A generic approach based on Linked Data to enhance Web information retrieval and increase user satisfaction

Mazen Alsarem[1]

Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR5205, F-69621, France
{firstname.lastname@insa-lyon.fr}

ABSTRACT. *Using Linked Data for improving information retrieval is becoming an important field of research, especially with the widespread use of semantic search engines. In this paper we propose using the Linked Data for improving user satisfaction during his interactions with an IR system. We want to provide the users with more intelligent snippets. In order to achieve this goal, we focus on graph ranking after having transformed the RDF graphs into a bipartite graph representation.*

RÉSUMÉ. *De nombreux travaux s'intéressent à l'utilisation des Linked Data pour améliorer la recherche d'information (ex. moteurs de recherche sémantiques, etc.). Dans ce travail, nous proposons de les utiliser pour améliorer la satisfaction de l'utilisateur durant ses interactions avec un système de RI. Nous voulons offrir à l'utilisateur des snippets adaptés à l'expression de son besoin d'information. Pour ce faire, nous nous concentrons sur l'appariement de graphes RDF, après les avoir transformé en graphes bipartis.*

KEY WORDS: *Linked Data, search results, Web search, information retrieval, semantic web, RDF model, RDF graphs matching, snippets, bipartite graph.*

MOTS-CLÉS: *Linked Data, recherche d'information, recherche sur le Web, le Web sémantique, le modèle RDF, appariement de graphes RDF, snippets, graphe biparti.*

[1] Thesis supervisors: Sylvie Calabretto, Pierre-Edouard Portier

1. Introduction

Given a user query, Web search engines return a list of ranked results. These results are relevant to the query, but mainly from a system point of view; therefore the user has to navigate between them in order to find which one is the most relevant for him (user point of view). One of the objectives of an information retrieval (IR) system is to increase user satisfaction by simplifying this post-query exploratory stage. One way to proceed is by providing more information about the results with intelligent snippets. We use Linked Data to build graph representations of both the documents and the query. We then apply a ranking algorithm on the graphs for producing enhanced snippets. We first introduce our architecture (Section 2), then in Section 3 we present a state of the art for the domains of enhancing the presentation of search results, and RDF graphs matching. Finally we introduce a method for the ranking of RDF graphs based on the transformation of an RDF graph to a bipartite graph (Section 4).

2. Approach and Architecture

Data model generation

We transform the documents into RDF graphs by combining three technologies: (1) “named-entities recognition” with DBpedia Spotlight (Mendes et al. 2011) and DBpedia lookup (2) “semantic annotations” such as RDFa and Microformats when they exist (we use apache any23), and (3) “Linked Data keywords search” which consists in a classical search-by-keywords process on the textual information associated with an RDF node (e.g. labels, names, descriptions, ...).

In our approach, the query is generally a small set of short keywords. Therefore we start our process by a lexical extension of the query, using synonymous terms from WordNet (Fellbaum 1998). This extended query will form the input data for the RDFizing algorithm which will generate an RDF graph for the query. The generated graph remains quite small compared to the graphs of the documents, a possible approach is to extend it under an *Expansion Quality* criterion. To calculate the *Expansion Quality*, we first evaluate the representativeness of the to-be-extended resource. Given a similarity measure between the nodes of a graph, we define the representativeness of a node (i.e. a resource) as its summed similarity with the other resources of the graph. Thus, we rank the resources of a graph according to their summed pairwise similarities, and we only extend the most representative ones, by adding related resources from Linked Data.

Intelligent snippets

We generate intelligent snippets from the content of the document, by transforming the document into a RDF graph and by ranking its triples with a RDF Graph matching algorithm (Section 4.1). Then we select the top N triples and we transform them into a textual or graphical representation easily accessible for the user. The next section describes related work.

3. Related Work

Enhancing search results using Linked Data

A few existing works focus on using Linked Data to enhance search results. Haas and al (Haas et al. 2011) extend Web search results by using Linked Data to include multimedia, key-value pairs, and elements that allow the user to interact with the search results page. However, their approach is only based on semantic annotations provided by the documents' authors or extracted by specific wrappers. Bai and al (Bai, Delbru, and Tummarello 2008) propose an approach for automatically generating snippets from RDF documents. In their work they are limited to semantic search engines and RDF files, and they don't manage classical Web pages.

Snippets

There are already existing works proposing enhanced snippets; Google Rich Snippets and Yahoo Search Monkey recognize a small fixed vocabulary of structured metadata (e.g. Opinion, Products ...) that will be extracted from the documents. Then, the enriched snippet will show these semantic annotations. It should be noted that those annotations can sometimes contain spam such as fake ratings, etc... Therefore, they are strongly oriented toward commercial uses. Bai et al (Bai, Delbru, and Tummarello 2008) worked on snippet generation for a semantic search engine (viz. Sindice), but their work is specifically oriented toward semantic search engines; therefore it can only manipulate RDF results.

4. Matching RDF graphs

Our approach is based on RDF graph matching, and similarity measures for RDF graphs. Many methods have been proposed to solve this problem; *SimRank* (Jeh and Widom 2002) is a recursive similarity function based on a simple assumption viz. "*two objects are similar if they are related to similar objects*". Fogaras and Racz (Dániel Fogaras, Fogaras, and Rácz 2005) proposed another version of SimRank they called "*efficient approximate method SimRank*". On the other hand, *SSDM* (Olsson et al. 2011) is based on *SimRank* and makes use of both structural and semantic content of linked data for finding similar entities. Another important work is *LDSD* (Passant 2010) which provides a set of measures for semantic distance between Linked Data resources. We did some experiments[2] to help us choose a matching algorithm, and we found that all these algorithms are very expensive (performance) when applied to RDF graphs. Therefore we first transform the RDF

[2] More details in http://ensen-liris.blogspot.fr

graphs into bipartite graphs (using an existing technique), and we then use existing and efficient algorithms for computing a similarity measure between bipartite graphs.

4.1. Matching RDF graphs by using a bipartite graph representation

Most RDF graphs matching algorithms; *SSDM*, *LDSD*, depend on the pairwise comparison of resources, and on finding paths between them, which is very costly in terms of time complexity. In addition these algorithms are only comparing subject and object resources and don't use the predicates. This can be considered a loss of potentially useful information. Therefore we start by converting the RDF graph to a bipartite graph (Hayes and Gutierrez 2004). Then we apply a bipartite graph matching algorithm, we chose the *Relevance Search* algorithm (Sun et al. 2005), we chose it for its simplicity, speed, and scalability, and finally we extend the result of the bipartite graph matching algorithm.

4.2. Extending the algorithm

RS is designed for comparing a node with a graph, or for ordering graph's nodes according to a selected node. To apply it on graph matching, we compute the RS score for each resource from graph *A* with graph *B*, and for each resource from graph *B* with graph A, then we add the scores to get a final score. *RS* is also designed for comparing a node with a graph that contains it. Therefore, we first make an exact intersection between the two graphs. Next, we apply the *RS* algorithm for each node of $A \cap B$ with graph *A* and graph *B*. We obtain three sets of nodes (viz. the intersection set, the neighborhood from *A* set, and the neighborhood from *B* set). We consider the union of these three sets as the matching set of nodes, and the sum of the scores (i.e. the score for each node in $(A \cap B)$ with both graphs) as the matching score. We chose to take the sum because we consider that graphs similarity increases with the number of shared entities.

4.3. Algorithms

Algorithm 1: Resource-Graph matching algorithm

With this algorithm, we associate a score of relevance for all the resources in the neighbourhood of the input resource.

```
Input: URIResource R, RDF graph G
Output: Score of matching $S_c$, List of pairs [(Resource $R_j$ , Score $S_{c_j}$ )] Res
1 Split G to a set of connected graphs: $G_s \leftarrow$ split (G);
2 Find graph that contains R in $G_s$ : $G_i \leftarrow$ find(R, $G_s$);
3 Construct the Bipartite graph representation for $G_i$ : $BG_i \leftarrow$ construct_Bi ( $G_i$);
```

```
4 Initialize the input of Relevance Search algorithm: $\overrightarrow{q_a} = 0$ Except for R-th element
  $\overrightarrow{q_a}(R) = 1$ ;
5 Apply Relevance Search algorithm: $\overrightarrow{u_a} \leftarrow RS(\overrightarrow{q_a}, BG_i)$;
6 Prepare the output list: Res ← [(Resource $R_j$, Score $S_{c_j}$ )] ← Convert ($\overrightarrow{q_a}$, $G_i$);
7 Sort the results over the score: Res ← Sort (Res);
8 Calculate the score of matching: $S_c$← sum_Score (Res);
9 Return $S_c$ , Res
```

Algorithm 2: Graph-Graph matching algorithm

Here we propose a matching algorithm that computes a score of relevance for two RDF graphs. It also returns all the resources shared by the two graphs and their corresponding scores (representing their importance). We use this algorithm for matching the graph of the query with the graphs of the documents, and we use the found shared resources for generating the intelligent snippets.

```
Input: RDF graphs $G_1$, $G_2$
Output: Score of matching $S_c$, List of pairs [(Resource $R_j$, Score $S_{c_j}$)] Res
1 Intersect the two graphs: L ← intersect ($G_1$, $G_2$);
2 For each resource $L_i \in L$
3   Matching the resource $L_i$ with the graph $G_1$:
    $L_{i1}$ ← [( $R_{j1}$ , $S_{c_{j1}}$ )] ← resource_graph_matching ($L_i$, $G_1$);
4   Matching the resource Li with the graph $G_2$:
    $L_{i2}$← [( $R_{j2}$ , $S_{c_{j2}}$ )] ← resource_graph_matching ($L_i$, $G_2$);
5   Merge $L_{i1}$ and $L_{i2}$: $Res_i$ ← [( $R_j$ , $S_{c_j}$ )] ← merge_lists ($L_{i1}$ , $L_{i2}$ );
6   Add $Res_i$ to Res: Res ← merge_lists ($Res_i$ , Res);
7 Sort the results over the score: Res ← Sort (Res);
8 Calculate $s_c$: $s_c$← sum_Score (Res);
9 Return Res, $s_c$
```

Experiments[3]

We evaluated the idea of converting an RDF graph to a bipartite graph before applying the relevance search algorithm. We built a movie recommendation system which, for a given film, returns an ordered list of related films, by analyzing the RDF graphs for this film, and comparing it, using our proposed approach to the graphs of the remaining films. To build this system we used DBpedia to retrieve the movie list (500 films) and their RDF graphs. We tested the system manually for all the proposed movies, and we found that for most of the movies, the results were satisfying.

[3] More details about this experiments in http://ensen-liris.blogspot.fr/

5. Conclusion and future work

We can use Linked Data for increasing users' satisfaction and adding many enhancements to IR systems. In this paper we analyze the query and the results returned by a search engine, and then propose ways for building an enhanced results page with intelligent snippets. The novelty is to be found in the transformation of the RDF graph into its bipartite representation before applying an adapted version of a known algorithm for bipartite graph matching. We evaluated our idea by building a movie recommender system. The next step of our research will be the implementation of a prototype for all the parts of the proposed architecture. In this future system, we will use our proposed matching algorithm to build other enhanced services. We will then evaluate this prototype by means of experiments covering many kinds of queries and search engines.

6. References

Bai, Xi, Renaud Delbru, and Giovanni Tummarello. 2008. "RDF Snippets for Semantic Web Search Engines." On the Move to Meaningful Internet Systems: p 1–15.

Dániel Fogaras, Balázs Rácz, D Fogaras, and B Rácz. 2005. "Scaling Link-based Similarity Search." Proceedings of the 14th international conference on WWW: p 641–50.

Fellbaum, Christiane. 1998. 71 British Journal Of Hospital Medicine London England 2005 "WordNet: An Electronic Lexical Database". ed. Christiane Fellbaum. MIT Press.

Haas, Kevin, Peter Mika, Paul Tarjan, and Roi Blanco. 2011. "Enhanced Results for Web Search." The 34th international ACM SIGIR: p 725–34.

Hayes, Jonathan, and Claudio Gutierrez. 2004. "Bipartite Graphs as Intermediate Model for RDF." The Semantic Web–ISWC 2004: p 47–61.

Jeh, Glen, and Jennifer Widom. 2002. "SimRank: a Measure of Structural-context Similarity." The eighth ACM SIGKDD international: p 538–43.

Mendes, PN, Max Jakob, García-silva A, and Christian Bizer. 2011. "Dbpedia Spotlight: Shedding Light on the Web of Documents." I-Semantics 2011: p 1–8.

Olsson, Catherine, Plamen Petrov, Jeff Sherman, and A Perez-Lopez. 2011. "Finding and Explaining Similarities in Linked Data." Proceedings of STIDS 2011.

Passant, Alexandre. 2010. "Measuring Semantic Distance on Linking Data and Using It for Resources Recommendations." Linked AI: AAAI Spring Symposium.

Sun, Jimeng, Huiming Qu, Deepayan Chakrabarti, and Christos Faloutsos. 2005. "Relevance Search and Anomaly Detection in Bipartite Graphs." ACM SIGKDD: p 48–55.