# A Semantic Web based solution for an autocompletion system

Davide Palmisano
Asemantics s.r.l.
Circ.le Trionfale, 27 00100 Roma, Italia
davide@asemantics.com

Michele Minno
Asemantics s.r.l.
Circ.le Trionfale, 27 00100 Roma, Italia
michele.minno@asemantics.com

Dan Brickley
VU University
De Boelelaan 1081a, 1081 Amsterdam,
The Netherlands
danbri@danbri.org

Michele Mostarda
Asemantics s.r.l.
Circ.le Trionfale, 27 00100 Roma, Italia
michele@asemantics.com

## Abstract

*The following paper describes a SKOS-based approach to index large ontologies in order to overcome such scalability and responsiveness issues that hinder an efficient and general purpose autocompletion system. Differently from such autocompletion systems in which the underlying vocabularies are limited, flat and easily indexable with the classical text indexing techniques, issues addressed in this paper deal mainly with large, highly unpredictable, linked data spaces as the one inspired by the Linked Data philosophy.*

## 1. Introduction

The term *autocompletion*, often used in the field of the user interfaces and, more generally, in the human-machine interaction technologies, refers to the capacity of the system to predict what the user is currently typing in order to complete the string automatically. Some autocompletion systems found an important application in several programming IDE tools and in some enhanced mobile softwares, where the user capabilities are limited by non QWERTY-standard keyboards [2].

Common application fields are characterized by underlying well-defined and limited vocabularies, where the words in the lexicon have different leading characters, constraints under which the autocompletion prediction is more feasible [3]. Actually autocompletion could find an interesting application when, on the contrary, the underlying vocabulary is rich, highly unpredictable and heterogeneous as the one represented by the Linking Open Data clouds [1]. In fact, apart from those obvious benefits that a generic user could get from such kind of systems, as the reduced amount of misspelling and errors, the autocompletion offers a feasible way to improve terms disambiguation in order to enable the user to identify in an exact manner the concept he/she wants to refer to [5].

Classical autocompletion systems, like the well-known *Google Suggest* that shows to the user a list of possible search keywords ordered by their occurrence value, are traditionally based on the idea of matching input strings with a list of usable words in a vocabulary. Instead each system aiming to the terms disambiguation needs to be based on an underlying ontology, in order not only to correctly complete the partial text written by the user with the rest of the string, but to match it with a concept of that ontology. To achieve this, the possible autocompletion choices presented to the user need to be categorized according to the concept they are instance of, or another mechanism like the one offered by SKOS.

In this paper an approach based on a SKOS-indexed snapshot of an RDF ontology is presented. For prototyping purposes the DBpedia RDF ontology is used, more specifically a relational view of it, in order to reduce the complexity of the real-time computation of the autocompletion choices presented to the user. This section continues with a subsection describing the formalization of the semantic autocompletion problem and ends with a quick overview about other efforts already present in literature. Section 2 presents the main aspects of our solution, jointly with the algorithms used. Conclusions and acknowledge close the paper.

### 1.1 The Semantic Web autocompletion: problem statement

One of the best attempts to formally define the semantic autocompletion problem is provided in [5], where the authors devise a real-time mechanism that prunes the graph, made of all the facets of a certain node, according to the potential matching of the label with the user typed string. Even if the authors assure the applicability of their solution providing three different deployment scenarios, it's our opinion that this solution, not providing as far as we can see an indexing phase on the target ontology, could be unsuitable from a scalability point of view when dealing with large data spaces.

Hildebrand et al. [6], provide a serious investigation of an autocompletion system which is also deployed in real scenarios. Their SKOS adoption, since demonstrate its suitability, deeply influenced the present work.

The rest of this section provides a formal statement of the problem as we modeled it. This formalization could be helpful to understand the algorithm we devised to produce the relational snapshot of the whole DBpedia dataspace, as described in the following section.

Informally, the problem of achieving an autocompletion service based on the linked data paradigm could be expressed as follow:

*given a string s, retrieve all the instances with a rdfs:label property value that start with the string s grouped by the most representative SKOS subjects*

where the key issue is represented by the meaning of the *"most representative SKOS subject"*. Since every node could be associated to several different SKOS subjects, as depicted in Figure 1, the good behavior of the autocompletion service is strictly influenced by the SKOS subject chosen to represent every resource.

Actually, as showed in Figure 1, the resource with the *rdfs:label "Kevin Spacey"* is associated with several different SKOS categories, each of them related to a different aspect of the *"Kevin Spacey"* resource. From the autocompletion point of view, the fact that the resource *"Kevin Spacey"* is presented to the user under the category *"American Actors"* is much more convenient than if it is presented as an *"American expatriates in the United Kingdom"* in order to enable the user to provide an effective disambiguation.

Choosing the narrower category a resource has or the most consistent one are solutions that, even if characterized by a straightforward computational complexity, often lead to undesirable results. Actually such kind of solutions don't take into consideration how the resource is linked with the rest of the data space, loosing the opportunity to discover new relationships and links between different instances and concept. In one word, they loose the opportunity offered by such linked knowledge.

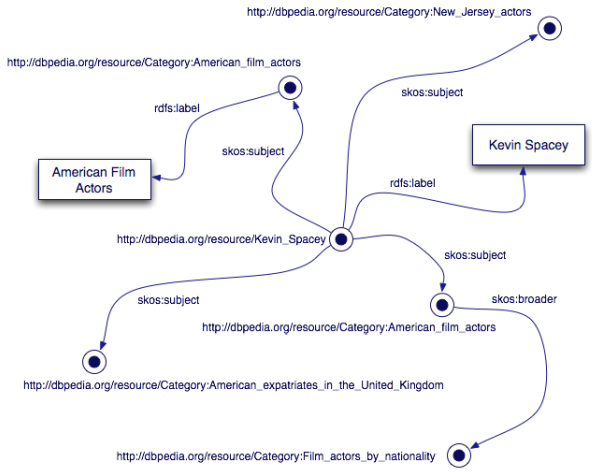


**Figure 1. Kevin Spacey example**

## 2 The relational snapshot solution

What we call here the relational snapshot is essentially an index built on an RDF ontology, where each tuple of the relation refers to a resource $R$ in the ontology and contains the following fields:

(URI of $R$, $rdfs:label$ of $R$, a $skos:subject$ $S$ of $R$, $rdfs:label$ of $S$)

where the $skos:subject$ $S$ plays the role of the category coupled with the instance $R$ presented to the user during the autocompletion phase.

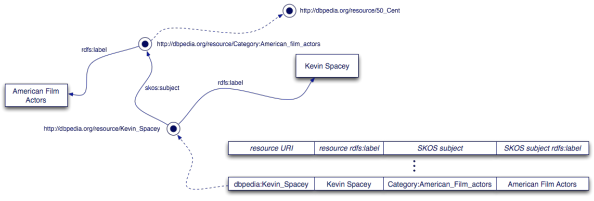


**Figure 2. A relational snapshot**

The adoption of a solution based on an index natively stored in a relational model is motivated mainly by the following two considerations:

- **Responsiveness of the adopted solution**: as an autocompletion system found in a Web based environment its natural usage and since the service could be accessed only remotely, the responsiveness raises serious issues around the scalability of the adopted solution. It comes obvious that a 30-years old consolidated technology still is the most appropriate one to address such scalability problems that come up when dealing with substring matching over a large data sets instead of using directly the native SPARQL substring matching constructs[4].
- **Transparency**: The relational view allows the end users of the autocompletion service to access the indexed ontology without regarding how it is stored.
- **Index adaptability**: Every changes in the ontology subjected to the indexing should reflect on the index itself. The adopted solution allows the modification of the relational view simply accessing to the stored tuples with SQL. RDF resources cancellation or new insertion could be resolved trivially by one single SQL delete or update on the index.

## 2.1 A SKOS based indexing

Even if the identification of the most representative SKOS subject of a certain node seems too much related to the perception that an user could have regarding a node, several positive assumptions could be done around the concept of pertinence of a resource to one SKOS subject measured as a degree of similarity between two resource.

More precisely,

- given two resources the degree of similarity is the number of SKOS subject that they have in common and,
- the most linked resource of a give skos subject is that resource with the most number of links from other resources in the dataspace.

With this two roughly defined properties is possible to state that the most representative SKOS subject of a given resource is the one that has the highest similarity degree between its most linked resource and the initial resource to be categorized. An attempt to formalize the algorithm that computes the most representative SKOS subject of a given resource is provided in the rest of this section, using the SPARQL syntax when needed.

## 2.2 The Most representative SKOS subject identification algorithm

Hereby follows the formal description of the algorithms that are the foundation of our indexing techniques. Even if a formal analysis of the computational complexity is not provided, this formalization is necessary to make this dissertation more readable and concrete. However, the computation complexity of the algorithm strongly depends on the underlying algorithm used to resolve the SPARQL queries on the targeted ontology.

**Input**: a resource URI r
**Output**: a SKOS subject URI s
$subjects \leftarrow SkosSubjects(r)$;
integer $similarity \leftarrow 0$;
SKOS subject URI $s \leftarrow subjects[0]$;
**foreach** $subject \in subjects$ **do**
  $mostLinked \leftarrow MostLinkedResource(subject)$;
  $actualSimilarity \leftarrow ResourceSimilarity(r, mostLinked)$;
  **if** $similarity \leq actualSimilarity$ **then**
    $similarity \leftarrow actualSimilarity$;
    $s \leftarrow subject$;
  **end**
**end**
**return** *s*

**Algorithm 1**: The most representative SKOS subject identification algorithm

**Input**: a resource URI r
**Output**: a set of SKOS subject URIs subjects
$subjects \leftarrow ExecuteSPARQLQuery$("SELECT DISTINCT ?uri WHERE ?r skos:subject ?uri ")
**return** *subjects*

**Algorithm 2**: The SPARQL query to retrieve the SKOS subjects of a resource

**Input**: a skos subject URI subject
**Output**: a resource URI resource
$subjectResources \leftarrow ExecuteSPARQLQuery$("SELECT DISTINCT ?uri WHERE ?uri skos:subject ?s");
$mostLinked \leftarrow subjectResources.first()$;
$linkedSize \leftarrow GetLinkedResources(mostLinked).size()$;
**foreach** $resource \in subjectResources$ **do**
  $actualLinkedSize \leftarrow GetLinkedResources(resource).size()$;
  **if** $linkedSize \leq actualLinkedSize$ **then**
    $linkedSize \leftarrow actualLinkedSize$;
    $mostLinked \leftarrow resource$;
  **end**
**end**
**return** *mostLinked*

**Algorithm 3**: The function that retrieves the most linked resource of a given SKOS subject

**Input**: a resource URI resource
**Output**: a set of resource URIs resources
$resources \leftarrow ExecuteSPARQLQuery$("SELECT DISTINCT ?uri WHERE ?uri ?prop ?r")
**return** *resources*

**Algorithm 4**: The SPARQL query that retrieves all the instances linked to a given resource

**Input**: a resource URI uri1, a resource URI uri2
**Output**: an integer N ( $0 \leq N \leq SkosSubjects(uri1))$
$similarity \leftarrow 0$
**foreach** $subject \in SkosSubjects(uri1)$ **do**
  **if** $subject \in SkosSubjects(uri2)$ **then**
    $similarity++$
  **end**
**end**
**return** *similarity*

**Algorithm 5**: The function that measures the degree of similarity between two instances

Using the defined MRSS algorithm is therefore possible to couple each resource URI to its *rdf:label* property value and its MRSS URI. This allows to achieve the autocompletion process, resolving each substring matching, as the following SQL query, where a $< substring >$ is matched with the label of an instance coupled with its MRSS:

*SELECT label, category FROM index WHERE label REGEXP ("*$< substring > \%$*") GROUP BY category*

## 3 Conclusion

The paper presented a work come out from the need of enabling the users of a profiling system to uniquely and explicitly identify instances of an ontology that represents concepts of interest. In this sense the Linking Open Data URI-based approach reveals all its potential, on condition that a suitable user interface takes into account all the issues related to the term disambiguation. Even if a lot of emphasis was given to the relational structure of the proposed indexing solution, alternative approaches, based on classical text retrieval techniques applied to SKOS, are under investigation, jointly with a concrete application of the currently adopted approach on the DBpedia dataspace.

## 4 Acknowledge

This work has been partially founded within the NoTube Project (EU FP7 Integrated Project 231761), during research activities focused on the Semantic Web technologies application in the field of user modeling.

## References

[1] T. Berners-Lee. Linked data. http://www.w3.org/DesignIssues/LinkedData.html, Juli 2006. Stand 12.5.2009.

[2] J. Hasselgren, E. Montnemery, P. Nugues, and M. Svensson. Hms: A predictive text entry method using bigrams. In *Workshop on Language Modeling for Text Entry Methods, 10th conference of the European Chapter of the Association of Computational Linguistics*, pages 43–49, 2003.

[3] E. Hyvnen and E. MŁkelŁ. Semantic autocompletion. In R. Mizoguchi, Z. Shi, and F. Giunchiglia, editors, *ASWC*, volume 4185 of *Lecture Notes in Computer Science*, pages 739–751. Springer, 2006.

[4] J. Prez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. In *International Semantic Web Conference*, pages 30–43, 2006.

[5] R. SinkkilŁ, E. MŁkelŁ, E. Hyvnen, and T. Kauppinen. Combining context navigation with semantic autocompletion to solve problems in concept selection. In K. Belhajjame, M. d'Aquin, P. Haase, and P. Missier, editors, *SeMMA*, volume 346 of *CEUR Workshop Proceedings*, pages 61–68. CEUR-WS.org, 2008.

[6] J. Wielemaker, M. Hildebrand, J. Ossenbruggen, and G. Schreiber. Thesaurus-based search in large heterogeneous collections. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 695–708, Berlin, Heidelberg, 2008. Springer-Verlag.