Lifting Spreadsheets Made Easy: Combining Power and Magic in Sheet2RDF

Armando Stellato[1[0000-0001-5374-2807]], Manuel Fiorelli[1[0000-0001-7079-8941]], Tiziano Lorenzetti[2[0000-0001-5676-8877]], Andrea Turbati[2[0000-0002-6214-4099]]

[1] ART Research Group
Tor Vergata University of Rome
Via del Politecnico 1, 00133 Rome, Italy
{stellato, manuel.fiorelli}@uniroma2.it
[2] Lore Star srl
Via Leonida Rech 77, 00156 Rome, Italy
{tiziano.lorenzetti, andrea.turbati}@lorestar.it

Abstract. In this paper, we present an update to Sheet2RDF, a system for lifting tabular data to RDF. Based on an updated comparison with related work, we highlight Sheet2RDF's combination of flexibility and usability, enabled by a powerful transformation language, now being coupled with a richer graphical wizard and heuristics/conventions that simplify the specification of the transformation. We then discuss the enhancements to Sheet2RDF: some of them, related to the expressiveness of the transformation, are motivated by new use cases, such as dataset update, transformation of relational databases, and processing multiple sheets (or tables). Other improvements are more related to usability and improving Sheet2RDF's combined approach; we highlight in this category the substantial improvements to the wizard: an explicit model allows saving, sharing, and reloading of wizard state, while scoped code fragments finally allow for combining the wizard and code editing, sidestepping the non-invertibility of the transformation code generation from the wizard through synced code fragments.

Keywords: Human-Computer Interaction, Ontology Engineering, Ontology Population, Text Analytics, UIMA

1 Introduction

The Semantic Web [1] has been a peculiar evolution of the Web: differently from other advancements, such as the read-write Web, or Web 2.0, it underwent a long development process and an even slower adoption by the masses. The reasons are manyfold: the massive stack of new languages, protocols, and technologies that took time to reach industry level on the one hand and, under a social perspective, the fact that publishing data is usually not interesting to end users, being confined to companies or organizations, the former not being eager to embrace a Linked Open Data revolution, especially for what concerns the “open” part. Indeed, with achieved technological maturity, and

© Springer-Verlag Berlin Heidelberg 2011

pushed by a need for social innovation such as Open Government initiatives, the publication of open data according to Semantic Web standards is today a reality: while many companies are experimenting new business models based on the disclosure of the data they own, public organizations and institutions have it as a must-have, with several "data." subdomains sprouting over the Web for all relevant public actors worldwide. Public organizations were full of so called "authority tables", offering classifications, thesauri or simply mere data that has been carefully assembled and vetted (hence the name "authority") by domain experts, usually stored using spreadsheets, which thus became the de facto standard for storing (and surely interchanging) information.

To cope with these challenges, in 2015 we introduced [2] SheetRDF, a platform for the acquisition and transformation of tabular data into RDF datasets. The system was based on Apache UIMA [3] and CODA [4,5], two frameworks for, respectively, knowledge acquisition and RDF triplification, which have been vertically adopted for lifting the content of spreadsheets to the semantics and organization of ontological content. A user interface for two RDF management platforms, Semantic Turkey [6] and VocBench [7], complemented the system, by supporting the user with a graphical editor for transformation rules, with syntax highlighting and completion based not only on the syntax of the language, but also feeding elements from the same data/vocabularies edited in the platform. Today, Semantic Turkey dropped its original Firefox plugin UI and is still being developed as a backend service-based platform for Semantic Web applications. VocBench has instead evolved, in its third incarnation [8,9], into a fully-fledged all-purpose collaborative Semantic Web development platform, widely adopted worldwide both as a single-user desktop tool and as a powerful collaborative environment in large organizations[1]. In this context, Sheet2RDF evolved as well, meeting higher standards of expressive power with richer transformation possibilities and usability, thanks to a graphical wizard capable of guiding the user through all the steps of spreadsheet data lifting, while keeping hooks to the transformation language. Other important addenda to this renewed resource include multiple-sheet management and database processing.

The paper is structured as follows: section 2 provides an updated survey on the state of the art. Section 3 describes our approach with respect to the state of the art, both in terms of the original system and of its subsequent evolution. Section 4 introduces the radical improvements to the platform, while section 5 draws conclusions.

2 Related Work

This section provides insights into systems (see **Table 1**) for lifting and importing data from a variety of tabular formats (e.g., Microsoft Excel, Open Office, CSV, TSV, and even relational databases) into RDF, based on a recent survey we conducted [10].

We first note that most systems produce arbitrary RDF data, with only a few notable exceptions, MappingMaster and Populous, which target OWL axioms, and TabLinker,

[1] VocBench adopters include Food and Agriculture Organization of the United Nations, European Commission, US Department of Agriculture (National Agricultural Library), LifeWatch ERIC (the e-Science European infrastructure for biodiversity and ecosystem research)

which targets the Data Cube [11] vocabulary. These exceptions arise from the need to abstract difficult RDF modeling, and especially for TabLinker, from the complex layout of the input. In fact, most systems deal with tabular data consisting of homogeneous rows, each describing a different resource. Spread2RDF also allows for columns to be grouped into blocks that describe sub-resources. PoolParty extends this layout by using indentation to represent hierarchical relationships. Finally, we should mention that Karma, RMLEditor and GraphDB OntoRefine (now independent from GraphDB) support hierarchical data models (e.g., XML) in addition to simpler tabular formats.

The systems differ greatly in the degree of mapping customization. At one extreme, Any23 does not allow any customization and takes a systematic approach to converting rows into resource descriptions by deriving properties from column headers. DataLift, GraphDB OntoRefine, LinkedPipes ETL, and UnifiedViews build on this idea by adding an extra step to refine the generated RDF data using SPARQL as the mapping language. Tarql combines these two steps by evaluating SPARQL constructs over tabular data. Other systems build on this one-step process but introduce custom mapping languages. Sparlify proposes a clever combination of SQL to query the input data and SPARQL graph patterns to generate RDF data (much like what Sheet2RDF does). On top of this, CSV2RDF uses a Semantic MediaWiki [12] extension to manage mappins as MediaWiki templates. RDF123 uses RDF templates, while Vertere and Csv2rdf4lod-automation describe the mapping in RDF. XLWrap combines both approaches. Mapping Master expresses mappings like spreadsheet formulas that can be applied to ranges of data (to accommodate complex layouts). Populous instead uses OPPL [13] for executable ontology design patterns [14]. LODRefine, Karma and RMLEditor provide a graphical domain-specific language (DSL) for a powerful transformation language: the latter two are based on extensions of the R2RML [15] specification for relational database mapping (expressed in RDF). GraphDB OntoRefine (9.4) introduced an analogous user interface. TopBraid Composer has instead a simpler wizard. Grafter and Spread2RDF uses an internal DSL developed in general-purpose programming languages (the former also provides Grafterizer as a web front-end). SKOS Play! defines conventions (adopted by Sheet2RDF as well) to encode the mapping in column headers. TabLinker is tied to actual spreadsheet formats because it uses cell styles and comments to represent the mapping. It also uses Open Annotation [16] to represent harmonization rules, and PROV [17] for provenance

Reuse is a concern in mapping specification (e.g., use the same pattern for SKOS-XL [18] preferred and alternative labels): Spread2RDF templates, Sheet2RDF patterns and triggers or cascades of SPARQL queries in UnifiedViews and LinkedPipes ETL.

The analyzed systems provide a variety of features to support their users: user interfaces, skeleton generators, suggestions based on optimization techniques and machine learning, syntax highlighting and completion, lookup of relevant terms (e.g., for mapping columns to ontology properties) in the target ontology or external services (e.g., [19]), use of NER and reconciliation services, access to crowdsourcing platforms, code editors and debuggers, and suggestions for automating the assembly of transformation processes.

Table 1. Comparative summary of tabular data to RDF conversion systems.

System name	Output	Input	Mapping	Assistance to Mapping
Any23 [20]	RDF	rows	—	—
Csv2rdf [21]	RDF	rows	MediaWiki template	Generation skeleton mappings
csv2rdf4lod-automation [22]	RDF	rows	Conversion ontology	—
DataLift [23]	RDF	rows	SPARQL	LOV, Linking services
Grafter [24]	RDF	rows	Clojure DSL	Grafterizer
GraphDB OntoRefine [25]	RDF	rows	SPARQL, SPIN, GREL, Graphical DSL	SPARQL editor with syntax highlighting and completion
Karma [26]	RDF	Hierarchical data	Graphical DSL, KR2RML	ML, Steiner tree optimization, lookup target ontology
LinkedPipes ETL [27]	RDF	rows	SPARQL	debugging, component suggestion
LODRefine [28]	RDF	Hierarchical data	Graphical DSL, GREL	lookup in the target ontology, CrowdFlower, NER services, reconciliation with DBpedia
Mapping Master [29]	OWL	varied layouts	M^2 Language	user interface
PoolParty [30]	RDF	indented rows	—	—
Populous [31]	OWL	rows	OPPL	—
RDF123 [32]	RDF	rows	RDF template	—
RMLEditor [33]	RDF	Hierarchical data	Graphical DSL, RML	LOV
Sheet2RDF	RDF	rows	PEARL	PEARL editor, heuristics, lookup in the target ontology
SKOS Play! [34]	RDF	rows	structured header	—
Sparqlify [35]	RDF	rows	SML	—
Spread2RDF [36]	RDF	column blocks	Ruby DSL	—
TabLinker [37]	Data Cube	data cubes	Annotated Excel file	—
Tarql [38]	RDF	rows	SPARQL	—
TopBraid Composer [39]	RDF	rows	Wizard	—
Unified Views [40]	RDF	rows	SPARQL	debugging
Vertere-RDF [41]	RDF	rows	RDF-based language	—

System name	Output	Input	Mapping	Assistance to Mapping
XLWrap [42]	RDF	varied layouts	Conversion ontology + RDF template	—

3 Methods and Rationale

Since its initial conception, Sheet2RDF has embraced the reuse of a mature framework for knowledge acquisition with a twofold rationale: i. exploit most of the framework already developed solutions and continue to grow with it ii. provide an integrated solution in those environments where such framework is already adopted. At the same time, in order not to make it a niche solution, the system should have been verticalized to handle spreadsheets, providing a convenient solution that looks and feels as originally tied to the domain as possible.

Other requirements we identified before design and development started were:

- be particularly *well suited for transforming record-like data*, especially authority tables and thesauri, as this was a common use case that we faced
- *neat separation of concerns*: easily address changes to the target RDF representation, without having to change the interpretation of the source content
- *optimal trade-off between automation and expressiveness*: make the system smart enough to guess transformations, and support humans in refining them
- *integrate the system into RDF (and SKOS in the specific) management systems*, blending background knowledge provided by the edited data with automatism and human capabilities through human-computer interfaces

The conversion of a spreadsheet to RDF can be seen as an instance of the general task of triplifying (semi/un)structured information. CODA was developed by us as an extension of the UIMA framework for Unstructured Information Management for transforming content extracted by UIMA (expressed as typed feature structures [43]) into RDF triples, according to arbitrarily defined target vocabularies. At the core of CODA is PEARL [44] (ProjEction of Annotations Rule Language), a transformation language from UIMA annotations (expressed through said feature structures) into RDF content.

Among the reusable features of CODA, we highlight *converters*, which are invokable functions described by a namespace, function name, and parameter list (with support for overloading). From a technological point of view, converters exploit an extension point of the CODA architecture and implement the general conversion function from a feature structure to an RDF node (i.e., IRI or Literal, since Bnodes have no name and thus no shape that needs to be created). These are developed in Java and then, once integrated into CODA, become effective according to their syntax. Converters can even be published on the Web and, by means of URI references, can be automatically downloaded into a CODA instance. In this way, given a transformation document that uses a given converter that is not present in the system, Sheet2RDF can look for its presence on the Web, resolve its contract, download it into the instance, and apply it to the document without any a-priori discovery, installation, configuration, etc.

Other advantages of reusing CODA are realized when Sheet2RDF is integrated into VocBench, which already uses CODA for other tasks (creation of custom forms and, very soon, acquisition of information from unstructured content): VocBench users benefit from a language for knowledge acquisition and transformation that can be used pervasively along the platform for different purposes. Furthermore, CODA converters in VocBench can be bound to the URI generation policy for each project: instead of defining the generation of the URIs according to the same pattern that is used in each project, the converter automatically adopts the same configuration of the project. Thus, if the same transformation needs to be ported to another project (e.g., because data in the same spreadsheet format is imported into another project), the syntax does not need to be changed, and the converters will dynamically adapt to the URI conventions for that particular dataset. The last peculiarity and one of the biggest improvements of the version integrated in VocBench is a versatile wizard, described in the next section.

With this rational and core features in mind, we aimed at creating a system sitting in between the two extremes that we have seen emerging from the description of the state of the art: those systems supporting rigid transformation processes, and those completely depending on humans for setting complex transformations. With a solid transformation language at its core and a layered set of helpers, wizards, configuration possibilities, heuristics and patterns, Sheet2RDF provides a versatile environment that can adapt to the complexity of each situation with a proportional level of elaboration.

4 New Sheet2RDF: More Expressive Power and Wizard

Starting from the core of the system and looking progressively at higher levels of abstraction, we will detail here the changes that Sheet2RDF has undergone over the years.

While the basics of the PEARL language syntax have not changed radically, several addenda have been made.

First, some converters have been made more powerful or versatile, while new converters have been bundled into the system.

The former group includes the `randomURIGenerator`, which can take more parameters and adapt to contextual settings, such as those provided by the ontology editing environment that hosts Sheet2RDF.

The latter group includes the `FormatterConverter` (syntax: `coda:formatter`), which behaves like a formatted print (`printf`) in C, accepting a format string with placeholders for values provided as other parameters of the converter, the object on which the converter is applied, or the output of other converters. This introduces an important improvement to PEARL that increases its expressive power: concatenating converters, by passing the invocation of one converter as an argument of another converter. This can improve the separation of concerns in converters, since each of them can be highly specialized, while different capabilities can be obtained through nested invocations.

A further improvement to PEARL lies in the introduction of multiple pools of *memoization* (Fig. 1). The memoization feature addresses the need for consistent behavior when using a non-deterministic URI generator. It consists of annotating one or more fields of the spreadsheet so that, each time a new input is encountered, it is marked and

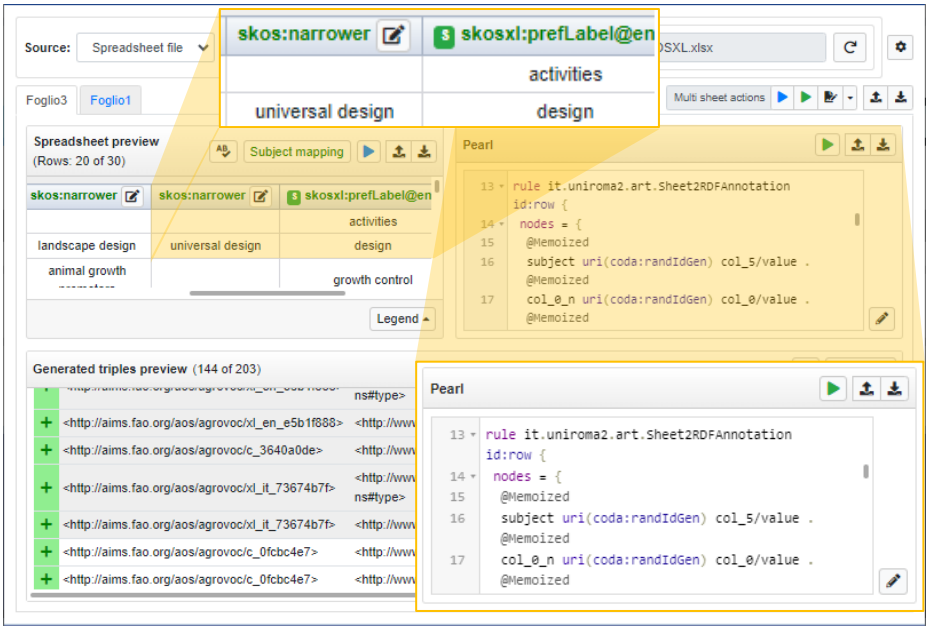

Fig. 1. An Excel file with no explicit URIs and identification based memoization of labels. Some parts of the image have been zoomed in.

stored. Whenever an already marked input is found, instead of generating its URI, the URI already associated with that input is recovered. This allows for some input elements to be defined as "keys" for the input, so that the same, consistent, URI is always generated for each occurrence of the same key. A common case is represented by thesauri, where the URI is usually not provided in the input spreadsheet and another element (usually a *preferred label*, which is unique in the thesaurus for each language) is used as a reference in the spreadsheet, although it does not directly concur in the generation of the URI (which is usually generated randomly or by means of a counter).

It is not uncommon to see multiple, distinct keys (e.g., for objects that are not described explicitly through a record, but implicitly by references). An optional label for the memoization annotation allows for the creation of these multiple, distinct pools.

Another important addition to PEARL lies in the ability to remove triples, not just add them. Often used for importing data, Sheet2RDF neglected deleting data, but this turned out to be useful in use cases such as batch deleting entities and updating datasets.

On the usability side, the user interface has been dramatically improved. The previous version of Sheet2RDF's integration within ST's Firefox UI and VocBench 2 was j a user interface for composing PEARL code, along with an overview of the spreadsheet's headers, letting the user know which ones were processed, ignored or waiting to be transformed. It was already a very convenient help, but some constraints severely limited the continued use of the wizard over time, both for maintenance and reuse. First, the transformation from the wizard state to the PEARL code is non-invertible (like most code builders, the job of which is done after generating a code skeleton). However,

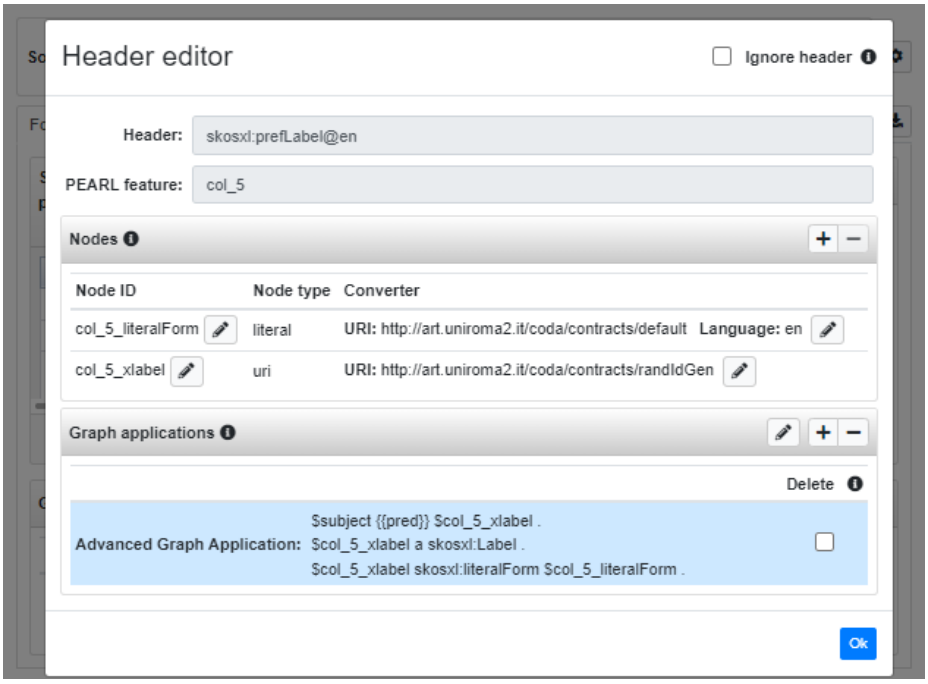

Fig. 2. Association of the column *skos:prefLabel@en* with the advanced graph pattern for reified SKOS-XL labels.

small changes made to the code later turn the wizard into a sort of first-stage rocket that is dropped and then lost forever, when it would have been desirable to keep using it. In addition, reuse was based on code only, as the state of the wizard could not be shared with other users. The new Sheet2RDF (within VocBench 3) overcomes these limitations not by attempting to invert the transformation function, rather by defining a rigorous internal model of the transformation that can be stored, reloaded and shared.

The user can thus continue working on the wizard by storing its state and restarting where they left off by reloading it. Embeddable code fragments address the non-invertibility. Instead of choosing between the wizard and the code editor, users can now open scoped code fragments (called *advanced graph applications*, as opposed to the simple one not requiring code - **Fig. 2**) and bind them to elements managed by the wizard.

The binding in the fragments can be further generalized, so that these can be stored as reusable code patterns that can be invoked and adopted in different projects. For example, the following pattern is a factory preset for reified SKOS-XL labels.

```
$subject          {{pred}}             $xlabel       .
$xlabel           a                    skosxl:Label  .
$xlabel           skosxl:literalForm   $literalForm  .
```

The special entity $subject is bound to the subject identified for the row, $xlabel and $literalForm are generated as elements of the reified label itself, and the placeholder {{pred}} can in turn be associated with the various predicates (e.g. skosxl:prefLabel or

skosxl:altLabel) that bind the label to the subject. The choice of the predicate is thus externalized, through variable bindings that instantiate the pattern over different cases.

Several heuristics (borrowed from "SKOS Play!") then automate and drive the application of advanced transformations:

- A header valued with the name of an RDF property (both full IRIs and qualified names, i.e., expressions of the form <prefix>:<localName> where the prefix is associated with a given namespace in the spreadsheet or the host system) results in the subject for each record being linked to the value, in the same record, in the cell below the header, through the property defined in the header. Also, if the property is known (e.g., it belongs to a core modeling vocabulary such as RDF(S), OWL, SKOS, SKOS-XL, or it is defined/imported in the project of the hosting editor) then its rdfs:range will be used to suggest compatible converters.
- A substring in a header of the form @<langcode>, where langcode is a two-digit (ISO 639-1) language code, indicates that the values in the cells below the header are strings tagged with that language code. Again, the converters suggested to the user are those that can generate language-tagged strings.

The PEARL code editor (including code fragment editing for the wizard) has been improved with syntax highlighting, completion, and a converter picker that guides the user through selecting a converter, one of its signatures and reading its documentation.

Two further addenda projected Sheet2RDF into a broader view of large-scale import efforts. The first one is support for multiple sheets, managed as a single global resource, even though they (and their contents) are individually identifiable. For instance, it is possible to apply cross-sheet synchronized *@memoized* annotations so that same content from linked headers of different spreadsheets has consistent URIs. This feature is prodrome to the second addendum to the system: support for relational databases.

In the state of the art, we mentioned some systems for triplifying relational databases that were adapted to import spreadsheets as if they were database tables. Sheet2RDF did the opposite. Based on the assumption that the massive import of data usually does not require complex queries and that, in any case, table *joins* can be obtained through cross-sheet memoization, Sheet2RDF has been extended to access multiple database tables as, de facto, multiple sheets within the same spreadsheet container.

5 Conclusion

In this paper we presented the evolution of Sheet2RDF since its introduction seven years ago. While some improvements were driven by new or better articulated use cases, most of the work was actually aimed at fulfilling Sheet2RDF's promise of combining flexibility, guidance, and automation. Therefore, we enhanced the underlying transformation language, and we dramatically improved the Sheet2RDF wizard. The wizard now supports a wider range of settings that in the previous version required code-level refinement of the transformation. In fact, this severely limited the usefulness of the wizard, since the non-invertibility of the transformation generation meant that the wizard could no longer be used after this refinement. In addition, an explicit, formal model of the state of the wizard now allows it to be stored, reloaded, and shared. The

ability to embed scoped code fragments now supports the combined used of the wizard and low-level coding, bypassing the non-invertibility of the transformation generation process. Finally, based on the aforementioned model, we have framed existing conventions in a more principled manner.

References

1. Berners-Lee, T., Hendler, J.A., 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. Fiorelli, M., Lorenzetti, T., Pazienza, M.T., Stellato, A., Turbati, A.: Sheet2RDF: a Flexible and Dynamic Spreadsheet Import&Lifting Framework for RDF. In Ali, M., Kwon, Y.S., Lee, C.-H., Kim, J., Kim, Y., eds. : Current Approaches in Applied Artificial Intelligence. International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2015). LNCS, vol. 9101. Springer International Publishing (2015), pp.131-140. doi: 10.1007/978-3-319-19066-2_13
3. 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). doi: 10.1017/S1351324904003523
4. Fiorelli, M., Gambella, R., Pazienza, M.T., Stellato, A., Turbati, A.: Semi-automatic Knowledge Acquisition through CODA. In : Modern Advances in Applied Intelligence - 27th International Conference on Industrial Engineering and Other Applications of Applied Intelligent System, IEA/AIE 2014, Kaohsiung, Taiwan, vol. Part II, pp.78--87. LNCS, vol. 8482 (2014). Springer, Cham. doi: 10.1007/978-3-319-07467-2_9
5. 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 (March-May 2014). doi: 10.1147/JRD.2014.2307518
6. 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). doi: 10.3233/SW-2011-0033
7. Stellato, A. et al.: VocBench: a Web Application for Collaborative Development of Multilingual Thesauri. In Gandon, F., Sabou, M., Sack, H., d'Amato, C., Cudré-Mauroux, P., Zimmermann, A., eds. : The Semantic Web. Latest Advances and New Domains. Proceedings of the 12th Extended Semantic Web Conference, ESWC 2015, Portoroz, Slovenia, 31 May - 4 June 2015. LNCS, vol. 9088. Springer, Cham (2015), pp.38-53. doi: 10.1007/978-3-319-18818-8_3
8. Stellato, A. et al.: Towards VocBench 3: Pushing Collaborative Development of Thesauri and Ontologies Further Beyond. In Mayr, P., Tudhope, D., Golub, K., Wartena, C., De Luca, E.W., eds. : 17th European Networked Knowledge Organization Systems (NKOS) Workshop. Thessaloniki, Greece, September 21st, 2017, Thessaloniki, Greece, pp.39-52 (2017)

9. Stellato, A. et al.: VocBench 3: A collaborative Semantic Web editor for ontologies, thesauri and lexicons. Semantic Web 11(5), 855-881 (Aug 2020). doi: 10.3233/SW-200370
10. Fiorelli, M., Stellato, A.: Lifting Tabular Data to RDF: A Survey. In Garoufallou, E., Ovalle-Perandones, M.-A., eds. : Metadata and Semantic Research. MTSR 2020. CCIS, vol. 1355. Springer, Cham (2021), pp.85-96. doi: 10.1007/978-3-030-71903-6_9
11. W3C: The RDF Data Cube Vocabulary. In: World Wide Web Consortium (W3C) (January 14, 2014). http://www.w3.org/TR/vocab-data-cube/. Accessed 12 December 2014
12. Krötzsch, M., Vrandečić, D., Völkel, M., Haller, H., Studer, R.: Semantic Wikipedia. Web Semantics: Science, Services and Agents on the World Wide Web 5(4), 251-261 (2007). doi: 10.1016/j.websem.2007.09.001
13. Egana, M., Antezana, E., Stevens, R.: Transforming the axiomisation of ontologies: The ontology pre-processor language. In : Proceedings of the Fifth OWLED Workshop on OWL: Experiences and Directions (OWLED-2008) (2008)
14. Gangemi, A., Presutti, V.: Ontology design patterns. In : Handbook on ontologies. Springer Berlin Heidelberg (2009), pp.221-243. doi: 10.1007/978-3-540-92673-3_10
15. Das, S., Sundara, S., Cyganiak, R.: R2RML: RDB to RDF Mapping Language. In: World Wide Web Consortium - Web Standards (September 27, 2012). https://www.w3.org/TR/r2rml/
16. W3C Open Annotation Community Group: Open Annotation Data Model. In: World Wide Web Consortium (W3C) (February 08, 2013). http://www.openannotation.org/spec/core/. Accessed 12 December 2014
17. W3C: PROV-DM: The PROV Data Model. In: World Wide Web Consortium (W3C) (April 30, 2013). http://www.w3.org/TR/prov-dm/. Accessed 12 December 2014
18. 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. Accessed 22 March 2011
19. Vandenbussche, P.-Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web. Semantic Web 8(3), 437-452 (December 2016). doi: 10.3233/SW-160213
20. Apache Any23 - CSV Extractor Algorithm. Available at: https://any23.apache.org/dev-csv-extractor.html
21. Ermilov, I., Auer, S., Stadler, C.: CSV2RDF: User-Driven CSV to RDF Mass Conversion Framework. In : Proceedings of the ISEM '13, September 04 - 06 2013, Graz, Austria (2013). doi: 10.1145/2506182.2506196
22. Lebo, T. et al.: Producing and Using Linked Open Government Data in the TWC LOGD Portal. In Wood, D., ed. : Linking Government Data. Springer New York (2011), pp.51-72. doi: 10.1007/978-1-4614-1767-5_3
23. Scharffe, F. et al.: Enabling linkeddata publication with the datalift platform. In : AAAI workshop on semantic cities (2012)
24. Grafter - Linked Data Machine Tools. Available at: http://grafter.org/
25. Ontotext: Ontotext GraphDB. Available at: https://www.ontotext.com/products/graphdb/

26. Knoblock, C.A. et al.: Semi-automatically Mapping Structured Sources into the Semantic Web. In Simperl, E., Cimiano, P., Polleres, A., Corcho, O., Presutti, V., eds. : The Semantic Web: Research and Applications. Proceedings of the 9th Extended Semantic Web Conference, ESWC 2012, Heraklion, Crete, Greece, May 27-31 2012. LNCS, vol. 7295. Springer International Publishing (2012), pp.375-390. doi: 10.1007/978-3-642-30284-8_32
27. Klímek, J., Škoda, P., Nečaský, M.: LinkedPipes ETL: Evolved Linked Data Preparation. In : The Semantic Web (Lecture Notes in Computer Science) vol. 9989. (2016), pp.95-100
28. LODRefine. Available at: https://github.com/sparkica/LODRefine
29. O'Connor, M.J., Halaschek-Wiener, C., Musen, M.A.: Mapping Master: A Flexible Approach for Mapping Spreadsheets to OWL. In : The Semantic Web – ISWC 2010. Springer Berlin Heidelberg (2010). doi: 10.1007/978-3-642-17749-1_13
30. PoolParty Semantic Suite. Available at: https://www.poolparty.biz/
31. Jupp, S. et al.: Populous: a tool for building OWL ontologies from templates. BMC Bioinformatics 31(1) (2012). doi: 10.1186/1471-2105-13-S1-S5
32. Han, L., Finin, T., Parr, C., Sachs, J., Joshi, A.: RDF123: From Spreadsheets to RDF. In Sheth, A. et al., eds. : The Semantic Web - ISWC 2008. LNCS, vol. 5318. Springer Berlin Heidelberg (2008), pp.451-466. doi: 10.1007/978-3-540-88564-1_29
33. Heyvaert, P. et al.: RMLEditor: A Graph-Based Mapping Editor for Linked Data Mappings. In : The Semantic Web. Latest Advances and New Domains. LNCS, vol. 9678. Springer, Cham (2016), pp.709-723. doi: 10.1007/978-3-319-34129-3_43
34. SKOS Play! - Thesaurus & Taxonomies. Available at: https://skos-play.sparna.fr/
35. Sparqlify. Available at: http://aksw.org/Projects/Sparqlify.html
36. Spread2RDF. Available at: https://github.com/marcelotto/spread2rdf
37. TabLinker. Available at: https://github.com/Data2Semantics/TabLinker
38. Tarql: SPARQL for Tables. Available at: https://github.com/cygri/tarql
39. TopBraid Platform. In: TopQuadrant. Available at: https://www.topquadrant.com/technology/topbraid-platform-overview/
40. Knap, T. et al.: UnifiedViews: An ETL tool for RDF data management. Semantic Web 9(5), 661-676 (2018). doi: 10.3233/SW-180291
41. Vertere-RDF. Available at: https://github.com/knudmoeller/Vertere-RDF
42. Langegger, A., Wöß, W.: XLWrap – Querying and Integrating Arbitrary Spreadsheets with SPARQL. In : The Semantic Web - ISWC 2009. LNCS, vol. 5823. Springer, Berlin, Heidelberg (2009), pp.359-374. doi: 10.1007/978-3-642-04930-9_23
43. Carpenter, B.: The Logic of Typed Feature Structures. Cambridge Tracts in Theoretical Computer Science (hardback) edn. 32. Cambridge University Press (1992)
44. 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)