Making Use of Linked Data for Generating Enhanced Snippets

Mazen Alsarem[1], Pierre-Édouard Portier[1], Sylvie Calabretto[1], and Harald Kosch[2]

Université de Lyon, CNRS
[1]INSA-Lyon, LIRIS, UMR5205, F-69621, France
[2]Universität Passau, Innstr. 43, 94032 Passau, Germany
{firstname.lastname}@insa-lyon.fr
{firstname.lastname}@uni-passau.de

Abstract. We enhance an existing search engine's snippet (i.e. excerpt from a web page determined at query-time in order to efficiently express how the web page may be relevant to the query) with linked data (LD) in order to highlight non trivial relationships between the information need of the user and LD resources related to the result page. To do this, we introduce a multi-step unsupervised co-clustering algorithm so as to use the textual data associated with the resources for discovering additional relationships. Next, we use a 3-way tensor to mix these new relationships with the ones available from the LD graph. Then, we apply a first PARAFAC tensor decomposition [5] in order to (i) select the most promising nodes for a 1-hop extension, and (ii) build the enhanced snippet. A video demonstration is available online[1].

Keywords: Linked Data, Information Retrieval, Snippets, Co-Clustering, Tensor Decomposition

1 Introduction

In this work, we show that the LOD (Linking Open Data) graph can be combined with analysis of the original textual content of a search engine's results to create efficient enhanced snippets. Contrary to other approaches (such as [4]), we don't rely exclusively on explicit semantic annotations (e.g. RDFa, ...) but we are able to enhance any result even if the corresponding web page doesn't contain annotations.

The paper is organized as follows: we start with a brief state of the art for the domains of snippets' enhancement, and snippet generation for corpus of RDF documents (Section 2). Then we describe our approach (Section 3) before concluding (Section 4).

[1] http://blog.ensen-insa.org

2 Related Works

Many approaches tried to enhance the Search Engine Result Page (SERP) snippets, but none of them seem to have used the LoD graph. First, Haas *et al.* [4] employed structured metadata to enhance the SERP with multimedia elements, key-value pairs and interactive features. They chose not to use the LoD graph to avoid the problem of the transfer of trust between the Web of documents and the Web of Data. Also Google Rich Snippet (GRS) [10] is a similar initiative that relies exclusively on structured metadata authored by the web pages' publishers.

We should also mention the works of Ge *et al.* [3], and Penin *et al.* [9] focused on the generation of snippets for ontology search, and also the work of Bai *et al.* [1] about the generation of snippets for native RDF documents.

In summation, we agree with Ge *et al.* [3] that the main benefit of possessing highly structured data from an RDF graph is the possibility to find non-trivial relations among the query terms themselves, and also between the query terms and the main concepts of the document. Moreover, we agree with Penin *et al.* [9] and Bai *et al.* [1] about the necessity to design a ranking algorithm for RDF statements that considers both the structure of the RDF graph and lexical properties of the textual data. However, we find ourselves in an inverted situation with genuine textual data from classical web pages, and RDF graphs generated from these web pages by using the LoD graph.

3 Proposal

Our main purpose is to highlight on a practical and convincing use case the benefits of a conjoint use of the web of documents and the web of data. Thus, for each result of the SERP, we want to build a RDF graph and combine this graph with a textual analysis of the document in order to obtain features from which to build an enhanced snippet. Meanwhile, our main concern is to limit the amount of noise introduced by this process.

In a first step, we use DBpedia Spotlight [6] to extract LoD resources from the content of the SERP result. Next, we use a SPARQL endpoint connected to the DBpedia dataset to introduce RDF statements between the resources we just found. At that time we have a first graph associated to the SERP result.

3.1 Multi-step Unsupervised Co-clustering

We need to build a more informed graph in order to efficiently select, in a subsequent step, a relevant subset of nodes to extend. Thus, our main objective is to highlight relevant relations between the resources based on an analysis of the textual data associated to the resources. Therefore, to discover more diverse relationships than the ones obtained from a naive lexical approach (e.g. cosine distance between the abstracts...), we introduce a new multi-step unsupervised co-clustering algorithm. At the core of our algorithm, lies a classical co-clustering algorithm based on SVD [7] that we adapted for the X-means [8] algorithm so as to avoid having to estimate the number of clusters. We use a co-clustering approach because the relation between resources of a given cluster will be qualified by the terms present in this cluster. First, we remove the stop-words from the texts associated to each node of the graph and we build a resource-term matrix on which we apply a co-clustering.

However, the first time the co-clustering algorithm is executed, the internal quality of the found clusters is very poor — with a mean silhouette index close to zero. Our approach is to decrease the size of the features' space (i.e. the number of terms and resources) until we find clusters of a good enough quality. When we find clusters of good internal quality and containing only terms we remove those terms and repeat the co-clustering. The underlying rationale for this strategy is that the terms that lie well in their own clusters but do not associate with resources will not help to discover and explain relationships between resources. However, we observed that this dimension reduction mechanism is insufficient since there can be clusterings with a poor average silhouette and with no clusters only made of terms. Therefore we also remove from the features' space the terms and the resources of the small clusters with a weak silhouette. Furthermore, when we can find no clusters only made of terms, no small clusters with a low silhouette, and when the average silhouette remains low, we heuristically reapply this recursive algorithm on the clusters with a poor silhouette. A more detailed presentation of the algorithm is available online in the form of a technical report[2].

3.2 Graph Extension by Tensor Decomposition

We would not benefit much from the LoD if we didn't extend the graph in order to find facts not present in the original document. However, we must be careful not to introduce noise during the extension. To do this, we propose to guide this process with the knowledge obtained from the previous multi-step co-clustering. Therefore, we have to combine the connectivity information of the graph with the labeled clusters. We propose to represent these two kinds of information in one structure: a 3-way tensor (denoted $\mathcal{T}$).

As in [2], the three modes of the tensor are associated respectively to the subject, the object and the predicate of the triples that constitute the graph. Thus, for each predicate, there is one horizontal slice that represents the adjacency matrix of the subgraph only made of the triples that link resources thanks to that predicate. We propose to add a horizontal slice for each cluster returned by the multi-step co-clustering algorithm. Such a slice represents the clique of all the resources present in the cluster. To each edge of the clique, we assign a weight linearly proportional to the silhouette index of the cluster. This coefficient of proportionality is chosen so as to make all the slices of the tensor comparable.

In order to select the nodes to extend, we start by applying a PARAFAC [5] tensor decomposition algorithm to the tensor. This decomposition computes a

[2] `http://blog.ensen-insa.org`

representation of the tensor as a sum of rank-one tensor (a rank-one three-way tensor is the outer product of three vectors), i.e.,

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{s}_r \circ \mathbf{o}_r \circ \mathbf{p}_r \,. \tag{1}$$

If there are n_r resources and n_p predicates, the lengths of each $\mathbf{s}_r$, $\mathbf{o}_r$ and $\mathbf{p}_r$ vectors are respectively n_r, n_r and n_p. The components of the vectors $\mathbf{s}_r$ and $\mathbf{o}_r$ represent, for the number r factor, the importance of each resource as it plays respectively the part of subject and object in relations involving the high-scored predicates of $\mathbf{p}_r$. Our strategy is to select for each factor r, with at least one high-scored predicate found in $\mathbf{p}_r$, the high scored resources of $\mathbf{s}_r$ and $\mathbf{o}_r$ as the nodes to extend (i.e. we query a DBpedia SPARQL endpoint to find new triples for which these nodes are subject).

3.3 Snippet Construction

We apply a new tensor decomposition on the extended graph and we select only the factors for which there are values in $\mathbf{p}_r$ larger than an experimentally fixed threshold (in our case 0.1). For those factors, we then select the resources of $\mathbf{s}_r$ and $\mathbf{o}_r$ with a weight greater than this same threshold. Thus, for each factor that describes an important sub-graph (i.e. for which there exist high-scored predicates) we isolate the main-resources that act either as hubs or authorities.

To each main-resource we associate a set of explanatory triples and we select a sentence of the SERP's result for its capacity to contextualize the main-resource. The triples associated to a main-resource are trivially derived from the factors where this resource appears either as an important subject or as an important object. The sentence associated to a main-resource is found by ranking the sentences of the document according to seven factors: (1) the presence of a DBpedia Spotlight annotation for the main-resource, (2) the presence of a DBpedia Spotlight annotation for a resource that appears with a high score in a group where the main-resource also has an high score, (3) the presence in the text of the local name of a predicate that belongs to one of the main-resource's groups, (4) the presence of query's terms, (5) the presence in the text of the local name of the resource, (6) the presence of a DBpedia Spotlight annotation for a resource that appears in the query, and (7) the presence of a DBpedia Spotlight annotation for a resource adjacent to the main-resource in the graph.

Finally, we apply a similar ranking strategy to find a sentence that is both close to the query and a place of co-occurrence for as many main-resources as possible. This sentence is shown in addition to the original excerpt chosen by the search engine.

4 Conclusion

We proposed a way of enhancing a traditional snippet with structured data coming from the LoD. Our intention was firstly to identify relationships between the main concepts of a document relevant to the user's information need, and secondly to link these concepts with either relevant concepts not present in the document, or facts about them. Our main challenge was to succeed in this task while introducing as little noise as possible. In this respect, we identified the keystone as the graph extension step for which more relational information was needed in order to efficiently choose the resources that would benefit most from an extension. Thus, we proposed a multi-step unsupervised co-clustering algorithm in order to discover additional relationships between the resources by taking into account textual data associated to them. Next, we represented both the structures coming from the LoD and those added by our co-clustering process with a 3-way tensor before running a tensor decomposition so as to identify the main resources we should extend. Our proposal has been implemented — a more detailed version of this work, screenshots and a demonstration are available online[3][4] — and we are now in the process of thoroughly evaluating the end-user satisfaction.

References

1. Bai, X., Delbru, R., Tummarello, G.: Rdf snippets for semantic web search engines. In: On the Move to Meaningful Internet Systems: OTM 2008, pp. 1304–1318. Springer (2008)
2. Franz, T., Schultz, A., Sizov, S., Staab, S.: Triplerank: Ranking semantic web data by tensor decomposition. In: The Semantic Web-ISWC 2009, pp. 213–228. Springer (2009)
3. Ge, W., Cheng, G., Li, H., Qu, Y.: Incorporating compactness to generate term-association view snippets for ontology search. Information Processing & Management pp. 513–528 (2012)
4. Haas, K., Mika, P., Tarjan, P., Blanco, R.: Enhanced results for web search. In: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. pp. 725–734. ACM (2011)
5. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review 51(3), 455–500 (2009)
6. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: Dbpedia spotlight: shedding light on the web of documents. In: Proceedings of the 7th International Conference on Semantic Systems. pp. 1–8. I-Semantics '11, ACM (2011)
7. Park, L.A., Leckie, C.A., Ramamohanarao, K., Bezdek, J.C.: Adapting spectral co-clustering to documents and terms using latent semantic analysis. In: AI 2009: Advances in Artificial Intelligence. pp. 301–311. Springer (2009)
8. Pelleg, D., Moore, A.W., et al.: X-means: Extending k-means with efficient estimation of the number of clusters. In: ICML. pp. 727–734 (2000)
9. Penin, T., Wang, H., Tran, T., Yu, Y.: Snippet generation for semantic web search engines. In: The Semantic Web, pp. 493–507. Springer (2008)
10. Steiner, T., Troncy, R., Hausenblas, M.: How google is using linked data today and vision for tomorrow. Proceedings of Linked Data in the Future Internet 700 (2010)

[3] http://demo.ensen-insa.org

[4] http://blog.ensen-insa.org