Semantic tagging for crowd computing

Roberto Mirizzi[1], Azzurra Ragone[1,2], Tommaso Di Noia[1], and Eugenio Di Sciascio[1]

[1] Politecnico di Bari – Via Orabona, 4, 70125 Bari, Italy
mirizzi@deemail.poliba.it, {ragone,dinoia,disciacio}@poliba.it
[2] University of Trento – Via Sommarive, 14, 38100 Povo (Trento), Italy
ragone@disi.unitn.it

(Extended Abstract)

Abstract. The recent blow up of crowd computing initiatives on the web calls for smarter methodologies and tools to annotate, query and explore repositories. There is the need for scalable techniques able to return also approximate results with respect to a given query as a ranked set of promising alternatives. In this paper we concentrate on annotation and retrieval of software components, exploiting semantic tagging relying on DBpedia. We propose a new hybrid methodology to rank resources in this dataset. Inputs of our ranking system are (i) the DBpedia dataset; (ii) external information sources such as classical search engine results, social tagging systems and wikipedia-related information. We compare our approach with other RDF similarity measures, proving the validity of our algorithm with an extensive evaluation involving real users.

1 Introduction

The emergence of the crowd computing initiative has brought on the web a new wave of tools enabling collaboration and sharing of ideas and projects, ranging from simple blogs to social networks, sharing software platforms and even mashups. As a way of example, we can refer to a platform to share software components, like `ProgrammableWeb`[3], where programmers can share APIs and mashups. When a user uploads a new piece of code, she tags it so that the component will be later easily retrievable by other users. Components can be retrieved through a keywords-based search or browsing across *categories*, *most popular* or *new updates*. The limits of such platforms, though very popular and spread out on the entire web, are the usual ones related to keywords-based retrieval systems, e.g., if the user is looking for a resource tagged with either *Drupal* or *Joomla!*, the resources tagged with *CMS* (*Content Management System*) will not be retrieved. For example, in ProgrammableWeb, APIs as *ThemeForest* and *Ecordia*[3] are tagged with *CMS* but not with *Drupal* nor *Joomla*, even if in their abstracts it is explicitly written that they are available also for the two specific CMSs.

[3] http://www.programmableweb.com, http://www.programmableweb.com/api/{themeforest|ecordia}

Partially inspired by Faviki[4], in this paper we propose a new hybrid approach to rank RDF resources within Linked Data [1], focusing in particular on DBpedia [3], which is part of the Linked Data Cloud. Given a query (tag), the system is able to retrieve a set of ranked resources (e.g., annotated software components) semantically related to the requested one.

A system able to compute a ranking among DBpedia nodes can be useful both in the **annotation** phase and in the **retrieval** one. On the one hand, while annotating a resource, the system will *suggest* new tags semantically related to the ones already elicited by the user. On the other hand, given a query formulated as a set of tags, the system will *return* also resources whose tags are *semantically* related to the ones representing the query. For instance, if a user is annotating an API for ProgrammableWeb with the tag *CMS* (which refers to DBpedia resource http://dbpedia.org/resource/Content_management_system), then the system will suggest related tags as *Drupal*, *Joomla* and *Magento* (each one related to their own DBpedia resource).

Main contributions of this work are:

- A tool for the semantic annotation of web resources, useful in both the *tagging* phase and in the *retrieval* one (see Section 2).
- A novel *hybrid* approach to rank resources on DBpedia w.r.t. a given query. Our system combines the advantages of a *semantic-based* approach (relying on a RDF graph) with the benefits of *text-based* IR approaches as it also exploits the results coming from the most popular *search engines* (Google, Yahoo!, Bing) and from a popular *social bookmarking system* (Delicious). Moreover, our ranking algorithm is enhanced by *textual and link analysis* (abstracts and wikilinks in DBpedia coming from Wikipedia).
- A *relative* ranking system: differently from PageRank-style algorithms, each node in the graph has not an importance value per se, but it is ranked w.r.t. its neighborhood nodes. That is, each node has a different importance value depending on the performed query. In our system we want to rank resources w.r.t. a given query by retrieving a ranked list of resources. For this reason we compute a weight for each mutual relation between resources, instead of a weight for the single resource, as in PageRank-style algorithms.
- An extensive evaluation of our algorithm with real users and comparison w.r.t. other four different ranking algorithms, which provides evidence of the quality of our approach.

The remainder of the paper is structured as follows: in Section 2 we introduce our motivating scenario and present a first implementation of the whole system, which is detailed in Section 3. Then, in Section 4 we show and discuss the results of experimental evaluation. In Section 5 we discuss related work with respect to our approach.

[4] http://www.faviki.com

2 Not Only Tag

In this section we describe a semantic social tagging system *Not Only Tag – NOT* (available at `http://sisinflab.poliba.it/not-only-tag`, see Figure 1) that can be used in the annotation and retrieval of software components.

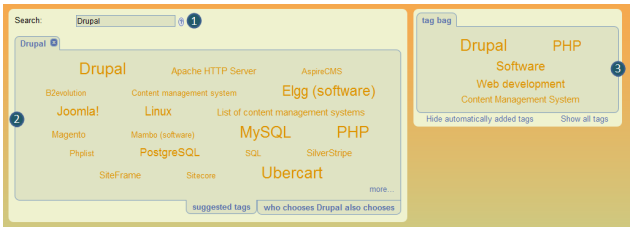

Fig. 1. Screenshot of *Not Only Tag* system.

The interaction with the system is very simple and intuitive. Let us suppose the user wants to annotate a software component. The user starts typing some characters (let us say "*Drup*") in the text input area (marked as *(1)* in Figure 1) and the system suggests a list of `DBpedia` URIs whose labels or abstracts contain the typed string. Then the user may select one of the suggested items. We stress here that the user does not suggest just a keyword but a `DBpedia` resource identified by a unique URI. Let us suppose that the choice is the tag *Drupal*, which corresponds to the URI `dbpres:Drupal`. The system populates a tag cloud (as shown in Figure 1 *(2)*), where the size of the tags reflects their relative **relevance** with respect to *Drupal* in this case (how the relevance is determined is explained in Section 3). We may see that the biggest tags are *Ubercart*, *PHP*, *MySQL*, *Elgg* and *Joomla!*. If the user goes with the mouse pointer over a tag, the abstract of the corresponding `DBpedia` resource appears in a tooltip. This is useful because it allows for a better understanding of the meaning of that tag. When the user clicks on a tag, the corresponding cloud is created in a new tab. Thanks to this feature the user can also navigate the `DBpedia` subgraph in an intuitive way. The user can collects suggested tags she considers relevant for the annotation by a drag and drop operation of the tag in her tag-bag area (indicated by *(3)* in Figure 1). Once the user selects a tag, the system automatically enriches this area with concepts related to the dropped tag. For example, in the case of *Drupal*, its most related concepts are *PHP*, *Software*, *Web Development*, *Content Management System* and so on. These new keywords represent the corresponding Wikipedia Categories showed in the Wikipedia page of *Drupal*. Also the tags appearing in the personal tag bag area are sized according to their relevance. Thanks to the `RDF` nature of `DBpedia`, they can be easily computed 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.

For a further deeper expansion of (semantic) keywords in the tag bag, we exploit the `skos:broader` and `skos:subject` properties within `DBpedia`. These two properties are used to represent an ontological taxonomy among Wikipedia categories. In particular, `skos:broader` links a category (subject) to its supercategory while `skos:subject` relates a resource to its corresponding Wikipedia category. The `SPARQL` query used to compute the expanded cloud related to a given resource is recursively repeated (offline) for all the related categories.

3 DBpediaRanker: RDF Ranking in DBpedia

In this section we will describe our hybrid ranking algorithm *DBpediaRanker*, used to rank resources (tags) in `DBpedia` w.r.t. a given keyword. For a more detailed description of the system the interested reader can refer to [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 graph browsing, and the consequent ranking of resources, is performed *offline* and, at the end, the result is a weighted graph where nodes are `DBpedia` resources and weights represent the similarity value between any pair of nodes. The graph so obtained will then be used at *runtime* (i) in the *annotation phase*, to suggest *similar* tags to users annotating e.g. their software components and (ii) in the *retrieval phase*, to display components annotated with tags semantically related to the ones used in the query.

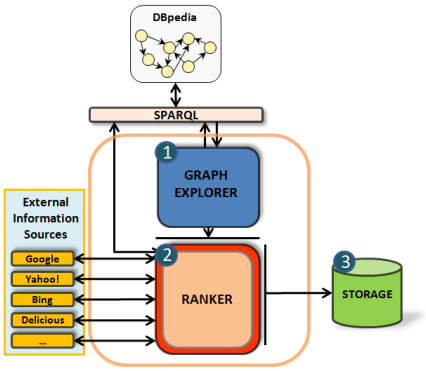

Fig. 2. The ranking system *DBpediaRanker*.

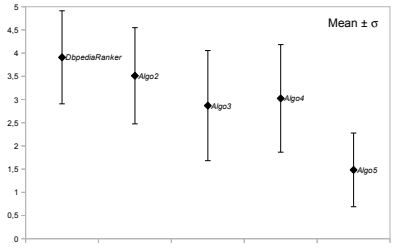

Fig. 3. Average ranks.

The system (see Figure 2) is composed by the following components:

– **Graph Explorer**: it explores the `DBpedia` graph, through `SPARQL` queries, starting from a set of initial seeds. Each node is explored within a predefined number of hops and following a predefined set of properties. Being *NOT*

centered on the *IT* domain, we were interested in exploring only a domain-specific subset of the `DBpedia` graph. For this reason the initial seeds were manually chosen to belong to that domain (i.e. to the same context). During our experiments we set the maximum explored depth equal 2, and we explored the graph following two properties belonging to the `SKOS` vocabulary, i.e., `skos:subject` and `skos:broader` (see Section 2). A deep explanation of these choices is provided in [9].

- **Context Analyzer**: it limits the exploration of the graph to a specific context. It uses the *Ranker* (see below) to determine if a resource belongs to the context. The context is represented by the most popular `DBpedia` categories, i.e., the categories that are reached more often during the exploration of the graph.
- **Ranker**: this is the core component of the system. It determines a similarity value between any pairs of nodes (uri_1 and uri_2) discovered by the *Graph Explorer*; this similarity value is the weight associated to the edge between the two resources. The score is computed by querying the external information sources with the following formula:

$$sim(uri_1, uri_2, info\_source) = \frac{p_{uri_1,uri_2}}{p_{uri_1}} + \frac{p_{uri_1,uri_2}}{p_{uri_2}} \quad (1)$$

  where p_{uri_1,uri_2} is the number of documents that contain (or have been tagged with) both the label associated to uri_1 and the one associated to uri_2, and p_{uri_1} and p_{uri_2} are the number of documents that contain (or have been tagged by) the label associated to respectively uri_1 and uri_2. Moreover, we look at, respectively, **abstracts** in Wikipedia and **wikilinks**, i.e., links between Wikipedia pages. Specifically, given two resources 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.
- **Storage**: all the similarity values between the pairs of resources, together with the "popularity" of each resource (calculated as the number of times that each node is reached in the exploration) are stored in a DBMS for an efficient retrieval at runtime.

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.

Algo2 is equivalent to our algorithm, but it does not take into account textual and link analysis in `DBpedia`. *Algo3* is equivalent to our algorithm, but it does not take into account external information sources, i.e., information coming from search engines and social bookmarking systems. *Algo4*, differently from our

algorithm, does not exploit textual and link analysis. Moreover, when it queries external information sources, instead of the formula (1), it uses the *co-occurrence* formula: $\frac{p_{uri_1,uri_2}}{p_{uri_1}+p_{uri_2}-p_{uri_1,uri_2}}$. *Algo5* is equivalent to *Algo4*, but it uses *similarity distance* [2] instead of co-occurrence. In order to assess the quality of our algorithm we conducted a study where we asked participants to rate the results returned by each algorithm. 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 evaluation system is still available at `http://sisinflab.poliba.it/evaluation`. During the evaluation phase, the volunteers were asked to rate the different ranking algorithms from 1 to 5, according to which list they deemed represent the best results for each query. The order in which the different algorithms were presented varied for each query. This has been decided to prevent users from being influenced by previous results. For the same reason the columns do not have the name of the ranking measure. The area covered by this test was the *ICT* one and in particular *programming languages* and *databases*. 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. During the testing period we collected 244 votes. In the evaluation the user could search for a keyword in the ICT domain by typing it in the text field, or she could directly select a keyword from a list below the text field that changed each time the page was refreshed. While typing the resource to be searched for, the system suggested a list of concepts obtained from `DBpedia`. Moving the mouse pointer on a cell of a column, the cells in other columns having the same label were highlighted. This allowed the user to better understand how differently algorithms rank the same resource and in which positions the same labels are in the five columns. Clicking on a concept, the corresponding Wikipedia page opened in an iframe. This facilitates the user to obtain more information about the clicked concept. Finally, the user could start to rate the results of the five algorithms, according to the following scale: (i) one star: *very poor*; (ii) two stars: *not that bad*; (iii) three stars: *average*; (iv) four stars: *good*; (v) five stars: *perfect*. The user had to rate each algorithm before sending her vote to the server. Once rated the current resource, the user could vote for a new resource if she wanted. For each voting we collected the time elapsed to rate the five algorithms: on the average it took about 1 minute and 40 seconds ($\sigma = 96.03$ s). The most voted resources were `C++`, `MySQL` and `Javascript` with 10 votes. In Figure 3 we plotted the mean of the votes assigned to each method. Error bars represent standard deviation. *DBpediaRanker* has a mean value of 3.91 ($\sigma = 1.0$). It means that, on the average, users rated it as *Good*. Examining its standard deviation, we see that the ranks are comprised between *Average* and *Perfect*. In order to determine if the differences between our method and the others are statistically significant we used the Wilcoxon test. From the Wilcoxon test we can conclude 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 (with $p < 0.0001$).

5 Related Work

Nowadays, a lot of websites expose their data as RDF documents; just to cite a few: *DBPL*, *RDF book mashup*, *DBtune*, *MusicBrainz*[5]. Therefore, it would be very useful to have some metrics able to define the relevance of nodes in the RDF graph, in order to give back to the user a *ranked* list of results, ranked w.r.t. the user's query. In order to resolve this issue several PageRank-like [11] ranking algorithms have been proposed [4, 7, 8, 6]. They seem, in principle, to be good candidates to rank resources in an RDF knowledge base. Yet, there are some considerable differences, that cannot be disregard, between ranking web documents and ranking resources to which some semantics is attached. Indeed, the only thing considered by the PageRank algorithm is the origin of the links, as all links between documents have the same relevance, they are just hyperlinks. For RDF resources this assumption is no more true: in an RDF graph there are several types of links, each one with different relevance and different semantics, therefore, differently from the previous case, an RDF graph is not just a graph, but a directed graph with labels on each edge. Moreover an RDF resource can have different origins and can be part of several different contexts and this information cannot be disregarded, instead it should be exploited in some way in the ranking process. *Swoogle* [4] is a semantic web search engine and a metadata search provider, which uses the *OntologyRank* algorithm, inspired by the PageRank algorithm. Nonetheless, we borrowed from Swoogle the idea of browsing only a predefined subset of the semantic links. Similarly to our approach also the *ReConRank* [7] algorithm explores just a specific subgraph: when a user performs a query the result is a topical subgraph, which contains all resources related to keywords specified by the user himself. In the subgraph it is possible to include only the nodes *directly* linked to the particular root node (the query) as well as specify the number n of desired hops, that is how far we want to go from the root node. The *ReConRank* algorithm uses a PageRank-like algorithm to compute the relevance of resources, called *ResourceRank*. However, like our approach, the *ReConRank* algorithm tries to take into account not only the relevance of resources, but also the "context" of a certain resource, applying the *ContextRank* algorithm [7]. Our approach differs from [7] due to the semantic richness of the DBpedia graph (in terms of number of links) the full topical graph for each resource would contain a huge number of resources. This is the reason why we only explore the links skos:subject and skos:broader. Hart et al. [6] exploit the notion of naming authority, introduced by [8], to rank data coming from different sources. In our case, we are not interested in an absolute ranking and we do not take into account naming authority because we are referring to DBpedia: the naming authority approach as considered in [6] loses its meaning in the case of a single huge source such as DBpedia. Mukherjea et al. in [10] presented a system

[5] www.informatik.uni-trier.de/~ley/db, www4.wiwiss.fu-berlin.de/bizer/bookmashup, dbtune.org, musicbrainz.org

to rank RDF resources inspired by [8]. In our approach, in order to compute if a resource is within or outside the context, we consider as *authority* URIs the most popular DBpedia categories. Based on this observation, URIs within the context can be interpreted as *hub* URIs. *TripleRank* [5], by applying a decomposition of a 3-dimensional tensor that represents an RDF graph, extends the paradigm of two-dimensional graph representation, introduced by HITS, to obtain information on the resources and predicates of the analyzed graph. In the pre-processing phase they prune dominant predicates, such as dbpprop:wikilink, which, instead, have a fundamental role as shown in the experimental evaluation. *Sindice* [12], differently from the approaches already presented, does not provide a ranking based on any lexicographic or graph-based information. It ranks resources retrieved by SPARQL queries exploiting external ranking services (as Google popularity) and information related to hostnames, relevant statements, dimension of information sources. Differently from our approach, the main task of Sindice is to return RDF triples (data) related to a given query.

Acknowledgments

We are very grateful to Gaetano Pavone for the implementation of *DBpediaRanker*. 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. R. Cilibrasi and P. Vitányi. The Google Similarity Distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383, 2007.
3. C. B. et al. Dbpedia - a crystallization point for the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, July 2009.
4. L. D. et al. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04*, pages 652–659, 2004.
5. T. Franz, A. Schultz, S. Sizov, and S. Staab. Triplerank: Ranking semantic web data by tensor decomposition. In *ISWC*, 2009.
6. A. Harth, S. Kinsella, and S. Decker. Using naming authority to rank data and ontologies for web search. In *International Semantic Web Conference*, 2009.
7. A. Hogan, A. Harth, and S. Decker. ReConRank: A Scalable Ranking Method for Semantic Web Data with Context. 2006.
8. J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.
9. R. Mirizzi, A. Ragone, T. Di Noia, and E. Di Sciascio. Ranking the linked data: the case of dbpedia. In *10th Int. Conf. on Web Engineering (ICWE)*, 2010.
10. S. Mukherjea, B. Bamba, and P. Kankar. Information Retrieval and Knowledge Discovery utilizing a BioMedical Patent Semantic Web. *IEEE Trans. Knowl. Data Eng.*, 17(8):1099–1110, 2005.
11. L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, 1998.
12. G. Tummarello, R. Delbru, and E. Oren. Sindice.com: Weaving the Open Linked Data. *The Semantic Web*, pages 552–565, 2008.