What if Exploratory Search and Web Search meet?*

Roberto Mirizzi, Azzurra Ragone, Tommaso Di Noia, and Eugenio Di Sciascio

Politecnico di Bari – Via Orabona, 4, 70125 Bari, Italy
mirizzi@deemail.poliba.it, {a.ragone,t.dinoia,disciacio}@poliba.it

Abstract. The power of search is with no doubt one of the main aspects for the success of the Web. Currently available search engines on the Web allow to return results with a high precision. Nevertheless, if we limit our attention only to lookup search we are missing another important search task. In exploratory search, the user is willing not only to find documents relevant with respect to her query but she is also interested in learning, discovering and understanding novel knowledge on complex and sometimes unknown topics.
In the paper we address this issue presenting LED, a web based system that aims to improve (lookup) Web search by enabling users to properly explore knowledge associated to her query. We rely on DBpedia to explore the semantics of keywords within the query thus suggesting potentially interesting related topics/keywords to the user.

1 Introduction

Surfing the wave of the social Web 2.0, several web search tools are available as plug-ins (add-ons) for the most common browsers that help users to navigate through search results in a fast, simple and efficient way. In a nutshell, they combine traditional search facilities with tag cloud exploration. Just to cite the most relevant we have: *DeeperWeb*[1], *Search Cloudlet*[2], *SenseBot - Search Results Summarizer*[3]. The main aim of these search tools is to help users in saving their time and refining queries with terms which are correlated to the ones they are looking for and that are mostly useful during an exploratory search [6]. This usually happens when the user has not a crystal clear idea of what she is looking for and (possibly) she needs suggestions on some directions in the exploration process. Indeed, the above mentioned tools couple the search results from classical search engines, e.g., Google, with a tag cloud suggesting relevant keywords, so that the user can expand her query by adding new terms just clicking on tags in the cloud. Some tools, like *Search Cloudlet*, can be integrated into Google, Yahoo! or Twitter web interface individually.

* This paper has been accepted for publication in *Proceedings of the Fourth Ph.D. Workshop in CIKM, PIKM 2010*, Toronto, Canada, October 30, 2010.

[1] http://www.deeperweb.com
[2] http://www.getcloudlet.com
[3] http://www.sensebot.net

Basically, all such tools exploit text retrieval techniques to suggest new tags looking for relevant keyword in the snippets of search engines results. Hence, terms in the tag cloud are simply the most frequent ones in the result snippets. None of these tools take into account the meaning/semantics associated to tags.

Here we present LED (acronym of ***L**ookup **D**iscover **E**xplore*), a tool for exploratory search exploiting semantic datasets of the Linked Data initiative [2, 1] for the tag cloud generation. Although LED shares with the other 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 with each other;
- 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 associated semantics;
- The semantics of the tags is related to the query and not to the (partial) content of search results;

Therefore, while for other tools the main aim is to give to the user a "visual summary" of search results, in LED the terms included in the tag cloud are the most relevant with respect to (w.r.t.) the typed keyword(s), not the most relevant w.r.t. the retrieved documents. For this reason the main purpose of LED is to suggest *similar* keywords to the one(s) typed in to allow users to refine their queries 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).

Main contributions of this work are:

- a novel approach to exploratory browsing that exploits the Semantic Web: in this work we focus on DBpedia and present an intuitive way to explore a knowledge base;
- a web application that help users to refine queries for search engines with the aid of semantics.

The remainder of this paper is structured as follows: in Section 2 we examine the main technologies behind our approach. In Section 3 we present in detail the interface of LED. Section 4 focuses on the back-end of the system and we give some flavors about experimental evaluation. A discussion and a comparison of related work is done in Section 5. Conclusion and future work close the paper.

2 Linked Data and DBpedia

The idea behind Linked Data [1] is about using the Web to allow linking data and easing the publication of new linked data on the Web. It describes a new method of exposing, connecting and sharing data through dereferenceable URIs on the Web. The goal is to extend the Web by publishing different open datasets as RDF triples on the Web and by setting RDF links between data items from

different data sources. According to this aim, URIs are fundamental to identify everything. Using HTTP URIs, things can be identified and looked up both by people and by agents. Linked Data uses RDF to describe things in the world.

DBpedia [2] is one of the main cloud of the Linked Data graph. It is the machine-understandable equivalent of the Wikipedia project. It is possible to pose queries to DBpedia (through its SPARQL endpoint http://dbpedia.org/sparql), and link other data sets on the web to DBpedia data. At the present moment DBpedia contains almost three million and half resources, including more than three hundred thousand persons, more than four hundred thousand places, thousands of films, companies, music albums, etc.. All this information is stored in RDF triples. The whole knowledge base consists of over one billion triples. DBpedia labels and abstracts of resources are stored in up to 92 different languages. The graph is highly connected to other RDF datasets of the Linked Data cloud. There are nearly half a million categories, inherited from Wikipedia, and almost one hundred thousand YAGO [11] categories.

Compared to other ontological hierarchies and taxonomies, DBpedia has the advantage that each term/resource is endowed with a rich description including a textual abstract. Another advantage compared to static hierarchies is that DBpedia evolves as Wikipedia changes. Hence, problems such as domain coverage, content freshness, machine-understand-ability can be addressed more easily when considering DBpedia. Each resource in DBpedia is referred by its own URI. This allows to precisely get a resource with no ambiguity. For example, the American corporation *Yahoo!* headquartered in California is referred to as the resource identified by the URI http://dbpedia.org/resource/Yahoo!, whereas the legendary *Yahoo* being in Gulliver's Travels is referred to as the URI http://dbpedia.org/resource/Yahoo_(Gulliver's_Travels).

Wikipedia authors try to organize articles by topics, clustering them into Categories[4]. They group together articles on related topics. Many categories contain several subcategories, thus representing a lightweight ontology. In order to avoid an explosion of articles belonging to a given category, articles are placed just in the most specific subcategories. For this reason, in Wikipedia it may be necessary a deep exploration to find all the categories an article belongs to. In DBpedia, the relations among categories and articles are elicited through the SKOS (***S**imple Knowledge **O**rganization **S**ystem*) vocabulary[5]. In particular skos:broader property links a category (that is the subject of a RDF triple) to its super-category, while skos:subject relates a resource to its corresponding Wikipedia category. Actually the skos:subject property is currently deprecated from the SKOS vocabulary, nevertheless it has not yet been replaced by any other property[6].

[4] http://en.wikipedia.org/wiki/Help:Category

[5] http://www.w3.org/2004/02/skos/

[6] A natural candidate for the replacement could be the dc:subject property, belonging to *Dublin Core* specification, as proposed in http://www.w3.org/2006/07/SWD/wiki/SkosDesign/Indexing.

3 LED: Lookup Explore Discover

In this section we introduce LED (acronym of ***L**ookup **E**x-plore **D**iscover*), a web-based tool that helps users in query refinement on search engines via exploratory search in DBpedia. The front-end of the system is available at `http://sisinflab.poliba.it/led/`. A sketch of the functional architecture is represented in Figure 2 while the algorithms behind LED interface are described in Section 4.

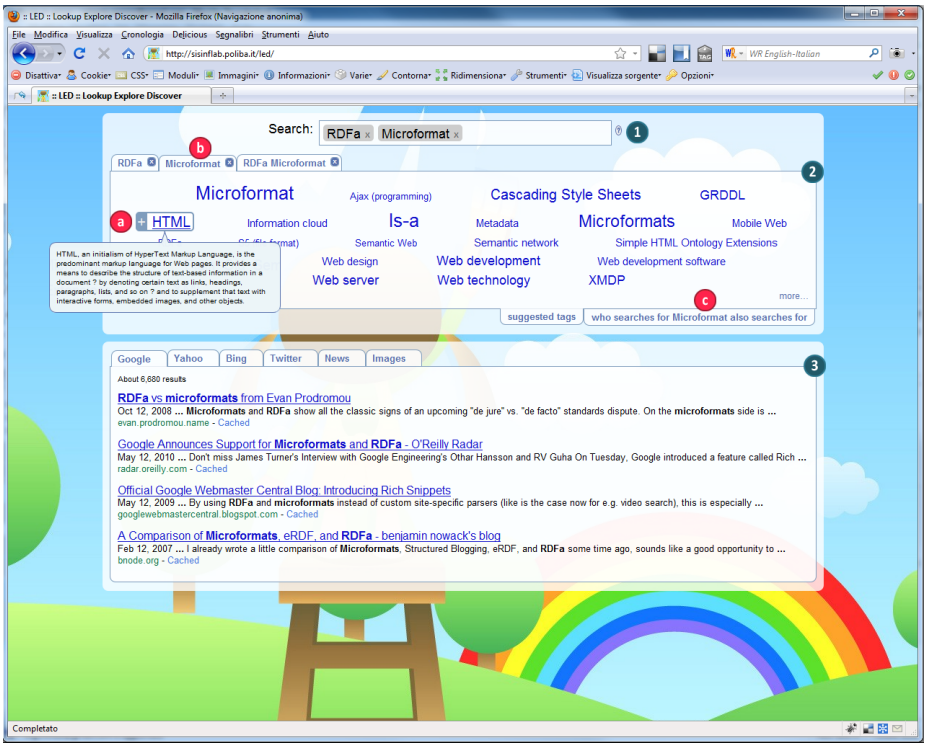

Fig. 1. The LED interface, available online at `http://sisinflab.poliba.it/led/`.

In a nutshell, the web-interface is composed by three different areas. The first one (marked with *(1)* in Figure 1) is basically an input text field for 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 Figure 1) consists of a tag cloud (or a set of tag clouds) populated with keywords that are semantically related with the selected one. 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 Figure 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.

In the following we will describe some technical details behind the three components of `LED`.

3.1 Lookup

The entry point to `LED` is a text input field (see *(1)* in Figure 1). Similarly to all 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[7] [3]. 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) [6]. Therefore, using additional external knowledge to refine the queries can be a good way to improve the search results and the user experience, maximizing the number of possibly relevant retrieved objects. In [9] the authors describe the main ideas behind a simple lookup search on a traditional search engine and a deeper (semantic-guided) exploratory search: a Web search engine allows the user to find what she already knows to exist while `LED` (and more generally exploratory search activities) allows the user to explore and discover what she probably did not know to exist.

When the user starts to type in 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[8]. The Web service is based on a Lucene index providing a weighted label lookup, which combines string similarity with a relevance ranking (similar to PageRank) in order to find the most likely matches for a given term [2]. The list of suggestions may help the user to disambiguate queries. For example, if you enter the word *Java*, the system allows to specify if you are referring either to the *programming language* or to the *island* (and other possible choices in addition).

The user is also allowed to 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 3.3).

[7] You can find more statistics searching on Google for `http://www.google.com/search?q=average+length+of+a+search+query`.

[8] `http://lookup.dbpedia.org/api/search.asmx`

3.2 Exploratory browsing

The most interesting part of LED interface is the central one (marked with *(2)* in Figure 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 (in Section 4 we detail how the similarity value between pairs of DBpedia concepts is computed). 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. For the future, to overcome this issue, we planned to exploit more datasets in the Linked Data cloud (e.g., linkedMDB for movies or DBLP for academic papers), and to exploit user behavior to determine 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.

If 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*[9] 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 [6] 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 Figure 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 quite easy to explore new knowledge with LED.

When the user clicks on a tag in the cloud, the clicked 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 Figure 1. Furthermore, LED allows the user to refine her query. Moving the mouse over a keyword in the cloud, a

[9] http://microformats.org/about

small icon representing a plus sign (+) next to the keyword is shown (see *(a)* in Figure 1). Clicking on it, or dragging it on the text input field, the new keyword is added to the original query and the tab corresponding to this new keyword is populated with its 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 Figure 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 $\langle RDFa, SemanticWeb \rangle$ is evaluated as 0.89 while in the second one the relevance of $\langle Microformat, SemanticWeb \rangle$ is 0.87. Hence, the semantic relevance of $\langle \{RDFa, Microformat\}, SemanticWeb \rangle$ is computed as $0.87 + 0.89$. The query refinement is applied also to results from the meta-search engine. Thanks to this feature the user can be more productive, saving time on her search activities[10]. The user can also exclude a previously added keyword from the query simply clicking on the $\times$ icon that appears next to a tag that has already been added to the query.

#	RDFa		Microformat		RDFa Microformat	
1.	GRDDL	0.95	HTML	0.94	GRDDL	1.86
2.	Resource Description Fram.	0.93	Cascading Style Sheet	0.91	HTML	1.85
3.	XHTML	0.91	GRDDL	0.91	Semantic Web	1.76
4.	HTML	0.91	Web development	0.90	Resource Description Fram.	0.93
5.	Embedded RDF	0.90	Web Technology	0.90	XHTML	0.91
6.	Microformat	0.90	RDFa	0.90	Cascading Style Sheet	0.91
7.	Semantic Web Services	0.89	XMDP	0.89	Embedded RDF	0.90
8.	Semantic Web	0.89	Semantic network	0.88	Web development	0.90
9.	XML	0.87	Web design	0.88	Web technology	0.90
10.	Ontology	0.86	Semantic Web	0.87	Semantic Web Services	0.89

Table 1. The ten most relevant tags for *RDFa*, *Microformat* and *RDFa Microformat*, respectively. The represented values are just for illustrative purpose.

At the present moment, since we do not have an active community using the system, `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 Figure 1).

3.3 The meta-search engine

The lower part of the `LED` interface displays results about the query formulated by the user and possibly refined. At this step `LED` acts as a meta-search engine and news aggregator, sending user requests to external sources and aggregating them.

[10] 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 months on their searches, this would save globally over six billion of working hours per year (this phrase was inspired by `http://www.getcloudlet.com/swm.php?page=donate`).

The results are collected from several external information sources, i.e., *Google*, *Yahoo*, *Bing*, *Twitter*[11]. 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 sources. 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.

4 DBpediaRanker: RDF Ranking in DBpedia

In this section we will 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* the interested reader may refer to [8].

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 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 3).

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 (as explained in the following). 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.

As the exploration starts from the seed nodes, a global variable $\mathcal{R}$ is initialized with the set of seed nodes and then it is further populated with other nodes

[11] Exploiting the available APIs exposed by content providers, we are currently expanding the set of information sources.

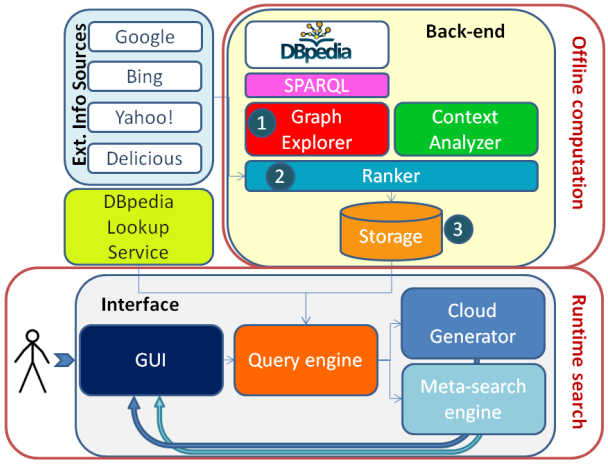

Fig. 2. The functional architecture of the whole system composed by the ranking back-end system `DBpediaRanker` and the `LED` interface.

reached during the graph exploration. If you want to explore only a domain specific portion of the whole `DBpedia` graph, context experts have to select the seed nodes. In the following we will present all the components of our system, whose architecture is sketched in Figure 2.

4.1 Graph Explorer

This module queries `DBpedia` via its `SPARQL` endpoint. Given a `DBpedia` URI, the explorer looks for other URIs connected to it via a set of predefined properties. The properties of `DBpedia` to be explored can be set in the system before the exploration starts. In our initial setting, we decided to select only the `SKOS` properties `skos:subject` and `skos:broader`. Indeed, these two properties are not specific of a particular context and are very popular in the `DBpedia` dataset. Hence, they can be used as a good starting point. Moreover, we observed that the majority of nodes reached by other properties were also reached by the selected properties, meaning that our choice of `skos:subject` and `skos:broader` properties does not disregard the effects of potentially domain-specific properties.

Given a root URI, this is explored up to a predefined distance, that can be configured in the initial settings. We found through a series of experiments that setting this distance, that we call MAX_DEPTH, equal to 2 is a good choice. Indeed, resources within two hops are still highly correlated to the root URI, while going to the third hop this correlation quickly decreases. Indeed, we noticed that if we set $MAX\_DEPTH = 1$ (this means considering just nodes directly linked) we loose many relevant relation between pairs of resources. On

the other hand, if we set $MAX\_DEPTH > 2$ we have too many non relevant resources.

In order to find the optimal value for MAX_DEPTH, we initially explored 100 seed nodes up to a $MAX\_DEPTH = 4$. After this exploration was completed, we retrieved the top-10 (most similar) related resources for each node (the details are provided in the following). The results showed that on the average the 85% of the top-10 related resources where within a distance of one or two hops. The resources two hops far from the seeds where considered as the most relevant the 43% of times ($\sigma = 0.52$). On the contrary the resources above two hops were rarely present among the first results (less than 15% of times). In Figure 3 the average percentage of top-10 related resources w.r.t. to the distance from a seed (MAX_DEPTH) is shown.

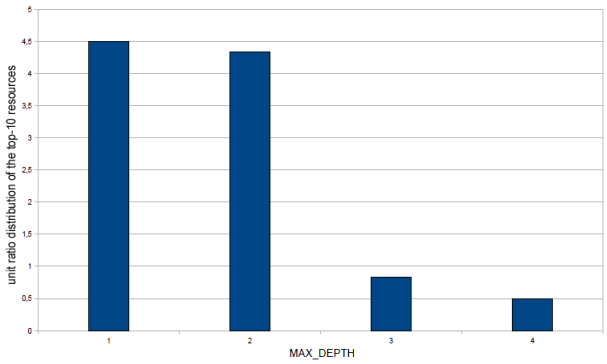

Fig. 3. Evaluation for MAX_DEPTH. It represents the average percentage (y axis) of the top-10 resources related to 100 seeds within a distance of 1 to 4 hops (x axis).

4.2 Ranker

Here we describe the ranker, the core component of the whole system. Given any pair of resources in the `DBpedia` graph it determines a similarity value between them; this similarity value is the weight associated to the edge between the two resources.

Given two URIs uri_1 and uri_2 in the same graph-path, it compares how much they relate with each other exploiting information sources external to `DBpedia` such as search engines and social tagging systems.

The aim of this module is to evaluate how strong is a semantic connection between two `DBpedia` resources using information taken from external sources. In our current implementation we consider as external sources both (i) web search engines (Google, Yahoo! and Bing) and (ii) social tagging systems (Delicious), plus (iii) Wikipedia-related information contained in `DBpedia`. Given

two DBpedia resources uri_1 and uri_2, we verify how many web pages contain (or have been tagged by) the value of the rdfs:label associated to uri_1 and uri_2. Then we compare these values with the number of pages containing (or tagged by) both labels. We select more than one search engine because we do not want to bind the result to a specific algorithm of a single search engine. Moreover, we want to rank a resource not only with respect to the popularity of related web pages on the web, but also considering the popularity of such resources among users (e.g., in Delicious). In this way we are able to combine two different perspectives on the popularity of a resource: the one related to the words occurring within web documents, the other one exploiting the social nature of the current web. Through formula (1) we evaluate the related similarity of two URIs uri_1 and uri_2 with respect to a given external information source is.

$$sim(uri_1, uri_2, is) = \left(\frac{p_{uri_1,uri_2}}{p_{uri_1}} + \frac{p_{uri_1,uri_2}}{p_{uri_2}} \right)_{is} \quad (1)$$

Given the information source $info\_source$, p_{uri_1} and p_{uri_2} represent the number of documents containing (or tagged by) the rdfs:label associated to uri_1 and uri_2 respectively, while p_{uri_1,uri_2} represents how many documents contain (or have been tagged by) both the label of uri_1 and uri_2. It is easy to see that the formula is symmetric and the returned value is in $[0, 2]$.

Ranker does not use only external information sources but exploits also further information from DBpedia. In fact, we also consider Wikipedia hypertextual links mapped in DBpedia by the property dbpprop:wikilink. Whenever in a Wikipedia document w_1 there is a hypertextual link to another Wikipedia document w_2, in DBpedia there is a dbpprop:wikilink from the corresponding resource URIs uri_1 to uri_2. Hence, if there is a dbpprop:wikilink from uri_1 to uri_2 and/or vice versa, we assume a stronger relation between the two resources. We evaluate the strength of the connection as follow:

$$wikiS(uri_1, uri_2) = \begin{cases} 0, \text{ no wikilink between } uri_1 \text{ and } uri_2; \\ 1, \text{ wikilink only from } uri_1 \text{ to } uri_2; \\ 1, \text{ wikilink only from } uri_2 \text{ to } uri_1; \\ 2, \text{ wikilink both from } uri_1 \text{ to } uri_2 \text{ and} \\ \quad \text{vice versa;} \end{cases}$$

Furthermore, given two resources uri_1 and uri_2, we check if the rdfs:label of uri_1 is contained in the dbpprop:abstract of uri_2 (and vice versa). Let n be the number of words composing the label of a resource and m the number of words composing the label which are also in the abstract, $abstractS(uri_1, uri_2) = \frac{m}{n}$, with $\frac{m}{n}$ in [0,1] as $m \leq n$. At the end, the similarity value between uri_1 and uri_2 is computed as the weighted sum of the functions:

$$
\begin{aligned}
similarity(uri_1, uri_2) = {} & w_{google} * sim(uri_1, uri_2, google) + \\
& w_{yahoo} * sim(uri_1, uri_2, yahoo) + \\
& w_{bing} * sim(uri_1, uri_2, bing) + \\
& w_{delicious} * sim(uri_1, uri_2, delicious) + \\
& w_{wikiS} * wikiS(uri_1, uri_2) + \\
& w_{abstract} * abstractS(uri_1, uri_2)
\end{aligned}
$$

In `LED` (see Section 3) $w_{google}, w_{yahoo}, w_{bing}, w_{delicious}, w_{wikiS}$ and $w_{abstract}$ are all set to 1.

4.3 Context Analyzer

The purpose of *Context Analyzer* is to identify a subset of `DBpedia` nodes representing a context of interest. For instance, if our system was applied to a vertical search-engine centered on, e.g., *databases* and *programming languages*, *DbpediaRanker* should explore and rank only the subgraph of `DBpedia` whose nodes are somehow related to *databases* and *programming languages* as well. This subgraph is what we call a **context**. The context is represented by the most popular Wikipedia *Categories* reached during the exploration of the graph, i.e., by the *Categories* that are encountered more often and for this reason are more interconnected to other nodes. For example, in the selected domain, the ten most popular categories are represented in Table 2.

#	**Resource**
1.	Programming languages
2.	Object-oriented programming languages
3.	Cross-platform software
4.	Software
5.	Programming language families
6.	Software by operating system
7.	Database management systems
8.	Scripting languages
9.	Free software projects
10.	Computer libraries

Table 2. The ten most popular `DBpedia` Categories after the analysis of five thousand resources.

Once we have a context $\mathcal{C}$, given a query represented by a `DBpedia` node *uri*, we check if *uri* belongs to the context $\mathcal{C}$, ranking *uri* w.r.t. to all the representative nodes of $\mathcal{C}$, i.e., the top *Categories*. Hence, for each new resource found

during the exploration, in order to evaluate if it is within or outside the context, we compare it with the most popular DBpedia categories in $\mathcal{C}$ (i.e., with the representative nodes of $\mathcal{C}$). If the score is greater than a given threshold, we consider the new resource within the context. The value $THRESHOLD$ is set manually. In order to find an optimal value for the threshold, we executed several test on the first fifty discovered nodes starting from seven seed nodes (see above). In these tests we considered different values for $THRESHOLD$, in the range $[1.0, 10.0]$, at intervals of 0.5. After manual evaluation we noticed that the best value for the context we analyzed is $THRESHOLD = 4.0$. Indeed, we observed that many non-relevant resources were considered as in context if the threshold was lower. On the contrary, a greater value of the threshold was too strict and blocked many relevant resources.

In order to identify and compute a context, we use *Graph Explorer* to browse the DBpedia graph, starting from an initial meaningful set of resources (**seed nodes** that in this case have to belong to the selected domain and must be selected by domain experts). In order to verify if there is an optimal number of initial seeds, we performed two tests, changing the number of the initial seed nodes. In the first test we set $seed\_nodes = 7$, in the second $seed\_nodes = 100$. In spite of the small number of seeds of the first test, we were able to discover approximatively the same domain-specific subgraph of DBpedia of the second test, due to highly interconnected nature of the graph.

4.4 Evaluation

In the experimental evaluation we compared our *DBpediaRanker* algorithm with other four different algorithms; some of them are just a variation of our algorithm but lack of some key features. In order to assess the quality of our algorithm we conducted a study where we asked to participants to rate the results returned by each algorithm.

Due to limited computational resources, at the present moment we have narrowed the DBpedia graph exploration through the *Context Analyzer* module (see Section 4.3) to the *ICT* domain and in particular *programming languages* and *databases*. So the area covered by this test is just *ICT*. Also the tag clouds of the LED interface (see Section 3.1) are not empty only if the searched keyword belongs to this domain.

The DBpedia resources that have been completely analyzed by *DBpediaRanker* are 8596. The off-line process to explore and rank the DBpedia subgraph took approximatively two weeks on a single machine with 4GB of RAM and a quad-core at 2.6GHz. Once the pairs of similar resources have been stored into a MySQL database, their retrieval is fast: given a DBpedia resource, its top-ranked neighbors are retrieved always in less than one second.

For each query, we presented five different rankings, each one corresponding to one of the ranking methods. The result lists consisted of the top ten results returned by the respective method. The test was performed by 50 volunteers during a period of two weeks. The users were Computer Science Engineering master students (last year), Ph.D. students and researchers belonging to the ICT

scientific community. For this reason, the testers can be considered IT domain experts. We determined the statistical significance of the collected data with the Wilcoxon test. The results of this test are detailed in [7] and give evidence to the fact that not only our algorithm performed always better than the others, but also that the (positive) differences between our ranking and the others are statistically significant.

5 Related Work

The existence of semantic metadata and related ontologies in the Semantic Web of Data allows to develop new tools and approaches for data exploration and visualization. `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 learning 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 [10]. In [15] 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. Differently from *SLN*, we adopt a different approach to elicit hidden knowledge and infer new meaningful links between resources in a `RDF` graph. In [16] the notion of *interactive semantics* is introduced as an "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 [12, 13] 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. [4] 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, even in this case, no semantic information is exploited.

On the side of collaborative tagging systems (CTS), different approaches have been proposed, most of them suggesting tags based on their popularity and rely on collaborative filtering algorithms. In [5] the authors propose an algorithm for tag recommendation in folkosonomies based on PageRank-style graph exploration. Song Yang et al. [14] adopt a machine learning approach to tag recommendations: 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 resources.

6 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, 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 described the components of our system and gave some clues about the validity of our approach through an experimental evaluation.

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 obtain more accurate and robust results. We are also interested in evaluating how our approach performs with real users and in comparing it w.r.t. other related works. Finally we plan to study 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.

Acknowledgment

This research has been supported by Apulia Strategic projects PS_092, PS_121, PS_025.

References

1. C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.
2. C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, July 2009.
3. E. Gabrilovich, A. Z. Broder, M. Fontoura, A. Joshi, V. Josifovski, L. Riedel, and T. Zhang. Classifying search queries using the web as a source of knowledge. *TWEB*, 3(2), 2009.
4. P. Han, Z. Wang, Z. Li, B. Kramer, and F. Yang. Substitution or complement: An empirical analysis on the impact of collaborative tagging on web search. In *WI '06: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 757–760, Washington, DC, USA, 2006. IEEE Computer Society.
5. R. Jäschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme. Tag recommendations in folksonomies. In *PKDD 2007: Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases*, pages 506–514, Berlin, Heidelberg, 2007. Springer-Verlag.
6. G. Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, 2006.

7. R. Mirizzi, A. Ragone, T. Di Noia, and E. Di Sciascio. Ranking the linked data: the case of dbpedia. In *10th International Conference on Web Engineering (ICWE 2010)*, 2010.
8. R. Mirizzi, A. Ragone, T. Di Noia, and E. Di Sciascio. Semantic tags generation and retrieval for online advertising. In *19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*, 2010.
9. R. Mirizzi, A. Ragone, T. Di Noia, and E. Di Sciascio. Semantic wonder cloud: exploratory search in dbpedia. In *2th International Workshop on Semantic Web Information Management (SWIM 2010)*, Lecture Notes in Computer Science. Springer-Verlag, 2010.
10. B. Shneiderman. *The Craft of Information Visualization: Readings and Reflections*, chapter The eyes have it: a task by data type taxonomy of information visualizations. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
11. F. Suchanek, G. Kasneci, and G. Weikum. YAGO: A core of semantic knowledge - unifying WordNet and Wikipedia. In *16th International World Wide Web Conference (WWW 2007)*, pages 697–706, 2007.
12. B. Velez, R. Weiss, M. A. Sheldon, and D. K. Gifford. Fast and effective query refinement. In *In Proc. of the 20th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 6–15, 1997.
13. M. L. Wilson, B. Kules, m. c. schraefel, and B. Shneiderman. From keyword search to exploration: Designing future search interfaces for the web. *Found. Trends Web Sci.*, 2(1):1–97, 2010.
14. S. Yang, Z. Lu, and L. Giles. Automatic tag recommendation algorithms for social recommender systems - microsoft research. *ACM Transactions on Web*, 2009.
15. H. Zhuge. Communities and emerging semantics in semantic link network: Discovery and learning. *IEEE Trans. on Knowl. and Data Eng.*, 21(6):785–799, 2009.
16. H. Zhuge. Interactive semantics. *Artificial Intelligence*, 174(2):190–204, 2010.