Sheet2RDF: A Flexible and Dynamic Spreadsheet Import&Lifting Framework for RDF

Manuel Fiorelli, Tiziano Lorenzetti, Maria Teresa Pazienza, Armando Stellato(✉), and Andrea Turbati

ART Research Group, University of Rome,
Tor Vergata, Via del Politecnico, 1 00133 Rome, Italy
{fiorelli,pazienza,stellato,turbati}@info.uniroma2.it,
tiziano.lorenzetti@gmail.com

Abstract. In this paper, we introduce Sheet2RDF, a platform for the acquisition and transformation of spreadsheets into RDF datasets. Based on Apache UIMA and CODA, two wider-scoped frameworks respectively aimed at knowledge acquisition from unstructured information and RDF triplification, Sheet2RDF narrows down their capabilities in order to restrict the domain of acquisition to spreadsheets, thus taking into consideration their peculiarities and providing informed solutions facilitating the transformation process, while still exploiting their full potentialities. Sheet2RDF comes also bundled in the form of a plugin for two RDF management platforms: Semantic Turkey and VocBench. The integration with such platforms enhances the level of automatism in the process, thanks to a human-computer interface that can exploit suggestions by users and translate them into proper transformation rules. In addition, it strengthens this interaction by direct contact with the data/vocabularies edited in the platform.

Keywords: Human-computer interaction · Ontology engineering · Ontology population · Text analytics · UIMA

1 Introduction

As organizations and public institutions are exposing their data under permissive and open licenses, they often resolve to spreadsheets and other tabular formats to make data available. The motivations rely on the acquaintance of publishers with spreadsheet editors, and the easiness of producing and reading tabular data.

There is, however, a need for methodologies and systems that support publishers in lifting these tabular data to a semantically clear form, as it is enabled by representation languages and practices of the Semantic Web [1]. To satisfy this need, we propose Sheet2RDF, an integrated system that supports a streamlined process for the transformation and lifting of spreadsheets to RDF.

The paper is structured as follows. Section 2 surveys related systems and motivates our work. Section 3 describes the approach underlying Sheet2RDF combining a powerful transformation specification language with the automatic generation of skeletal

M. Ali et al. (Eds.): IEA/AIE 2015, LNAI 9101, pp. 131–140, 2015.
DOI: 10.1007/978-3-319-19066-2_13

specifications based on the recognition of known modeling patterns in the data. Section 4 presents the system architecture. Section 5 reports on some use cases of Sheet2RDF and the feedback of its users. In Section 6, we draw the conclusions.

2 Related Works

In this section, we briefly describe some of the systems and approaches for the triplication of spreadsheets, underlining their specificities. The systems may differ in their support to different data formats: e.g. Microsoft Office, Open Office, CSV, TSV. Obviously, spreadsheet editors provide a more convenient environment for manual editing, while plain-text serializations based on predefined delimiters often enable machine-to-machine data interchange. Some systems exploit additional metadata usually not available in plain-text serializations, thus requiring at least a further preprocessing step.

Any23 (`https://any23.apache.org/dev-csv-extractor.html`) implements a systematic transformation of CSV (and otherwise delimited value formats) into RDF. The transformation strategy assumes that the first row is a header, providing the properties, while subsequent rows describe individual resources, by providing in each cell the value for the property associated with the corresponding column. Any23 has limited choices for the representation of property values, either as URIs or as literals the type of which is inferred from the values themselves.

Systematic strategies for the conversion of tabular data produce a sort of raw RDF, which usually necessitate further processing to meet specific modeling patterns and improve their quality. The DataLift platform [2] uses this two-step approach, which initially turns various data formats (not only spreadsheets) into RDF, and then leverages SPARQL [3] Constructs to refactor the RDF data originally obtained.

Tarql (`https://github.com/cygri/tarql`) somehow conflates these two steps, by allowing the evaluation of SPARQL queries directly on CSV data. Tarql is a command line program, which provides no specific assistance for writing the SPARQL queries specifying the mapping.

The system csv2rdf4lod-automation [4] follows an iterative workflow, which begins with a raw transformation of CSV into RDF, and then proceeds with successive refinements, which are expressed in RDF through a dedicated conversion vocabulary. Available enhancements include: resource typing, adoption of user-supplied vocabularies, reification of property values, custom mapping of values to URIs, and the generation of more than one resource from each row. This system also lacks a dedicated user interface, especially to support the writing of mappings.

Spread2RDF (`https://github.com/marcelotto/spread2rdf`) uses an internal Ruby-based DSL to specify the transformation. This approach may be advantageous in terms of expressivity, while at same time forcing users to adopt a new syntax, sometimes really different from the ones they are acquainted with. Conversely, Tarql uses SPARQL, a standard and popular query language among Semantic Web practitioners, likely to be interested in this type of systems.

RDF123 [5] provides an application for writing mapping specifications, as well as a web service for executing them against spreadsheets. It moves away from the assumption that rows represent homogenous resources. Hence, it uses more complex mappings, which may possibly be conditioned to specific data contained in the rows.

TabLinker (https://github.com/Data2Semantics/TabLinker) does not commit on the existence of a predetermined structure in the spreadsheet, while leveraging annotations in the spreadsheet to properly interpret the data organization. It is specifically tailored to produce data conforming to the RDF Data Cube [6] vocabulary. TabLinker uses Open Annotation Core Data Model [7] and PROV [8] to add annotations and provenance information to the cells.

Spreadsheets and tabular data in general may be considered a somewhat limited form of relational data. As such, it is possible to leverage existing RDB to RDF converters. Sparqlify-CSV (http://aksw.org/Projects/Sparqlify.html), as an example, allows to load CSV files into Sparqlify, which is a middleware for viewing relational data as virtual RDF graphs.

The limitation of a completely automated conversion motivated the development of a system [9], which uses Semantic MediaWiki to crowdsource the generation of a mapping specification for Sparqlify-CSV.

LODRefine (https://github.com/sparkica/LODRefine) is a distribution of OpenRefine that includes several extensions related to Linked Open Data. Specifically, it complements the data cleaning features of OpenRefine with the ability to link values to entities from remote datasets. Moreover, from our viewpoint, it provides a graphical language to represent mappings from tabular data to RDF conforming to user provided vocabularies. The system seems not to have any mechanism for automatic generation of the mappings.

LODRefine is a standalone system to clean data and transform it to RDF. It is also possible to find import and transformation capabilities directly in some systems concerning with the actual management and development of RDF data. The thesaurus editor PoolParty is able to import Excel files (https://grips.semantic-web.at/display/POOLDOKU/Import+Excel+Taxonomies), which adhere to the one resource per row convention. In addition, PoolParty organizes spreadsheets, trying to visualize the conceptual taxonomy through the insertion of empty cells in the rows.

TopBraid Composer is an RDF development environment, which also offers a generic facility to import both conceptual and factual knowledge from a CSV file (http://www.topquadrant.com/composer/videos/tutorials/spreadsheets/import.html). A wizard-based user interface allows the customization of the import, by mapping to an existing ontology, or by providing additional information about the conceptual content, e.g. the range of the properties inferred by the column names.

The preceding review shows that the combination of expressiveness and convenience is quite uncommon among existing systems. If the system disburdens the user from heavy configuration, in many cases the reason is that the system implements a rigid transformation based on strict assumptions about the input data. Conversely, more capable systems rely on mapping specifications, which need to be completely specified by the user, often without the support of a dedicate user interface.

Sheet2RDF aims to fill this aspect, by combining a rich mapping specification language with an intelligent approach for the semi-automatic generation of skeletal mappings. Sheet2RDF follows the convention-over-configuration principle, when it tries to relieve the user from writing the mappings at hand, by automatically generating suitable transformations based on the recognition of known modeling patterns. Further user input supports the refinement of the generated mapping, possibly informed by an already existing RDF dataset. However, the user may always resort to the underlying mapping specification language, if the spreadsheet requires a very specific transformation.

3 The Approach

The conversion of a spreadsheet to RDF may be considered an instance of the more general task of triplificating unstructured and semi-structured information. CODA [10,11] is one of the various systems supporting the achievement of this goal.

While such a system may be used as it is, we leverage the verticality of the domain (spreadsheets) to build a more focused layer on top of CODA. Fig. 1 represents the overall approach that Sheet2RDF implements to construct such an additional layer.

At the lower layer, CODA uses (in fact, it extends) UIMA [12] in order to analyze the input spreadsheet and, then, it executes a PEARL [13] document prescribing the transformation. The support of Sheet2RDF to deal with this setting is manifold. By first, Sheet2RDF defines a meta-type system that is then instantiated into a concrete

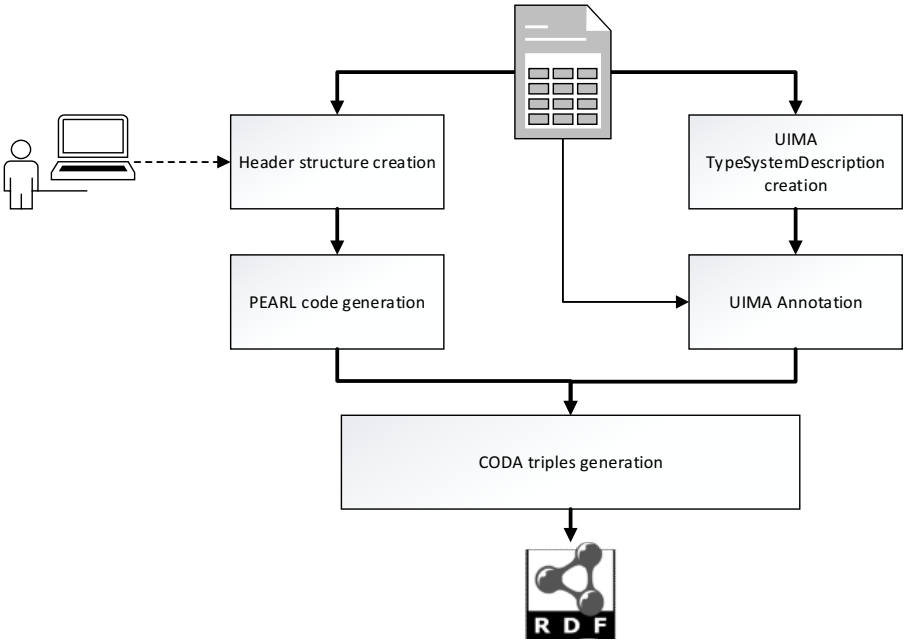

Fig. 1. Sheet2RDF approach

one for each spreadsheet. This type system models the data that will be read from the spreadsheet through a UIMA annotator provided by Sheet2RDF. At the same time, a mutable data structure mirroring the instantiated type system enables further customization from the user. In fact, user input requires a dedicated user interface, possibly providing visualizations based on already existing knowledge in the target semantic repository. Sheet2RDF uses this data structure, possibly adjusted by the user, to generate a skeleton of the PEARL transformation that lifts the spreadsheet to RDF. The user may refine this skeleton, until the PEARL document exactly matches the intended transformation. The generation of the skeleton is informed by a number of conventions that are looked for inside the data structure that reflects the spreadsheet header. A number of heuristics have been designed to match these conventions and refine the transformation rules. Ideally, the system should be able to generate a complete specification for a spreadsheet fully conforming to the foreseen conventions.

The use of an expressive transformation language guarantees the applicability to real-world scenarios, while at the same time facilities that automate the generation of the transformation specifications disburden the user from writing these specifications, as much as possible given the adherence of the input to specific conventions.

We have already developed a user interface for the system as extensions for the knowledge acquisition and management platform Semantic Turkey [14]. A user interface for the Collaborative Thesaurus Editor VocBench [15,16] is under development.

4 Sheet2RDF Architecture

Sheet2RDF consists of a library, a command-line utility and an extension for Semantic Turkey (and soon also for VocBench).

The library implements the core functionalities of Sheet2RDF, which include:

1. Generation of a skeletal PEARL transformation specification,
2. Refinement of the specification based on user input,
3. Use of CODA to execute the transformation that produces new RDF triples.

The Sheet2RDF library uses UIMA to analyze the spreadsheet (determining its header). It then produces a UIMA type system including a feature structure type that represents the rows and that has different features corresponding to each column name. Repeated columns are associated with a single feature, the value type of which is an array of elements. This type system is used by a further UIMA analyzer to read the actual content of the spreadsheet; consequently, it is also used as basis to write the corresponding PEARL transformation specifications.

As already said, the system also generates a skeleton of the transformation, by applying a set of heuristics to the aforementioned type system. The assumption is that roughly each row corresponds to an entity description, and that each triple in this description can be built from each column of the spreadsheet. The heuristics consider the entity corresponding to the row as the subject, the predicate as something inferable from (or explicitly associated to) the header of the column, and the object as (an RDF node obtained from) the value in the cell at the crossing between the row and the

column. In some particular cases, the recognized property could fire the automatic generation of more complex triple patterns in the target transformation specification. As an example, SKOS-XL [17] labelling properties also require the reification of the label into a URI.

While the heuristics are meant to generate an almost complete specification, the system also assumes that the user may provide further input that helps to refine the generated transformation. Therefore, an additional data structure has been introduced to hold this additional information provided by the user. In the end, the transformation rules are produced by merging the information provided by the type system and the additional supporting data structure. The automatically generated transformation rules may be handcrafted to match the desired output, whenever the heuristics fail to produce a complete specification. However, a better approach is to allow the user filling missing information for the application of heuristics (e.g. associate a column with an RDF property), and customizing the details (e.g. whether to reify property values, as in case of skos:definitions) through a more convenient interface that avoids writing a transformation rule. Indeed, the command-line utility only supports the basic functionalities of Sheet2RDF, while this more advanced input requires a more complete user interface.

One such interface has been developed as an extension for Semantic Turkey, which enables the user to refine the proposed rules (without actually writing them), possibly by exploiting the already assessed information in an RDF dataset. The extension does not simply relay user input to the library (as it happens in the command-line utility), but it also integrates in the loop the user and the underlying triple store, which is managed by Semantic Turkey. This integration allows the user to refine the transformation skeleton by supplying additional hints, which are based on information already assessed in the triple store. As an example, the user may associate a column from the spreadsheet with an existing property. In this task, the user is supported with a hierarchical visualization that is populated with the information contained in the current ontology.

5 Use Cases and Community Feedback

The first and immediate use case we faced (actually the one that led to the development of the system) was the automatic generation of Thesauri and Knowledge Organization Systems (KOSs) [18] in general. The standard modeling language in the RDF family for representing KOSs is SKOS [19], together with its extension for reified labels SKOS-XL [17]. Thanks to SKOS and SKOS-XL, the communities of librarians, metadata developers and archivists opened up to the Semantic Web, though this shift was not painless. The processes that used to lead to the development of KOSs in the pre-RDF era were following the classical “domain experts handling something more or less understandable (mostly to them) to data geeks” workflow pattern. These craftworks caused a proliferation of in-house conventions and (mostly one-time) conversions to some formal (though no more standard than the original one) data representation.

No wonder that the preferred representation adopted by librarians was spreadsheets, as that solution offered a rare combination of high usability with a minimal underlying structure adopted to detail and organize the concept descriptions, their hierarchical organization and their relationships. Our experience as developers of Knowledge Management Systems (Semantic Turkey and VocBench), and especially the feedback we gathered from users, showed us that spreadsheet-import is a much desired feature, but its demand is an order of magnitude larger when considering KOS developers. The immediate problem with this feature is that it is perceived as an import, while it is actually, as already discussed in this paper, a transformation, in that most of the assumptions underlying the organization of data in the spreadsheet (which offer no more than a bidimensional matrix) are implicit in the mind of its creator.

We thus examined a few real use cases we received from organizations willing to bring their Excel tables into SKOS or SKOS-XL. These use cases emerged in different contexts, such as research projects, being directly involved in the consortium (SemaGrow [`http://semagrow.eu/`]) or through partnership with consortium members (as for agINFRA [`http://aginfra.eu/`]), or as direct feedback gathered through the community groups (mailing lists, forums..) of our aforementioned knowledge management platforms. In some cases we have been directly involved in the realization of the target RDF representation (Soil Data Linked Dataset, FAO Land and Water thesaurus, FAO Topics vocabulary), while in other cases we provided support through the mailing list to users directly using Sheet2RDF in importing their own data. A few characteristics emerged as recurring across the various examined cases:

1. *A concept-per-line approach*: in all cases, a concept description was expressed through a single line of the spreadsheet file. This is no wonder: while multiple lines could fit more closely (and completely) the structure of an RDF graph (e.g. each row representing an individual triple), the choice of spreadsheet is mostly dictated by a need for simplicity, creating a match between identity of concepts and lines in the spreadsheet was a natural path rather than a meditated choice.

2. *Concept references (especially hierarchy)*: thesauri often come as hierarchy of concepts, and in order to represent a hierarchy (or express any relationship between concepts) across linear descriptions, there is the need for identifiers. At the same time, usually these identifiers were not available or, in the better cases, not clearly expressed. Labels represented the most common case of implicit identifier. This cleared out any possibility for ambiguous names; however, in some of the analyzed cases there were an explicit terminological separation between “terms” (which uniquely define concepts in a domain) and alternative lexicalizations.

3. *Implicit bindings between subsets of values*: often, users represented as a linear array of independent characteristics, elements that were actually grouped into bound subsets of elements dependent from each other. Specific cases include:
   (a) Reified labels, lexical descriptions etc..: very often in thesauri, lexical descriptors of any kind bring metadata with them, such as editorial notes (related to the same lexical content rather than to concepts) date-of-creation/update, provenance (e.g. author of the description) and so on. As such, columns following a

reified label are usually not providing further characteristics of the concept but they describe its label. This binding may be guessed by a person by reading the spreadsheet content, while for sure it is not explicit to machines processing the spreadsheet.

(b) Information about atomic elements sparse across two or more cells: a common case are labels (even simple literals) and their associated language, most often when there are conceptual resources represented non-uniformly in a plethora of languages. While the common approach is to define the language of the labels in a column header (e.g. one or more columns for English labels, for French ones), users might avoid excessive proliferation of columns and data sparseness in the sheet, by having pairs of columns, holding the label content and language respectively. Usually new column pairs are just added upon need, when they are not enough to represent the concept with the larger number of labels.

Thanks to the expressiveness of PEARL, all of the above phenomena could be managed successfully. However, our intent was to cover as much as possible of the typical cases, minimizing the amount of human effort to be carried on the PEARL transformation. This can be dealt with either in a completely automatic fashion, by having sheet2rdf guessing what is needed in/how to handle a particular situation, or through human computer-interaction. In the latter case, the machine is able to detect potential patterns and suggest choices to be taken by the user, presenting them in an understandable way and still hiding the transformation technicalities

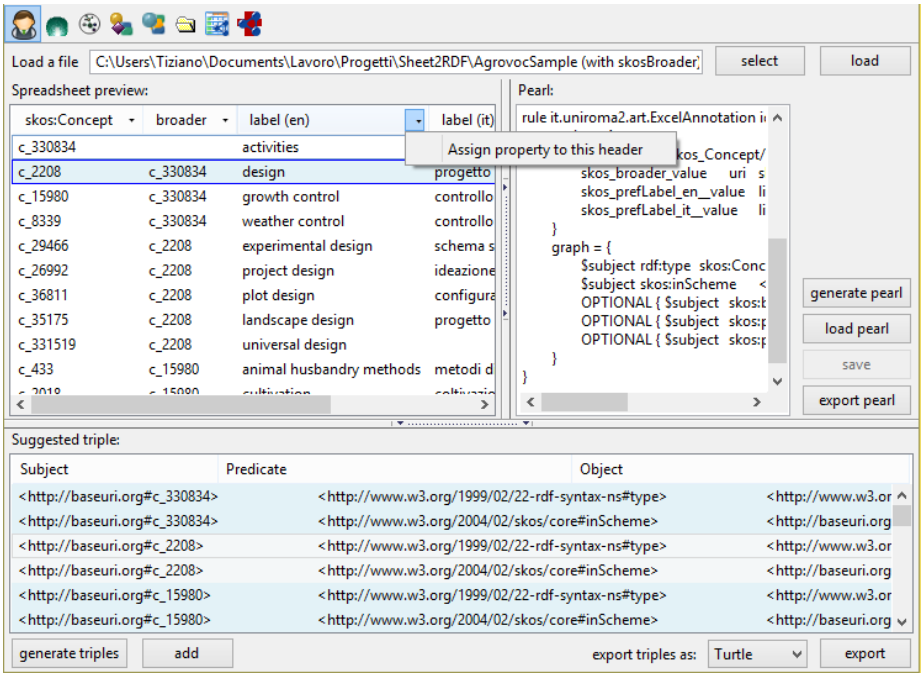

Fig. 2. Sheet2RDF user interaction

which the naïve user it not able to handle. In addressing these issues, we have followed a combination of both approaches. The first one has been pursued by defining a series of conventions that are recognized by the system and that automate the transformation generation (`http://art.uniroma2.it/sheet2rdf/docu-mentation/heuristics.jsf`). At the same time, by benefiting from the possibilities offered by Sheet2RDF integration into editing systems, we have plugged interactive wizards (see Fig. 2) raising issues/warnings to the user, or just signaling that human intervention is welcome (e.g. by highlighting the header of a column and providing menu boxes with choices) in order to disambiguate among different possible choices. We set default choices (made customizable per system and by the user), so to minimize even this interactive part by tailoring choices to systems/user needs.

6 Conclusion

In this paper, we presented Sheet2RDF, a system for importing and lifting spreadsheets to RDF. Sheet2RDF combines a flexible mapping specification language with the automatic generation of skeletal mappings, by recognizing patterns in the input spreadsheet. This approach filled a void in the landscape of currently available systems, which tend to be either minimally customizable or highly dependent on human intervention for the specification of the transformation.

Sheet2RDF supports human participation in the process by providing several levels of interaction, each one trading generality for ease of usage. Automatically generated mapping skeletons aim to cover commons cases, while manual refinement may be required for specific cases. A graphical user interface enables some refinements without the need to explicitly use the underling mapping specification language.

Acknowledgments. This research has been partially supported by the EU funded project SemaGrow (`http://www.semagrow.eu/`) under grant agreement no: 318497.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific American **279**(5), 34–43 (2001)
2. Scharffe, F., Atemezing, G., Troncy, R., Gandon, F., Villata, S., Bucher, B., Hamdi, F., Bihanic, L., Képéklian, G., Cotton, F., Euzenat, J., Fan, Z., Vandenbussche, P.-Y., Vatant, B.: Enabling linkeddata publication with the datalift platform. In: AAAI Workshop on Semantic Cities (2012)
3. Prud'hommeaux, E., Seaborne, A.: SPARQL query language for RDF. In: World Wide Web Consortium - Web Standards, January 15 2008. http://www.w3.org/TR/rdf-sparql-query/

4. Lebo, T., Williams, G.: Converting governmental datasets into linked data. In: Proceedings of the 6th International Conference on Semantic Systems, New York, NY, USA, pp. 38:1–38:3 (2010)
5. Han, L., Finin, T.W., Parr, C.S., Sachs, J., Joshi, A.: RDF123: from spreadsheets to RDF. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 451–466. Springer, Heidelberg (2008)
6. W3C: The RDF data cube vocabulary. In: World Wide Web Consortium (W3C), January 14 2014. http://www.w3.org/TR/vocab-data-cube/
7. W3C Open Annotation Community Group: Open annotation data model. In: World Wide Web Consortium (W3C), February 08 2013. http://www.openannotation.org/spec/core/
8. W3C: PROV-DM: the PROV data model. In: World Wide Web Consortium (W3C), April 30 2013. http://www.w3.org/TR/prov-dm/
9. Ermilov, I., Auer, S., Stadler, C.: CSV2RDF: user-driven CSV to RDF mass conversion framework. In: Proceedings of the ISEM 2013, Graz, Austria, September 04–06 2013
10. Fiorelli, M., Pazienza, M.T., Stellato, A., Turbati, A.: CODA: Computer-aided ontology development architecture. IBM Journal of Research and Development **58**(2/3), 14:1–14:12 (2014)
11. Fiorelli, M., Gambella, R., Pazienza, M.T., Stellato, A., Turbati, A.: Semi-automatic knowledge acquisition through CODA. In: Ali, M., Pan, J.-S., Chen, S.-M., Horng, M.-F. (eds.) IEA/AIE 2014, Part II. LNCS, vol. 8482, pp. 78–87. Springer, Heidelberg (2014)
12. Ferrucci, D., Lally, A.: Uima: an architectural approach to unstructured information processing in the corporate research environment. Nat. Lang. Eng. **10**(3–4), 327–348 (2004)
13. Pazienza, M.T., Stellato, A., Turbati, A.: PEARL: ProjEction of annotations rule language, a language for projecting (UIMA) annotations over RDF knowledge bases. In: LREC, Istanbul (2012)
14. Pazienza, M.T., Scarpato, N., Stellato, A., Turbati, A.: Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management. Semantic Web Journal **3**(3), 279–292 (2012)
15. Caracciolo, C., Stellato, A., Rajbahndari, S., Morshed, A., Johannsen, G., Keizer, J., Jacques, Y.: Thesaurus maintenance, alignment and publication as linked data: the AGROVOC use case. International Journal of Metadata, Semantics and Ontologies (IJMSO) **7**(1), 65–75 (2012)
16. Stellato, A., Rajbhandari, S., Turbati, A., Fiorelli, M., Caracciolo, C., Lorenzetti, T., Keizer, J., Pazienza, M. T.: VocBench: a Web Application for Collaborative Development of Multilingual Thesauri. In : The Semantic Web: Trends and Challenges. Springer International Publishing (2015) (accepted for publication)
17. World Wide Web Consortium (W3C): SKOS simple knowledge organization system eXtension for labels (SKOS-XL). In: World Wide Web Consortium (W3C), August 18 2009. http://www.w3.org/TR/skos-reference/skos-xl.html
18. Hodge, G.: Systems of Knowledge Organization for Digital Libraries: Beyond Traditional Authority Files. Council on Library and Information Resources, Washington, DC (2000)
19. World Wide Web Consortium (W3C): SKOS simple knowledge organization system reference. In: World Wide Web Consortium (W3C), August 18 2009. http://www.w3.org/TR/skos-reference/