based tool that helps users in query refinement on search engines and exploratory knowledge search in DBpedia. The front-end of the system is available at http://sisinflab.poliba.it/led/ and currently is focused on search in the IT domain.

In a nutshell, the web-interface is composed by three different areas. The first one (marked with (1) in Fig. 1) is mainly a keyword lookup: here the user types in some characters to obtain an auto-complete list of related keywords. The second area (marked with (2)) in Fig. 1 consists of a tag cloud (or a set of tag clouds) populated with automatically generated tags that are semantically related with the keyword entered by the user. Each of them corresponds to a DBpedia resource. In fact, the displayed tag is the value of the rdf:label of the corresponding resource. The user may also search for multiple keywords thanks to a tab-based structure of the area devoted to search results. As shown in Fig. 1, one tab for each searched keyword is open (RDFa and Microformat, respectively) and one with all the keywords (RDFa Microformat).

The tag cloud allows the user to discover new knowledge that is hidden in a traditional search engine. Thanks to LED it is also possible to explore knowledge, browsing through suggested tags, and refine the query by adding more specific keywords. Finally, the third areas of the interface (marked
with (3) in Fig. 1) acts as a meta-search engine: it displays the results coming from the three most popular search engines (i.e., Google, Yahoo! and Bing), the social network and microblogging service Twitter, the news aggregator Google News, Flickr and Google Images, according to the query formulated in the first section and refined in the second section.

2.1. Lookup

The entry point to LED is a text input field. Similarly to classical web search engines, through this field the user formulates her query as a set of keywords.

Various studies estimate the average length of a search query between 2.4 and 2.7 words [4]. Nevertheless a query made just of two or three keywords can convey only a small amount of information. Keeping in mind this assumption, search engines are tuned toward maximizing precision in the first results (i.e., minimizing the number of possibly irrelevant documents that are retrieved) [8]. Therefore, using additional external knowledge to refine the queries can be a good way to narrow down the search results and improve the user experience, maximizing the number of possibly relevant retrieved objects. In [11] the authors describe the main ideas behind a simple lookup search on a traditional search engine and a deeper (semantic-guided) exploratory search: a search engine allows to find what you are looking for and already know, while LED (and more generally exploratory search activities) allows to explore and discover what you probably did not know to exist.

When the user starts to type some characters in the text search field, the system returns a list of DBpedia resources that contain the entered string either in their rdf:label or in their dbpedia-owl:abstract. This list is populated by querying the DBpedia URI lookup web service\(^4\).

Fig. 1. The LED interface, available online at http://sisinflab.poliba.it/led/.
ther to the *programming language* or to the *island* (and other possible choices in addition).

The user can also ignore the suggestions and look for something that simply does not exist in DBpedia. This is allowed by LED: in this case the system will act as a standard meta-search engine/aggregator (see Section 2.3).

2.2. Exploratory browsing

The most interesting part of LED interface is the tag cloud (marked with (2) in Fig. 1). When the user selects a keyword (i.e., a DBpedia resource) from the autocomplete list, the system populates a cloud containing tags that are semantically related to it and the size of each tag reflects its semantic relevance w.r.t. the selected keyword (see Section 3). If a keyword does not exist in DBpedia, the knowledge base can not be used to discover and explore new knowledge, so in this case the tag cloud is empty. In order to overcome this issue, we are currently working integrating resources coming from more datasets in the Linked Data cloud (e.g., LinkedMDB for movies or DBLP for academic papers) as well as to exploit user behavior to compute related concepts.

When the user moves the mouse pointer over a keyword in the tag cloud, the dbpedia-owl:abstract of the corresponding DBpedia resource is shown in a tooltip. The abstract allows the user to better understand the meaning of the tag. Since the user reads the description of a tag directly into LED, there is no need to leave the application to obtain info about the specific tag. This could reduce trial-and-error tactics in browsing strategies, allowing to achieve the goal faster. By clicking on a tag in the cloud, the selected keyword becomes the new query and the associated results of the meta-search engine are updated in the meta-search area, as indicated with (3) in Fig. 1. Once the user clicks on a tag, the corresponding cloud is created in a new tab. Thanks to this feature the user can navigate the knowledge base in an intuitive way. Browsing among related tags helps the user to obtain a clearer idea about her query. When the user is looking for something on a search engine, it may happen that she has just a vague idea about how to express her query, and she does not know exactly what she is looking for. In this case LED is more powerful than a traditional search engine, because it allows to explore and discover new knowledge, starting from an initial idea of the user. For example, suppose you want to learn more about *microformats*5 and related technologies. In a traditional lookup search, you would enter the keyword *microformat* on a search engine to obtain a list of documents containing the keyword. Then, you should navigate through results by selection, navigation and trial-and-error tactics [8] to learn new information. On the contrary, if you enter *microformat* in LED, you get information about what *microformat* is, and its top-related technologies (with their description). As you can see from Fig. 1, the suggested keywords are highly relevant w.r.t. the selected one: in the cloud there is for example *HTML*, i.e., the markup language used to represent microformats in web pages, RDFa and GRDDL, that are the main alternatives to microformats to semantically annotate documents. As you can see from this example, it is possible to explore new knowledge with LED, moving from searching to learning.

Furthermore, LED allows the user to refine her query adding new tags in two different ways, either dragging and dropping them in the text input field or clicking on the plus icon near the tag. Indeed, moving the mouse over a keyword in the cloud, a small icon representing a plus sign (+) next to the keyword is shown (see (a) in Fig. 1). Clicking on it the new keyword is added to the original query. Once a new tag is added from the cloud, the tab corresponding to this new keyword is populated with its semantically related tags. Moreover a new tab is created, containing the most relevant tags w.r.t. all the concepts composing the new query (see (b) in Fig. 1). The relevance value of each tag in this new tab is calculated as the sum of the relevance values of the tags belonging to each separate keyword in the query. The third column of Table 1 shows the top results according to a query composed by RDFa and Microformat. In the first column we see that the semantic relevance of the pair (RDFa, Semantic Web) is evaluated as 0.89 while in the second one the relevance of (Microformat, Semantic Web) is 0.87. Hence, the semantic relevance of (RDFa Microformat, Semantic Web) is computed as 0.87 + 0.89. The query refinement is applied also to results from meta-search engine. Thanks to this feature the user can be more pro-

5http://microformats.org/about
ductive, saving time on her search activities\textsuperscript{6}. The user can also exclude a previously added keyword from the query simply clicking on the \texttimes{} icon that appears next to a tag that has already been added to the query.

2.3. The meta-search engine

The lower part of the LED interface displays results about the query formulated by the user. LED acts as a meta-search engine and news aggregator, sending user requests to external sources and aggregating them. The idea behind this activity is that more comprehensive search results can be obtained by combining the results from several sources as Google, Yahoo, Bing, Twitter, Flickr\textsuperscript{7}. The first three sources are used to retrieve a ranked list of documents indexed by search engines, allowing the user to immediately look at the differences among the results provided by the different search engines. Each result set is displayed in a separate tab. Another tab is dedicated to show the most recent contents coming from Twitter, the popular microblogging platform. Recent news from Google News related to the query formulated by the user are displayed in a separate tab. The last tab is dedicated to relevant images from Google Images and Flickr.

When the user specifies her query, the external sources are invoked through their REST APIs.

3. DBpediaRanker: RDF Ranking in DBpedia

In this section we will briefly describe our hybrid ranking algorithm DBpediaRanker, used to rank resources in DBpedia w.r.t. a given keyword. For a more comprehensive description of DBpediaRanker and for results of user evaluation the interested reader may refer to [10,9].

In a nutshell, DBpediaRanker explores the DBpedia graph and queries external information sources in order to compute a similarity value for each pair of resources reached during the exploration. The RDF graph browsing, and the consequent ranking of resources, is performed offline and, at the end, the result is a weighted graph where the nodes are DBpedia resources and the weights represent the similarity value between any pair of nodes. The graph so obtained will then be used at runtime by LED: (i) to suggest similar keywords to users for knowledge exploration and query refinement, and (ii) in the document selection phase, to display documents semantically related to the query (keywords) posed to the search engines (see Section 2).

The similarity value between any pair of resources in the DBpedia graph is computed querying external information sources (search engines and social bookmarking systems) and exploits textual and link analysis in DBpedia. For each pair of nodes in the explored graph, we perform a query to each external information source: we search for the number of returned web pages containing the labels of each nodes individually and then for the two labels together. Moreover, we look at, respectively, abstracts in Wikipedia and wikilinks, i.e., links between Wikipedia pages. Specifically, given two nodes $uri_1$ and $uri_2$, we check if the label of node $uri_1$ is contained in the abstract of node $uri_2$, and vice versa. The main assumption behind this check is that if a DBpedia resource name appears in the abstract of another DBpedia resource it is reasonable to think that the two resources are related with each other. For the same reason, we also check if the Wikipedia page of resource $uri_1$ has a link to the Wikipedia page of resource $uri_2$, and vice versa. The architecture of DBpediaRanker is depicted in Fig. 2.

4. LED as a service: Not Only Tag (NOT)

In this section we show how LED, with slight modifications of its interface and without changes in the back-end system, can become a semantic tagging system. The tool grown out of LED is NOT, which stands for Not Only Tag (http://sisinflab.poliba.it/not-only-tag). NOT (see Fig. 3) is the core of a semantic social tagging system for an effective and efficient annotation and retrieval of any type of Web resources. It can be used to recommend similar tags to users in

\textsuperscript{6}Just for lovers of statistics: if only 1\% of the World’s population used LED, and it helped to save each user 10 minutes per month on their searches, this would save globally over six billion of working hours per year (this phrase was inspired by \url{http://www.getcloudlet.com/swm.php?page=donate}).

\textsuperscript{7}Exploiting the available APIs exposed by content providers, we are currently expanding the set of information sources.
Table 1
The ten most relevant tags for RDFa, Microformat and RDFa Microformat, respectively.

<table>
<thead>
<tr>
<th>#</th>
<th>RDFa</th>
<th>Microformat</th>
<th>RDFa Microformat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GRDDL</td>
<td>0.95 HTML</td>
<td>0.94 GRDDL</td>
</tr>
<tr>
<td>2</td>
<td>Resource Description Framework</td>
<td>0.93 Cascading Style Sheet</td>
<td>0.91 HTML</td>
</tr>
<tr>
<td>3</td>
<td>XHTML</td>
<td>0.91 GRDDL</td>
<td>0.91 Semantic Web</td>
</tr>
<tr>
<td>4</td>
<td>HTML</td>
<td>0.91 Web development</td>
<td>0.90 Resource Description Framework</td>
</tr>
<tr>
<td>5</td>
<td>Embedded RDF</td>
<td>0.90 Web Technology</td>
<td>0.90 XHTML</td>
</tr>
<tr>
<td>6</td>
<td>Microformat</td>
<td>0.90 RDFa</td>
<td>0.90 Cascading Style Sheet</td>
</tr>
<tr>
<td>7</td>
<td>Semantic Web Services</td>
<td>0.89 XMDP</td>
<td>0.89 Embedded RDF</td>
</tr>
<tr>
<td>8</td>
<td>Semantic Web</td>
<td>0.89 Semantic network</td>
<td>0.88 Web development</td>
</tr>
<tr>
<td>9</td>
<td>XML</td>
<td>0.87 Web design</td>
<td>0.88 Web technology</td>
</tr>
<tr>
<td>10</td>
<td>Ontology</td>
<td>0.86 Semantic Web</td>
<td>0.87 Semantic Web Services</td>
</tr>
</tbody>
</table>

Fig. 2. The functional architecture of the whole system composed by the ranking back-end system DBpediaRanker and the LED interface.

Fig. 3. The NOT interface, available online at http://sisinflab.poliba.it/not-only-tag/

A lot of web sites use tags to annotate and retrieve content, just to cite a few, Programmable Web\(^8\) to tag APIs and mashups; Amazon\(^9\) for books, movies, music and games; Flickr\(^10\) for pictures, Delicious\(^11\) to store and retrieve favorite bookmarks, Youtube\(^12\) for videos, and many more. NOT could be exposed as a service (see Section 5) and integrated in one or more of these web sites, in order to make more efficient the tagging process. Indeed, in tagging we have two main issues:

- in the annotation phase: tagging is a time-consuming task. In order to make a resource easily retrievable the user should use different tags to match all the possible user queries;
- in the retrieval phase: an annotated resource is retrieved only if there is an exact match between the string representing the annotation and the searched tag.

As a way of example, we can refer to one of the several books tagged on Amazon, e.g., a book about semantic web technologies\(^13\), whose tags are: ontology, rdf, semantics, semanticweb, semantic web, semantic_web, owl. Looking at the tags it is clearly visible the misuse of tags, since the same tag (same meaning) is written three times in three different forms: **semantic web**, **semantic_web** and **semanticweb** in order to catch all the possible

---

8\[\text{www.programmableweb.com/}\]
9\[\text{www.amazon.com/}\]
10\[\text{www.flickr.com/}\]
11\[\text{www.delicious.com/}\]
12\[\text{www.youtube.com}\]
13\[\text{In this particular case we refer to the book by John Davies, Rudi Studer, and Paul Warren: Semantic Web Technologies: Trends and Research in Ontology-based Systems, accessed on Amazon the 26th of July, 2010.}\]
searches. Then there are more “specific” tags, as rdf, ontology and owl; furthermore, you can notice that if a book is about rdf, owl and ontology it should be automatically inferred by the system (without the need to be explicitly specified by the user) that is also about semantic web.

Moreover, an effective system should be able to return also approximate results w.r.t. the user’s query, results ranked based on the similarity of used tags to the user’s request. Referring to the previous example, it means that a book tagged only with owl should be retrieved and suggested as relevant when you are looking for books about e.g., semantic web, because of its similarity with the query, even if the exact searched tag is not present in its description.

Another issue strictly coupled with the keyword-based nature of current tagging systems on the web is synonymy. Different tags having the same meaning can be used to annotate the same content.

Here we show how NOT can help the user to select appropriate tags in the annotation phase. The interaction with the system is very simple and intuitive. Keeping the example of the book in Amazon, suppose the user wants to tag the same book we referred in the previous example using NOT. She will start to type some characters (i.e., “ontology lan”) in the text input area (marked as (1) in Fig. 3) and the system will suggest a list of DBpedia URIs whose labels or abstracts contain the typed string, in this case: Web Ontology Language, Ontology language, Ontology (information science), Multimedia Web Ontology language. At this point, the user may select one of the suggested items. Again, we stress here that the system does not suggest just a keyword but the suggested items. Moreover, all the tags automatically added by the system are colored in orange, to distinguish them from the ones explicitly added by the user (colored in red), and can be deleted if the user do not want to tag a resource with one of them, e.g., she can delete Data or Data management. Thanks to the RDF nature of DBpedia, categories in the tag-bag (indicated by (3) in Fig. 3) can be easily retrieved via nested SPARQL queries. In DBpedia, for each URI representing Wikipedia category there is a RDF triple having the URI as subject, rdf:type as property and skos:Concept as object. As an example for the Wikipedia category Semantic Web, in DBpedia we have:

dbpres:Category:Semantic_Web rdf:type skos:Concept

Actually, in order to expand the (semantic) keywords in the tag-bag, NOT also exploits the skos:broader and skos:subject properties within DBpedia. These two properties are used to represent an ontological classification among Wikipedia categories:

dbpres:Web_Ontology_Language skos:subject
dbpres:Category:Semantic_Web

dbpres:Category:Semantic_Web skos:broader
dbpres:Category:Knowledge_representation

The SPARQL query used to compute the expanded cloud related to a given resource is recursively repeated for all the related categories. For example, for the Web Ontology Language resource the first query is:
SELECT DISTINCT ?hasValue
WHERE {
  ?hasValue a skos:Concept.
  FILTER( isIRI(?hasValue) ).
  FILTER(?p = skos:subject || ?p = skos:broader)
}

5. API

The algorithms behind the tag cloud generation discussed in Section 2 and Section 4 and are also publicly available as a RESTful web service [12]. The web API is implemented over HTTP and is based on the principles of REST architecture. Our aim is to allow innovative web applications to exploit the results we obtained to build smarter tools on the top of our API. Moreover, being available on the Internet, it can be used by anyone for comparison with other metrics for different purposes, such as tag cloud enrichment, ranking of RDF graphs (in particular DBpedia), contextual tagging.

The LED API consists of two methods on a REST endpoint. To perform an action using the LED API, a request to the endpoint has to be sent specifying a method and its arguments. Currently the response is formatted in JSON, a machine-readable data-interchange format that makes easier the construction of applications in JavaScript. With the LED API it is also possible to bypass the same origin policy requesting the response to be formatted in JSONP.

The callable methods are:

- get/tags - fetch all tags related to a general keyword. This method returns a list of DBpedia resources that contain the searched keyword either in their rdfs:label or in theirs dbpedia-owl:abstract.
- get/related-tags - fetch all tags related to the selected one. This method returns a list of DBpedia resources that are related to a given resource. The user can get either any resource that is somehow related to the given one or only the resources that are ancestors of the given one, i.e., resources that are more general than the selected one. These last resources correspond to Wikipedia Categories.

Table 2 shows the details of the request parameters.

A sample response for the method get/tags is the following:

```javascript
callbackTags({
  "query": "php",
  "results": [
    {
      "label": "PHP",
      "uri": "http://dbpedia.org/resource/PHP"
    },
    {
      "label": "PhpBB",
      "uri": "http://dbpedia.org/resource/PhpBB"
    },
    {
      "label": "PEAR",
      "uri": "http://dbpedia.org/resource/PEAR"
    },
    {
      "label": "PhpMyAdmin",
      "uri": "http://dbpedia.org/resource/PhpMyAdmin"
    }
  ],
  "total": 10
})
```

Similarly, a sample response for the method get/related-tags is:

```javascript
callbackRelatedTags({
  "records": {
    "system": 42,
    "users": 0
  },
  "maxRelevance": 4.1037796302354,
  "main": {
    "uri": "http://dbpedia.org/resource/RDFa",
    "label": "RDFa",
    "description": "RDFa (or Resource Description Framework in attributes) is a ...",
    "relevance": 1.8909135979731,
    "normalizedRelevance": 5
  },
  "system": [
    {
      "uri": "http://dbpedia.org/resource/CGRDDL",
      "label": "CGRDDL",
      "description": "CGRDDL (pronounced 'griddle') is a markup format ...",
      "relevance": 4.0825179030441,
      "normalizedRelevance": 10
    },
    {
      "uri": "http://dbpedia.org/resource/Category:Semantic_Web",
      "label": "Semantic Web",
      "description": "",
      "relevance": 2.9482104095809,
      "normalizedRelevance": 7
    },
    ...
  ]
})
```

The value of relevance represents the similarity between two resources: the selected one and the other...
suggested one. For example, from the above code, the pair \((RDFa, GRDDL)\) has a relevance score of 4.082. The normalizedRelevance value represents the relevance value normalized to the max of the returned result set. It is comprised within a range of \([0,10]\) and varies at discrete steps. This value is used in LED and in NOT to scale the size of the tags in the generated cloud.

6. Related Work

The existence of semantic metadata and related ontologies in the Web of Data allows to develop new tools and approaches for data exploration and visualization. Nowadays, RDF datasets with their formalized knowledge are not only developed for machine-to-machine interaction but also for knowledge discovery and navigation. The whole knowledge available to machines could be exploited to help the user in her searching activities, easing the learn process. One of the main issues to be faced is how to manipulate these huge repositories of information in a “overview first, zoom and filter, then details-on-demand” fashion as pointed out in [13]. In [17] the author proposes Semantic Link Network (SLN), a semantic data model to semantically link resources and derive implicit semantic links according to a set of relational reasoning rules. Inspired by this model, we adopt a different approach to elicit hidden knowledge and infer new meaningful links between resources in a RDF graph. In [18] the notion of interactive semantics is introduced as “an open, self-organized and evolving social interactive system and its semantic image”. From this perspective our choice of relying on the Linked Data cloud and in particular on DBpedia well fits with the vision proposed by the author. Similarly to LED, in [14,15] an approach for query refinement is presented, however, differently from LED, a pure text-based is exploited to enrich Web searching with exploration services. Peng et al. [5] conduct a study about on how to enhance existing web search paradigms with tags. In particular they use tags in Delicious to improve Google search function: tags are distinguished in search keywords, useful for query refinement and exploration keywords, useful for exploratory purposes. However, in this case as well, no semantic information is exploited.

On the side of collaborative tagging systems (CTS), different approaches have been proposed, most of them suggest tags based on their popularity and rely on collaborative filtering algorithms. In [7] the authors propose an algorithm for tag recommendation in folksonomies based on graph exploration PageRank-like. Song Yang et al. [16] adopt a machine learning approach to tag recommendation; they propose a document-centered recommendation opposed to a user-centered one. However all these approaches do not take into account semantics, that is the fact that tags transport a meaning and are more than simple keywords as they are symbols for concepts.
Among semantic-based approaches, there is Faviki\(^{17}\), which shares some commonalities with our tool NOT. Faviki is a tool for social bookmarking that helps users to tag documents using DBpedia [2] terms extracted from Wikipedia. Although it is a good starting point to cope with synonymy, it does not solve the problem of ranking tags w.r.t. a query. Moreover it does not provide the user with any suggestion of tags related to the ones selected during the annotation phase, e.g., if the user tags a resource with \textit{owl}, the tool does not suggest to tag the page with \textit{rdf} too. ConTag [1] is a semantic tagging recommendation system which relies on the PIMO ontology, an ad-hoc ontology built to link tags to concepts. ConTag, as our tools, attaches a unique URI to tags, thus allows to cope with problems typical of current tagging systems as synonyms, homonyms, acronyms and different spelling. However, differently from our approach the recommendation consists in visualizing an alignment scheme instead of most related tags as in our case. TagClusters [3] is a tool for tag recommendation and tag-based retrieval: tags are firstly clustered and then the system suggests tags to users in the same clusters. After deriving the hierarchical structure of tags, the semantic similarity is calculated based on the co-occurrence. However, the evaluation of our system performed in [9] has shown that the co-occurrence formula could not be a good choice, e.g., when a resource is extremely more popular than the other.

7. Conclusion and future work

In this paper we presented LED, a system for exploratory search and query refinement in web searches. We motivated our approach in the search-engine scenario, showing how exploiting semantic information in DBpedia it is possible both (i) to help users in the process of query formulation and refinement, and (ii) to enhance the document selection process, displaying more specific documents related to the keywords entered in the search engine. We explained how LED can improve the way web searches are done nowadays and how user could save time using such tool.

We also introduced NOT and describe how it can be used as the core module for a semantic-based tagging system. Thanks to NOT we show how an approach based on the semantics of tags may help the user both in the annotation phase and in the retrieval one.

The back-end system of both LED and NOT is made also available as a RESTful web service.

Currently, we are mainly investigating how to deal with cases where a single knowledge base such as DBpedia is not sufficient to cover the informative demand. We intend to combine a content-based recommendation (based also on other available datasets) and a collaborative-filtering approach, in order to integrate the two stages to obtain more accurate and robust results. At the present moment, since LED has just been released, we do not have an active community using it, so LED concepts recommendation is only content-based. Nevertheless, the interface is ready to suggest keywords exploiting related searches from other users (as indicated with (c) in Fig. 1) exploiting a collaborative filtering approach. We are also interested in evaluating the usability of the proposed new interface for exploratory browsing with the support of the HCI community, in order to contribute to the design to the future search interfaces for the web.

References


\(^{17}\)http://www.faviki.com


