Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 7194–7203
Marseille, 11–16 May 2020

Editing OntoLex-Lemon in VocBench 3

Manuel Fiorelli[1], Armando Stellato[1], Tiziano Lorenzetti[1], Andrea Turbati[1], Peter Schmitz[2], Enrico Francesconi[2], Najeh Hajlaoui[2], Brahim Batouche[2]
[1] University of Rome "Tor Vergata", Department of Enterprise Engineering, via del Politecnico 1, 00133 Roma, Italy
{fiorelli,turbati}@info.uniroma2.it, stellato@uniroma2.it, tiziano.lorenzetti@gmail.com
[2] Publications Office of the European Union, Luxembourg
{Peter.SCHMITZ, Enrico.FRANCESCONI}@publications.europa.eu,
{Najeh.HAJLAOUI, Brahim.BATOUCHE}@ext.publications.europa.eu

Abstract
OntoLex-Lemon is a collection of RDF vocabularies for specifying the verbalization of ontologies in natural language. Beyond its original scope, OntoLex-Lemon, as well as its predecessor Monnet *lemon*, found application in the Linguistic Linked Open Data cloud to represent and interlink language resources on the Semantic Web. Unfortunately, generic ontology and RDF editors were considered inconvenient to use with OntoLex-Lemon because of its complex design patterns and other peculiarities, including indirection, reification and subtle integrity constraints. This perception led to the development of dedicated editors, trading the flexibility of RDF in combining different models (and the features already available in existing RDF editors) for a more direct and streamlined editing of OntoLex-Lemon patterns. In this paper, we investigate on the benefits gained by extending an already existing RDF editor, VocBench 3, with capabilities closely tailored to OntoLex-Lemon and on the challenges that such extension implies. The outcome of such investigation is twofold: a vertical assessment of a new editor for OntoLex-Lemon and, in the broader scope of RDF editor design, a new perspective on which flexibility and extensibility characteristics an editor should meet in order to cover new core modeling vocabularies, for which OntoLex-Lemon represents a use case.

Keywords: Lemon, OntoLex, VocBench, Lexicon, RDF

1. Introduction

The OntoLex[1] W3C Community Group released on 10 May 2016 a final report (Cimiano, McCrae, & Buitelaar, 2016) defining the OntoLex-Lemon model (McCrae et al., 2017) for the representation of lexicons in connection to ontologies. The so-called *lemon* model, standing for Lexicon Model for Ontologies, was agreed by an international group including representatives of most past efforts in this area. The community group refined the foundation laid down by Monnet *lemon* (McCrae, Spohr, & Cimiano, 2011), and clarified its modularity, by articulating the model as a set of OWL ontologies specifying different aspects of the ontology-lexicon interface.

Beyond its intended scope, OntoLex-Lemon (as well as its predecessor Monnet *lemon*) was applied to the representation and interlinking of wordnets and other language resources on the Semantic Web. Indeed, Chiarcos, Nordhoff, and Hellmann (2012) have acknowledged the benefits of the adoption of the linked data best practices in linguistics, and *lemon* (in general) became critical to the construction of the so-called Linguistic Linked Open Data[2] (LLOD), a sub-cloud of the Linked Open Data cloud[3] related to language resources.

While ontology and RDF editors should support OntoLex-Lemon editing, McCrae, Montiel-Ponsoda, and Cimiano (2012) argued with respect to *lemon* that "generic data-driven editors would be difficult to use for non-expert users". They noticed the lack of dedicated rendering (i.e. visualization) of certain model elements and, moreover, the inability to manipulate certain model elements as if they were a single piece irrespectively of these being realized as a complex graph of many interrelated resources. These authors addressed their concerns about generic editors by developing a dedicated *lemon* editor called *lemon source*.

However, we contend that this approach is unsatisfactory as well. Firstly, an editor dedicated to (a specific application of) a model (in this case *lemon*) might lose the flexibility of RDF, which allows for combining different vocabularies and models, unless the editor supports importing and using arbitrary ontologies. However, following this path, we would be forced to recreate an ontology editor inside a *lemon* editor. A second problem, related to the former, is that a purpose-built editor may lack most functionalities already found in ontology and RDF editors, unless they are recreated from scratch (as McCrae et al. did for collaboration, history, etc.). Moreover, the very assumption behind these dedicated editors somehow neglects the flexibility of modern "data-driven editors", which usually offer configuration options and extension points to cover different scenarios beyond the basic ones. Indeed, support for multi-model editing is often achieved by layering model-specific features and customizations onto basic functionalities, providing a tailored yet coherent experience across different models, possibly mixed together. Jupp, Bechhofer, and Stevens (2009) investigated the extension of the popular ontology editor Protégé to support SKOS editing. Similarly, our collaborative knowledge development environment VocBench 3[4] (Stellato et al., 2017; Stellato et al., 2019) achieves support for ontologies and thesauri as specializations of generic RDF editing. Under the hood, VocBench is powered by the knowledge management framework Semantic Turkey[5] (Pazienza et al., 2012).

[1] https://www.w3.org/community/ontolex/
[2] http://www.linguistic-lod.org/
[3] https://lod-cloud.net/
[4] http://vocbench.uniroma2.it/
[5] http://semanticturkey.uniroma2.it/

```
:englishDBpediaLexicon a lime:Lexicon ;
   lime:language "en" ;
   dc:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
   lime:entry :comics-character-entry .

:comics-character-entry a ontolex:MultiWordExpression ;
   a lexinfo:NounPhrase ;
   lime:language "en" ;
   dc:language <http://id.loc.gov/vocabulary/iso639-1/en> ;
   ontolex:canonicalForm [ a ontolex:Form ;
      lexinfo:number lexinfo:singular ;
      ontolex:writtenRep "comics character"@en] ;
   ontolex:otherForm [ a ontolex:Form ;
      lexinfo:number lexinfo:plural ;
      ontolex:writtenRep " comics characters"@en] ;
   decomp:constituent :comics-character-comp1, :comics-character-comp2 ;
   rdf:_1 :comics-character-comp1 ;
   rdf:_2 :comics-character-comp2 ;
   ontolex:sense [ a ontolex:LexicalSense ;
      ontolex:reference dbo:ComicsCharacter ;
      synsem:isA :comics-character-arg  ] ;
   ontolex:synBehavior [ a ontolex:SyntacticFrame ;
      a lexinfo:NounPredicateFrame ;
      lexinfo:copulativeArg :comics-character-arg ] ;
   vartrans:translatableAs <http://../personaggio-dei-fumetti> .

:comics-character-comp1 a decomp:Component ;
   decomp:correspondsTo :comics-entry  .

:comics-character-comp2 a decomp:Component ;
   decomp:correspondsTo :character-entry  .
```

Figure 1:An OntoLex-Lemon lexical entry for the class dbo:ComicsCharacter

The aforementioned drawbacks of dedicated *lemon* editors motivated us to investigate the approach that has already proved successful for other models: augmenting an ontology or RDF editor with facilities required for conveniently editing OntoLex-Lemon lexicons and onto-lexicon interfaces, while benefiting from the tool's already available feature set and flexible editing capabilities. While we extended our own VocBench 3, the same approach and observations might apply equally well to other editors, such as Protégé or TopBraid Composer. The requirements of the present work were sketched in Fiorelli et al. (2018).

Our work was carried on within the development of the Public Multilingual Knowledge Management Infrastructure for the Digital Single Market[6] (PMKI), an action of the ISA[2] programme[7] that aims to overcome language barriers within the EU by means of multilingual tools and services. In this context, there was a need for coordinated instruments for advanced lexicalization of RDF resources (be them ontologies, thesauri and datasets in general) and for alignment of their content, while OntoLex-Lemon was chosen as the preferred model for the representation of language resources. A system that seamlessly supports these diverse assets as well as their joint use is certainly appealing to this goal. PMKI will exploit both VocBench as a shared, collaborative editing platform for thesauri and lexicons, and Semantic Turkey as a core RDF service suite for implementing its forthcoming dissemination platform, which will provide a public web portal for uploading and browsing these diverse assets.

2. Related Work

McCrae et al. (2011) surveyed numerous applications of OntoLex-Lemon, including the representation of diverse language resources. These works often discuss the applicability and limitations of the model, and sometimes propose extensions. The extensibility of OntoLex-Lemon is a consequence of its roots in the Semantic Web, but extensions of the model may require intrusive modifications of purpose-built editors.

Lemon source is an editor for (Monnet) *lemon* based on the paradigm of semantic wikis (such as OntoWiki (Frischmuth et al., 2015)). Unfortunately, there is no publicly available version of this system that is still usable.

While lexical entries often become complex, McCrae and Hunger (2014) found recurring patterns in their structure,

6 https://ec.europa.eu/isa2/actions/overcoming-language-barriers_en

7 https://ec.europa.eu/isa2/

eventually defining a catalog of *lemon* patterns for ontology-lexicons. A formal language based on these patterns enables to express lexical entries concisely without dealing with their RDF serialization, which can be generated using a dedicated converter[8]. Differently from our implementation based on *custom forms*, this converter is unable to interpret an existing RDF serialization through the pattern language. *Lemonade* (Rico & Unger, 2015) is a *lemon* editor based on some of these patterns, and therefore constrained to a particular application of *lemon* and unsuitable to scenarios requiring to combine different vocabularies. Currently, Lemonade is only available as a service.

LexO (Bellandi et al., 2017; Bellandi, Giovannetti, & Weingart, 2018) is a collaborative editor of lexical and termino-ontological resources, and particularly aimed at philologists, historical linguists and lexicographers. Ease of use is thus of paramount importance, which in general required a high level of abstraction with respect to the underlying *lemon* model. Actually, there are different versions of LexO, born in the context of the projects LexO was applied to. Each version supports different language phenomena, requiring dedicated extensions of the view and, often, of *lemon*: word roots for Arabic, transliteration and tone for Chinese, and handling of multiple alphabets, multi-language phrasemes and ancients forms without the canonical ones (because never attested) for Old-Occitan. LexO-lite[9] is the new *general-purpose* version (still to be released), which is going to support all modules of the OntoLex-Lemon model (while previous versions were based on Monnet *lemon*).

3. OntoLex-Lemon

OntoLex-Lemon is a model for the interfacing of ontologies and lexicons: its aim is to characterize how an ontology is verbalized in natural language to an extent beyond the possibility of current *lexicalization models* (such as RDFS and SKOS/SKOS-XL). The model is realized as a suite of ontologies, called *lemon* modules, which deal with syntax-semantics interfacing (*synsem*), decomposition of lexical entries (*decomp*), variation and translation (*vartrans*) and linguistic metadata (*lime*) (Fiorelli et al., 2015), while a core module (*ontolex*) defines the backbone upon which the other modules rest.

Figure 1 illustrates the use of the various modules to describe the lexical entry "comics character" denoting the class dbo:ComicsCharacter in the DBpedia ontology. Lexical entries are first grouped into a *lexicon*, which holds metadata such as the *language* and the *number of lexical entries* (not shown in the example, for conciseness). A lexical entry has a *canonical form* (usually corresponding to its *lemma*) and zero or more *other* (inflected) *forms*, each holding *written representations* accounting for different orthographies (e.g. color vs colour). OntoLex-Lemon relies on third-party linguistic ontologies for a vocabulary of linguistic annotations and their values: in the example, we use LexInfo (Cimiano et al., 2011) to differentiate between *singular* and *plural number*. The example lexical entry is a *multi-word expression*, which has two *constituents*, corresponding to the words "comics" and "character". These *constituents* are *components*, holding additional information related to a particular use of the lexical entry. The properties rdf:_1 and rdf:_2 (in general, rdf:_N) are used to express the order of the *constituents*. OntoLex-Lemon defines the meaning of a lexical entry by connecting it to an ontology concept. This association can be reified as a *sense* object, in order to further qualify that association (e.g. register, usage, etc.). Our example word *denotes* a class in the DBpedia ontology, which has one semantic argument (synsem:isA). Its *syntactic behavior* is a *noun predicate frame*, meaning that the lexical entry can occur in stereotypical contexts like "X is a comics character" or "the comics character is X", where X is the *copulative argument*. The *synsem* vocabulary binds the syntactic and semantic arguments, by unifying them (i.e. use one RDF resource to identify both). Cimiano, Unger, and McCrae (2014) discussed the use of ontology lexicons for natural language generation and interpretation with respect to ontologies. Assuming another lexical entry for dbr:Superman, our example lexical entry allows to interpret "Superman is a comics character" as the triple dbr:Superman a dbr:ComicsCharacter. Beyond the representation of ontology lexicons, the model has been used to represent wordnets and other lexical resources. In this case, constructs such as WordNet's synsets are modeled as *lexical concepts*, while a *sense* is connected to these concepts via the property *is lexicalized sense of*. The *vartrans* module supports relations between lexical entries, lexical senses and lexical concepts. By using this module, one can say that a lexical entry is *translatable as* another (in some contexts), and that a sense has another sense as *translation*.

It is noteworthy that the OntoLex-Lemon model has diverse elements of complexity:

- reification: e.g. forms, lexical entries, etc.;
- indirection: the written representation of a form of a lexical entry, or the syntax-semantic interface realized via argument unification;
- integrity conditions (not expressible in OWL): e.g. the lexical entries should be expressed in the language associated with the lexicon.

4. Facilities for OntoLex-Lemon

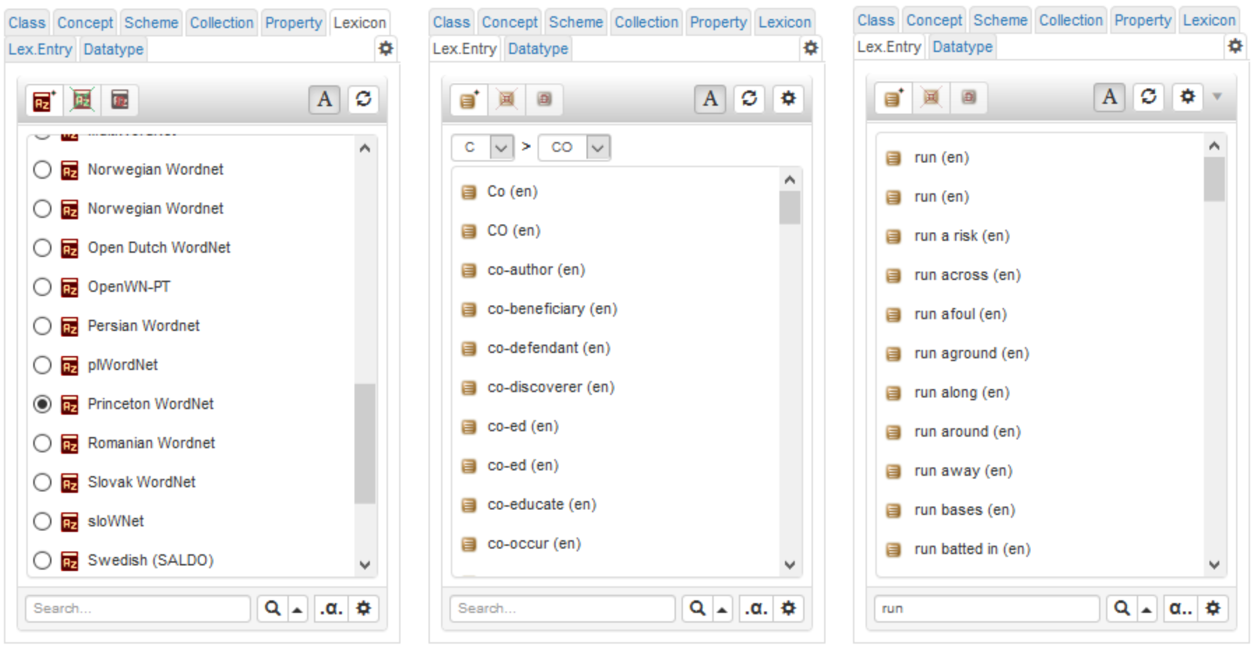
In the following sections, we will describe the facilities that were added to VocBench 3 to address the major challenges associated with editing OntoLex-Lemon.

4.1 Lexicon and Lexical Entry Management

In a typical ontology editor, lexicons can be found in the instance list associated with the class tree, by selecting the class lime:Lexicon. The description of a lexicon enumerates its lexical entries as the values of the property lime:entry. This approach lacks abstraction over the model and mixes the domain model and the modeling vocabularies (e.g. OntoLex-Lemon) inside the class tree. Lexical entries could also be inspected as instances of the class ontolex:LexicalEntry, without their repartition across lexicons. Additionally, if the editor is not sufficiently flexible, lexicons and, especially, lexical entries might not be rendered, and be shown as URIs (possibly shortened as qualified names). Therefore, an important extension

[8] https://github.com/jmccrae/lemon.patterns

[9] https://github.com/andreabellandi/LexO-lite



Norwegian Wordnet
Norwegian Wordnet
Open Dutch WordNet
OpenWN-PT
Persian Wordnet
plWordNet
Princeton WordNet
Romanian Wordnet
Slovak WordNet
sloWNet
Swedish (SALDO)

C > CO

Co (en)
CO (en)
co-author (en)
co-beneficiary (en)
co-defendant (en)
co-discoverer (en)
co-ed (en)
co-ed (en)
co-educate (en)
co-occur (en)

run

run (en)
run (en)
run a risk (en)
run across (en)
run afoul (en)
run aground (en)
run along (en)
run around (en)
run away (en)
run bases (en)
run batted in (en)

Figure 2: Lexicon list (on the left) and corresponding lexical entries indexed by (two-letter) prefixes (on the middle) or returned by a ("starts with") search (on the right). The system is managing the whole collection of 34 wordnets collected by Open Multilingual Wordnet (Bond & Paik, 2012)

consists in additional browsing panels (alongside the class tree, the property tree, etc.) to list the lexicons and the lexical entries (see Figure 2). The two panels are connected, since the latter only shows the content of the lexicon selected in the former (in figure: Princeton WordNet). These panels are associated with creation and deletion operations. When creating a lexicon, the user is requested to enter the lexicon's language, which is recorded through properties of the *lime* module. This metadata influences the creation of lexical entries for it, since it will be only possible to specify their canonical form in that language (or a variant thereof). Moreover, this constraint applies as well to the addition (via the resource view) of other forms or further representations of an existing form.

VocBench supports two approaches to list the lexical entries (see Figure 2), by either i) showing the entries with a common prefix of 1 or 2 letters, or ii) showing the results of a search over the lexical entries. The latter approach was introduced to overcome some scalability issues. For similar reasons, we complemented the traditional concept tree with a new search-based view, which shows a flat list of concepts found via search.

4.2 Concept Set and Lexical Concept Management

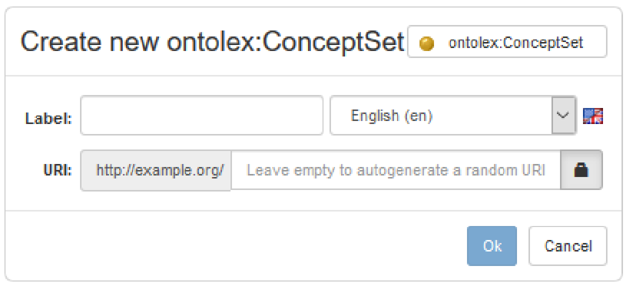
OntoLex-Lemon defines the notion of *concept set* as a collection of *lexical concepts*, declaring these classes as subclasses of skos:ConceptScheme and skos:Concept, respectively. VocBench 3 already had dedicated panels for the visualization of concept schemes and concepts, which were aware of possible sub-classes. Consequently, they could be reused unaltered in OntoLex-Lemon projects. However, in this context, the creation dialog for a new *concept scheme* (see Figure 3, observe in particular the top-right corner with the class modifier), resp. a new *concept*, is configured to create an ontolex:ConceptSet (instead of a skos:ConceptScheme), resp. an ontolex:LexicalConcept (instead of a skos:Concept).

4.3 OntoLex-Lemon Aware Rendering

McCrae et al. (2012) stressed the importance of a suitable rendering of the components of a lexicon. Accordingly, we extended VocBench 3 to display lexical entries via their canonical form. Furthermore, OntoLex-Lemon lexicalizations of ontologies and RDF datasets should be used to produce a human-friendly rendering of the lexicalized entities. To that end, we developed an implementation of the *Rendering Engine* extension point for OntoLex-Lemon. An intrusive modification of the system enabled the suggestion of this *rendering engine* for OntoLex-Lemon projects.

4.4 Dedicated Resource View Templates

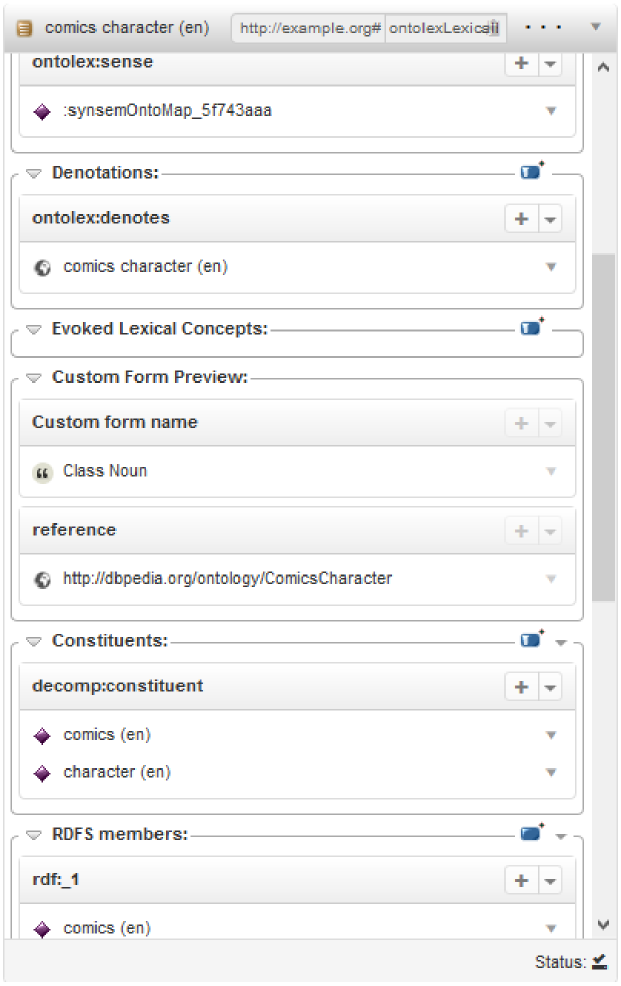
The *resource view* is a general data visualization panel displaying RDF resources in terms of their property values, divided into different sections organized by type of property (with a generic *properties* section for properties not failing in any category). For certain types of resources (e.g. classes, properties, concepts, etc.) the *resource view* has a dedicated *template* consisting of *elective sections*. These *elective sections* optimize the layout of each resource view with properties often associated to its managed type of resource, while a default *template* guarantees that the *resource view* is compatible with any resource. Accordingly, we defined dedicated templates for most of


Create new ontolex:ConceptSet ontolex:ConceptSet

Label: English (en)

URI: http://example.org/ Leave empty to autogenerate a random URI

Figure 3: Dialog for the creation of a concept set


comics character (en) http://example.org# ontolexLexical

ontolex:sense

:synsemOntoMap_5f743aaa

Denotations:

ontolex:denotes

comics character (en)

Evoked Lexical Concepts:

Custom Form Preview:

Custom form name

Class Noun

reference

http://dbpedia.org/ontology/ComicsCharacter

Constituents:

decomp:constituent

comics (en)

character (en)

RDFS members:

rdf:_1

comics (en)

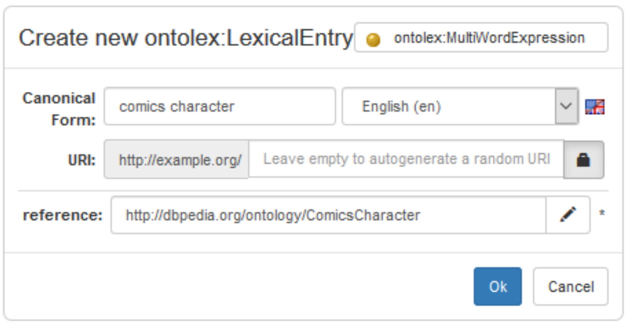
Status:

Figure 4: Resource view on the example lexical entry “comics character”

the entities defined by OntoLex-Lemon. Figure 4 shows (part of) the *resource view* displaying our example lexical entry “comics character” (see Section 3). It consists of the following sections: *types*, *lexical forms*, *lexical senses*, *denotations*, *evoked lexical concepts*, *custom form preview*, *constituents*, *RDFS members* and other *properties*. The meaning of most sections simply follows from the associated property of OntoLex-Lemon. Nonetheless, some of these sections will be discussed further in this paper.

4.5 Support for Decomposition

In our introductory example, the *lexical entry* “comics character” (see Figure 1) is connected to its tokens through the property decomp:constituent, as well as through the properties rdf:_N (required to encode the order of the tokens). In the resource view (see Figure 4) these properties are separated (obeying to a triple-oriented perspective), nonetheless the values of the property decomp:constituent are ordered based on the information provided by the RDF membership properties. Following McCrae et al. (2012), we provided a dedicated editor to manipulate the sequence of tokens as a whole, while the system takes care of low-level triple updates.


Create new ontolex:LexicalEntry ontolex:MultiWordExpression

Canonical Form: comics character English (en)

URI: http://example.org/ Leave empty to autogenerate a random URI

reference: http://dbpedia.org/ontology/ComicsCharacter *

Ok Cancel

Figure 5: Dialog for the creation of the class noun “comics character”

4.6 Support for Redundant Patterns

In OntoLex-Lemon, the same relationship can be encoded in redundant ways. For example, the binding of a *lexical entry* to an ontology concept can be expressed as a simple triple (in either direction) or reified via a *sense* resource. Our policy about these redundant patterns is as follows: upon creation, ask the user whether to create other variants as well, while the system deletes every variant (without asking the user). A property and its inverse are asserted by VocBench, unless one of the arguments is not locally defined (e.g. a third-party lexicon is developed for an existing ontology, so the references of the senses are really just mentions of external resources). In these cases, VocBench does not generate triples having mentioned resources as subjects.

4.7 Custom Forms for Ontology-Lexicon Design Patterns

The lexical entry in Figure 1 requires the assertion of dozens of triples about different subjects and, occasionally, subtly related (e.g. common object expressing the binding of syntactic and semantic arguments). However, it can be represented succinctly with the language of design patterns for ontology-lexicons (McCrae & Unger, 2014) as follows:

ClassNoun(“comics character”, dbo:ComicsCharacter) with plural “comics character”

We implemented[10] most patterns as VocBench *custom forms* (already described in Fiorelli et al. (2017)), which enable to specify: i) the transformation of user input into RDF nodes, ii) a graph template to be instantiated with these nodes. These *forms* can be attached to properties (i.e. *custom ranges*) or to classes (i.e. *custom constructors*). Specifically, our custom forms for ontology-lexicons are *constructors* for the class ontolex:LexicalEntry. When users create a lexical entry, they can select a custom constructor, which augments the creation dialog with custom fields. Figure 5 depicts the dialog for a class noun, in which standard fields for the *canonical form* and the selection of the entry type (e.g. *multiword expression*) are complemented (below a horizontal line) by a custom field for the *reference* class of the entry being created. The pictured form is filled with the information to generate the lexical entry “comics character” in Figure 1 (minus its decomposition into tokens).

In addition to simplifying the construction of complex resources, *custom constructors* ease the comprehension of

[10] https://bitbucket.org/art-uniroma2/lemon-vb-customforms/

data, as well. When computing the resource view, VocBench considers the custom constructors associated with any class of the resource, and then it identifies the one that best explains the data (i.e. its graph pattern matches the most triples): a *custom form preview* section is added to the *resource view* (see Figure 4), showing the name of the *custom form* and the values bound to the *form* variables.

4.8 Search Aware of OntoLex-Lemon

Full-text search in VocBench 3 attempts to match *lexicalizations* of resources expressed through some *lexicalization model*, chosen project-wide. The introduction of OntoLex-Lemon as an additional *lexicalization model* required an extension of this mechanism, so that a resource can be found via the *written representation* of the lexical entries associated with it. Differently from other *lexicalization models*, lexical entries are first-class citizens of a lexicon, as they can be related to each other and reused in different *lexicalizations*. Therefore, we extended the search to make it able to find lexical entries. The *advanced search* and the *custom search* capabilities of VocBench support, respectively, the specification of more complex search criteria (e.g. in terms of different property values) and the use of custom search dialogs (powered by saved SPARQL queries).

4.9 LIME Metadata Exporter

In Section 4.1, we explained that VocBench 3 enforces some integrity constraints based on certain *lime* metadata (e.g. the language of a lexicon). Actually, *lime* supports a much richer description of lexicons, concept sets, and how these are related to each other or to ontologies/RDF datasets. This description combines descriptive metadata (usually entered by hand) and statistics (best computed automatically). VocBench 3 addressed this kind of use case with its *Dataset Metadata Exporter* extension point, which can be implemented for different metadata vocabularies. As part of this work, we developed an implementation for *lime*, which uses our Lime API (Fiorelli, Pazienza, & Stellato, 2017) to compute several statistics. Figure 6 depicts the form that can be filled with descriptive metadata (persisted across sessions); upon export, this information is combined with the computed statistics. Fiorelli et al. (2019) showed the utility of *lime* metadata for automatic and robust configuration of semantic mediation processes.

5. Evaluation

In the following sections, we report on how we evaluated the quality of our system by means of an ensemble of approaches.

5.1 Conformance to OntoLex-Lemon

In this section, conformance is loosely intended as the degree to which VocBench supports different parts of OntoLex-Lemon through a convenient combination of general editing capabilities and dedicated extensions. Therefore, it should be understood as a usability review carried on by the developers of the systems[11].

5.1.1 Core Module

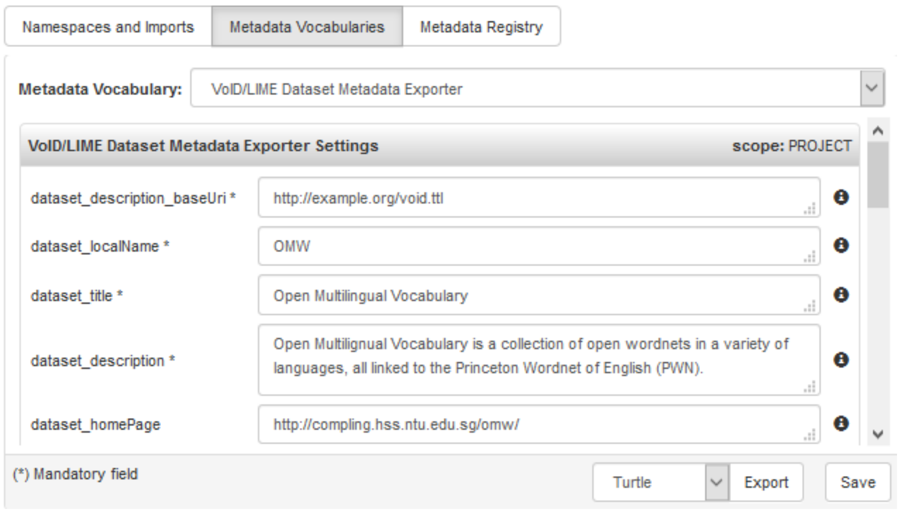
Additional panels or customizations of existing ones support listing and browsing both lexicons and concept sets. The creation of a lexical entry together with its canonical form and the creation of a lexical form together with its written representation are both handled as macro-operations. We managed the different ways of creating a lexicalization (i.e. plain triple vs sense), while the management of conceptualizations relating lexical entries and lexical concepts will be completed in a forthcoming release. VocBench does not render lexical senses by default; however, a *custom form* for the property ontolex:sense supports the rendering via a property path and the form-based preview.

5.1.2 Syntax and Semantics Module

The correspondence between syntactic and semantic arguments may be edited via triple-level operations. However, the *custom forms* implementing the design patterns for ontology-lexicons ease the comprehension and editing of this module in a specific use case.

5.1.3 Decomposition Module

We addressed the tokenization of lexical entries and the subterm relation. Their phrase structure should be edited triple by triple, and it can't be displayed as a whole: nesting of resource views would be a partial solution.



Metadata Vocabulary: VoID/LIME Dataset Metadata Exporter

VoID/LIME Dataset Metadata Exporter Settings   scope: PROJECT

dataset_description_baseUri *	http://example.org/void.ttl
dataset_localName *	OMW
dataset_title *	Open Multilingual Vocabulary
dataset_description *	Open Multilignual Vocabulary is a collection of open wordnets in a variety of languages, all linked to the Princeton Wordnet of English (PWN).
dataset_homePage	http://compling.hss.ntu.edu.sg/omw/

(*) Mandatory field

Figure 6: Dialog for the generation of Lime metadata

[11] http://www.uxforthemasses.com/usability-reviews/

5.1.4 Variation and Translation Module

The corresponding ontology is loaded implicitly in OntoLex-Lemon datasets, so that standard triple-level operations can be used to edit relationships expressed through simple triples. To improve the user experience, we provide a dedicated selector of lexical senses. However, we do not support reified relationships, since there is no dedicated browser nor are they reported in the resource view of the related entities (unless these entities contain in turn a link to the reified relationship).

5.1.5 Lime Module

Metadata are checked to forbid user requests that violate some integrity constraints (e.g. lexical entries should be in the language of the lexicon). Additionally, we support the interpolation of hand-written metadata and automatically computed statistics to produce a dataset description.

5.2 Support for Design Patterns for Ontology Lexicons

We support 11 of the 15 patterns that are listed in the catalog to date. In addition, we have a partial support for *ConsequenceVerbs*. For comparison, Lemonade only supports 3 patterns (all supported by our implementation). In Section 4.7, we showed that these patterns ease the development of a lexical entry, by substantially reducing the number of explicit actions that the user shall carry on. Consequently, our support of different design patterns correlates with the usability of the system.

We implemented the full form of the patterns, although some of them (e.g. *StateVerbs*) offer abbreviations based on default values. In fact, our implementation introduces some approximations. Firstly, we use generic syntactic frames (e.g. lexinfo:VerbFrame), while the reference implementation tries to compute the most specific frame based on the provided arguments (e.g. lexinfo:TransitiveFrame). Secondly, our forms always assign a POS tag to the lexical entry (e.g. lexinfo:commonNoun), while in the reference implementation multiword expressions are given a phrase type (e.g. lexinfo:NounPhrase). The grammar of the pattern language includes productions such as ⟨arg⟩ → Subject | … | PrepositionalObject (⟨string⟩). In the language of custom forms, we flatten this definition: we have adjacent fields for the argument and the optional accompanying string. Regarding the patterns that we do not support (*RelationalMultivalentNoun*, *EventVerb*, *ScalarAdjective*), the root cause is the inability to represent a variable number of fields in the form. Evaluating the support for the design patterns, we ignored tokenization, other (inflected) forms and linguistic annotation, already supported by VocBench.

5.3 Comparison with Related Work

In this section, we compare our OntoLex-Lemon editor to related systems. Because of their limited availability and high heterogeneity, we couldn't perform a competitive usability testing nor an all-to-all-comparison. Instead, we decided to compare our work with each of them from the viewpoint of functionality, in order to assess qualitatively their relative usability and utility[12].

5.3.1 Lemon Source

This system is compatible with Monnet *lemon*, and consequently it doesn't support lexical concepts introduced in OntoLex-Lemon. Lexico-semantic resources are thus seen as a special case of ontology lexicons, rather than as first-class citizens like in VocBench 3. Lemon Source explicitly addresses the rendering of certain model artifacts (e.g. senses, parse trees, etc.) and the manipulation of complex graph patterns as a whole (e.g. subcategorization frames). Additionally, it bootstraps the creation of lexicons from the mere labels that can be found in most ontologies, by applying a (configurable) pipeline of NLP tools (e.g. a parser to compute the parse tree of a lexical entry) and reusing lexical entries in existing resources (e.g. Princeton WordNet). Currently, VocBench 3 lacks these NLP capabilities, but it features extensive support for cross-project linking and Linked Data exploitation (however, necessary extensions of these capabilities are still under development). Since automatically generated entries may be unsuitable for publication, Lemon Source includes a workflow mechanism that enables to validate these entries. Following the paradigm of wikis, Lemon Source supports tracking the history of pages. VocBench 3 supports both functionalities as optional addons, respectively, history and validation. Both Lemon Source and VocBench support multiple projects, multiple users and access control.

5.3.2 Lemonade

Lemonade integrated Lemon Assistant (the web frontend) and LEIRE (a linter), currently available as unintegrated services[13,14]. The goal was to avoid or catching errors, respectively using a higher-level abstraction (i.e. the design patterns) and through the linter. Nowadays, the frontend is a wizard to create individual lexical entries in the language of design patterns. Therefore, search and browsing capabilities are irrelevant. Lemon Assistant only supports class nouns, state verbs and relational nouns. Indeed, VocBench supports all of them and much more (see Section 4.7). Let us compare in greater details the wizards of Lemon Assistant to our custom forms. We will refer to relational nouns (structurally identical to state verbs), which are the most complex. The wizard has a field for the lemma and an optional field for the plural form. In VocBench, the field for the lemma is provided by the core creation dialog, while fields for other forms are not included (to be language independent). However, our custom forms can be extended with these extra fields. A relational noun denotes a property: in Lemon Assistant it is searched (with autocompletion) inside a preloaded ontology, while VocBench supports both search-based and browsing-based selection. Unfortunately, custom forms can't constraint the chosen reference to be a property. Relational nouns have two syntactic and semantic arguments, which can be bound either linearly (i.e. subject-subject) or reversely (i.e. subject-object). Lemon Assistant provides a dropdown menu to select either mapping. Custom forms do not support such menus, so we developed two separate forms. This approach is less usable, requiring that users figure out beforehand which mapping will be suitable, and in case of mistake forcing them back to the selection of the appropriate pattern. As an alternative, we developed a combined form, in which the user must

[12] https://www.nngroup.com/articles/usability-101-introduction-to-usability/

[13] http://lemonadetools.linkeddata.es/lemonAssistant/

[14] http://lemonadetools.linkeddata.es/leire/

indicate the syntactic argument (as an RDF property) bound to each semantic argument. In fact, our form is potentially more general than Lemon Assistant, which assumes that the syntactic arguments are a *copulative subject* and a *possessive adjunct*, or a *prepositional object* if a preposition is indicated. The drawback of our approach is the possibility to provide inconsistent information, such as using a preposition with arguments that do not expect it. With respect to Lemon Assistant, our forms also support domain and range restrictions, e.g. the word "son" denotes the property "dbo:child", if the object is male. Furthermore, Lemon Assistant doesn't support multiword expressions, and it is limited to English, Spanish and German. This limitation stems from its ability to generate usage examples of a lexical entry, which are useful to catch errors in the specification of a lexical entry (e.g. wrong selection between linear and reverse mapping). This feature is not supported by VocBench 3. Overall, custom forms in VocBench 3 are more general and they have been used to implement more design patterns. They are not only used to create lexical entries, but also to analyze and interpret RDF data. On the other side, Lemon Assistant is slightly more usable, because of (easily fixable) missing constraints in our user interface and (more difficult to implement) lack of usage examples in our custom forms. However, the utility of Lemon Assistant is limited to a specific application of *lemon*, because it is not a general *lemon* editor. Hence, it doesn't support seemingly simple tasks, such as specifying relationships between lexical entries or between senses.

5.3.3 LexO

LexO is the most advanced from the linguistics viewpoint, even though support for different phenomena is scattered among different, specialized versions of the system. We compared VocBench to the demo site indicated in the presentation page of the system[15]. LexO supports browsing lemmas, (other) forms, senses and the ontology. The first three panels are flat lists, which can be filtered via a search constraint (either *starts with* or *contains*): they are similar to the search-based mode of VocBench. The ontology panel is just a class hierarchy with options to create, delete or rename classes. It is not possible to edit the details of classes, nor is it possible to create properties or instances. Conversely, VocBench features a much more comprehensive support for ontology, thesaurus and general RDF editing. When the user selects a lemma, a form or a sense, the system opens an editing panel on the right-hand side, showing (in a single page) the lemma, the forms and the senses. It is possible to add multiple forms, distinguished by language annotations (e.g. number or gender), as well as multiple senses. A sense can contain a definition, and relations to other senses. These actions are also supported in VocBench. Furthermore, VocBench supports the use of unanticipated properties (simply importing or specifying their definition). It seems to be possible to add comments on forms and senses, but this capability was not enabled in the demo. Like Lemon Source, it is possible to flag a lemma as validated. LexO supports multi-word expressions and, applying some basic NLP capabilities (e.g. tokenization), suggests the components in the decomposition. Conversely, in VocBench, decomposition should be specified manually. LexO also provides a read-only "dictionary view" like a page in a printed dictionary. The advanced search in LexO is subsumed by the one in VocBench, but LexO also supports a diachronic search depending on a dedicated extension of *lemon*. LexO sidesteps problems with rendering and search by generating the identifier of lexical entries, forms and senses from the actual words.

The documentation of the still unreleased LexO-lite (see Section 2) indicates the support for most of OntoLex-Lemon and other improvements.

6. Conclusions

We presented an extension of VocBench 3 addressing most issues associated with the use of OntoLex-Lemon inside generic data-driven editors. The evaluation was positive, but it also revealed some shortcomings to be addressed in future. Interestingly, these are related to general improvements of VocBench (e.g. improve custom forms), showing that our extensions heavily depend on the mechanisms provided by VocBench rather being an entirely separated addition to the system, thus supporting the importance of a unified platform for RDF management.

While, due to its recent release, we cannot provide extensive evidence of the impact of the OntoLex extension for VocBench, there are a few factors that position this extension as a corner stone in the development of lexicons and ontology-lexicon interfaces. Firstly, VocBench is nowadays widely adopted: the Publications Office of the European Commission provides an instance hosting more than 50 projects belonging to several Directorate Generals (DGs) within the commission itself, while diverse organizations belonging to United Nations, various member states of the EU, other organizations from China and US, are adopting VocBench for managing their linked open data; not to count the hundreds of users adopting it in private/academic/commercial settings. Secondly, while born as a sort of niche in the Semantic Web universe, the LLOD is gathering more interest and growing, with several resources being ported to OntoLex and linked to the cloud. While the initial effort has been focused on recovering existing resources by lifting their content to OntoLex (in order to "bootstrap" the cloud), the attention is now switching towards means to author new resources. Additionally, while OntoLex implicitly required to develop a model to represent lexical resources, its original objective remains to be a vocabulary for linking lexicons (or, in general, advanced lexical descriptions) to ontologies, a role for which VocBench can provide an extensive support that can be hardly matched by other systems, bringing together ontology, thesaurus and lexicon development in one solution. Finally, as remarked in sections 2 and 5.3, VB3 is the only stable and developed system for developing lexicons and for advanced lexical enriching of ontologies.

7. Acknowledgements

This work has been supported by the PMKI project, under the 2016.16 action of the ISA^2 Programme (https://ec.europa.eu/isa2/). ISA^2 is a programme of the European Commission for the modernization of public administrations in Europe through the eGovernment solutions.

[15] http://licolab.ilc.cnr.it/index.php/en/software-and-demo/

8. Bibliographical References

Bellandi, A., Giovannetti, E., & Weingart, A. (2018). Multilingual and Multiword Phenomena in a lemon Old Occitan Medico-Botanical Lexicon. Information, 9(3). doi:10.3390/info9030052

Bellandi, A., Giovannetti, E., Piccini, S., & Weingart, A. (2017). Developing LexO: A Collaborative Editor of Multilingual Lexica and Termino-ontological Resources in the Humanities. Proceedings of Language, Ontology, Terminology and Knowledge Structures Workshop (LOTKS 2017),co-located with the 12th International Conference on Computational Semantics (IWCS), 19 September 2017 Montpellier. Retrieved from http://www.aclweb.org/anthology/W17-7010

Bond, F., & Paik, K. (2012). A survey of wordnets and their licenses. Proceedings of the 6th Global WordNet Conference (GWC 2012). Matsue, Japan, January, 9-13, 2012, (pp. 64-71).

Chiarcos, C., Nordhoff, S., & Hellmann, S. (Eds.). (2012). Linked Data in Linguistics. Springer.

Cimiano, P., Buitelaar, P., McCrae, J., & Sintek, M. (2011). LexInfo: A declarative model for the lexicon-ontology interface. Web Semantics: Science, Services and Agents on the World Wide Web , 9(1), 29-51. Retrieved from http://www.sciencedirect.com/science/article/pii/S1570826810000892

Cimiano, P., McCrae, J. P., & Buitelaar, P. (2016). Lexicon Model for Ontologies: Community Report, 10 May 2016. Community Report, W3C. Retrieved from https://www.w3.org/2016/05/ontolex/

Cimiano, P., Unger, C., & McCrae, J. (2014). Ontology-Based Interpretation of Natural Language. Synthesis Lectures on Human Language Technologies, 7(2), 1-178. Retrieved from http://dx.doi.org/10.2200/S00561ED1V01Y201401HLT024

Fiorelli, M., Lorenzetti, T., Pazienza, M. T., & Stellato, A. (2017). Assessing VocBench Custom Forms in Supporting Editing of Lemon Datasets. In J. Gracia, F. Bond, J. P. McCrae, P. Buitelaar, C. Chiarcos, & S. Hellmann (Eds.), Language, Data, and Knowledge (Lecture Notes in Artificial Intelligence) (Vol. 10318, pp. 237-252). Springer, Cham. doi:10.1007/978-3-319-59888-8_21

Fiorelli, M., Pazienza, M. T., & Stellato, A. (2017). An API for OntoLex LIME datasets. OntoLex-2017 1st Workshop on the OntoLex Model (co-located with LDK-2017). Galway.

Fiorelli, M., Stellato, A., Lorenzetti, T., Schmitz, P., Francesconi, E., Hajlaoui, N., & Batouche, B. (2019). Metadata-driven Semantic Coordination. In E. Garoufallou, F. Fallucchi, & E. William De Luca (Eds.), Metadata and Semantic Research (Communications in Computer and Information Science) (Vol. 1057). Springer.

Fiorelli, M., Stellato, A., Lorenzetti, T., Turbati, A., Schmitz, P., Francesconi, E., . . . Batouche, B. (2018). Towards OntoLex-Lemon editing in VocBench 3. AIDAinformazioni(special issue).

Fiorelli, M., Stellato, A., Mccrae, J. P., Cimiano, P., & Pazienza, M. T. (2015). LIME: the Metadata Module for OntoLex. In F. Gandon, M. Sabou, H. Sack, C. d'Amato, P. Cudré-Mauroux, & A. Zimmermann (Eds.), The Semantic Web. Latest Advances and New Domains (Lecture Notes in Computer Science) (Vol. 9088, pp. 321-336). Springer International Publishing. doi:10.1007/978-3-319-18818-8_20

Frischmuth, P., Martin, M., Tramp, S., Riechert, T., & Auer, S. (2015). OntoWiki – An authoring, publication and visualization interface for the Data Web. Semantic Web, 6(3), 215-240. doi:10.3233/SW-140145

Jupp, S., Bechhofer, S., & Stevens, R. (2009). A Flexible API and Editor for SKOS. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, . . . E. Simperl (Eds.), The Semantic Web: Research and Applications (Lecture Notes in Computer Science) (Vol. 5554, pp. 506-520). Springer, Berlin, Heidelberg. doi:10.1007/978-3-642-02121-3_38

McCrae, J. P., & Unger, C. (2014). Design Patterns for Engineering the Ontology-Lexicon Interface. In P. Buitelaar, & P. Cimiano (Eds.), Towards the Multilingual Semantic Web (pp. 15-30). Springer Berlin Heidelberg. doi:10.1007/978-3-662-43585-4_2

McCrae, J. P., Bosque-Gil, J., Gracia, J., Buitelaar, P., & Cimiano, P. (2017). The OntoLex-Lemon Model: Development and Applications. In I. Kosem, C. Tiberius, M. Jakubíček, J. Kallas, S. Krek, & V. Baisa (Ed.), Electronic lexicography in the 21st century. Proceedings of eLex 2017 conference., (pp. 587-597).

McCrae, J., Montiel-Ponsoda, E., & Cimiano, P. (2012). Collaborative semantic editing of linked data lexica. In N. Calzolari, K. Choukri, T. Declerck, Doğan, M. Uğur, B. Maegaard, . . . S. Piperidis (Ed.), Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12). Istanbul Lüfti Kirdar Convention & Exhibition Centre, Turkey, 21-27 May 2012, (pp. 2619-2625). Retrieved from http://www.lrec-conf.org/proceedings/lrec2012/pdf/544_Paper.pdf

McCrae, J., Spohr, D., & Cimiano, P. (2011). Linking Lexical Resources and Ontologies on the Semantic Web with Lemon. In The Semantic Web: Research and Applications (Lecture Notes in Computer Science) (Vol. 6643, pp. 245-259). Springer Berlin Heidelberg. doi:10.1007/978-3-642-21034-1_17

Pazienza, M. T., Scarpato, N., Stellato, A., & Turbati, A. (2012). Semantic Turkey: A Browser-Integrated Environment for Knowledge Acquisition and Management. Semantic Web Journal, 3(3), 279-292. doi:10.3233/SW-2011-0033

Rico, M., & Unger, C. (2015). Lemonade: A Web Assistant for Creating and Debugging Ontology Lexica. In Natural Language Processing and Information Systems (Lecture Notes in Computer Science) (Vol. 9103, pp. 448-452). Springer, Cham.

Stellato, A., Fiorelli, M., Turbati, A., Lorenzetti, T., van Gemert, W., Dechandon, D., . . . Costetchi, E. (2019). VocBench 3: a Collaborative Semantic Web Editor for Ontologies, Thesauri and Lexicons. Semantic Web.

Stellato, A., Turbati, A., Fiorelli, M., Lorenzetti, T., Costetchi, E., Laaboudi, C., . . . Keizer, J. (2017). Towards VocBench 3: Pushing Collaborative Development of Thesauri and Ontologies Further Beyond. In P. Mayr, D. Tudhope, K. Golub, C. Wartena, & E. W. De Luca (Ed.), 17th European Networked Knowledge Organization Systems (NKOS) Workshop. Thessaloniki, Greece, September 21st, 2017, (pp. 39-52). Retrieved from http://ceur-ws.org/Vol-1937/paper4.pdf