Verbalising OWL ontologies in isiZulu with Python

C. Maria Keet[1], Musa Xakaza[1], and Langa Khumalo[2]

[1] Department of Computer Science, University of Cape Town, South Africa
mkeet@cs.uct.ac.za, XKZMUS001@myuct.ac.za
[2] Linguistics Program, University of KwaZulu-Natal, South Africa
khumalol@ukzn.ac.za

Abstract. Ontologies as a component of Semantic Web technologies are used in Sub-Saharan Africa mainly as part of ontology-driven information systems that may include an interface in a local language. IsiZulu is one such local language, which is spoken by about 23 million people in South Africa, and for which verbalisation patterns to verbalise an ontology exist. We have implemented the algorithms corresponding to these patterns in Python so as to link it most easily to the various technologies that use ontologies and for other NLP tasks. This was linked to Owlready, a new Python-based OWL API, so as to verbalise an ontology in isiZulu. The verbaliser can run in 'ontology inside' mode, outputting the sentences in the terminal for further processing in an ontology-driven information system, and in GUI mode that displays colour-coded natural language sentences for users such as domain experts and linguists. The demo will showcase its features.

1 Introduction

The use of Semantic Web technologies in Sub-Saharan Africa focuses predominantly on ontology-driven information systems. Examples include the integration of flower-visiting biodiversity data from natural history museums [3], agriculture and health data in Senegal [9], e-government monitoring for development projects [5], learning platforms [4], and localisation of OpenMRS[3] that uses the medical ontology SNOMED CT [11]. While some existing ontologies obviously can be reused, others are being developed to represent the knowledge more relevant for the region, and several existing ontologies would benefit from localisation. For instance, with a localised SNOMED CT and OpenMRS, one should be able to use it to generate patient discharge notes in an indigenous language.

To accommodate the varied use of ontologies especially for ontology-mediated natural language interfaces, an OWL verbaliser was developed for isiZulu, which is one of the 11 official languages in South Africa and the most popular language by first language speakers. This was based on the isiZulu verbalisation patterns and algorithms presented in [6, 7] and the Python OWL API Owlready [8]. It is a

[3] https://www.transifex.com/openmrs/OpenMRS/

proof-of-concept verbaliser that shows it can be done, despite having a grammar that does neither fit in existing language annotation models nor in pre-existing verbalisers, and in such a way that the core linguistic knowledge as well as the data and technologies can be reused independently.

We will describe the system design and implementation, provide brief notes on evaluation and what an attendee may expect from the demo.

2 System Design and Implementation

Design considerations. Unlike ontology verbalisation for English that uses mostly a template-based approach to insert the vocabulary elements, for isiZulu, there are verbalisation *patterns* that take into account context, such as verb conjugation and the strings for the quantifiers (examples further below). Also, the verbaliser had to meet multiple use case scenarios. This made it unfeasible to implement it with one or more existing technologies. Importantly, the use cases focus on text generation in intelligent user interfaces and patient discharge notes from electronic health records, rather than the sole purpose of facilitating user interaction with the ontology (knowledge acquisition, validation, documentation). This means that the language and linguistics components have to be reusable across applications, rather than tailor-made to OWL. Noting that isiZulu (and related languages) are under-resourced, any 'most generic' design and technology possible was preferred, so that the few resources available can be reused, cf. highly specialised formats that still would need to be adapted to accommodate the grammar [2]. For instance, it is helpful to make a separate list of nouns with their respective noun class (there are 17 for isiZulu), rather than extending, e.g., *lemon* [10]: the noun class of the noun determines the surface realisation for universal and existential quantification, conjugation, and negation, yet that list of nouns with their respective noun class can be reused in morphological analysers and in computer-assisted language learning.
Architecture and implementation. The components of the verbaliser and their interaction are shown in Fig. 1. The *verbaliser algorithms* file consists of the algorithms for named class subsumption ($C \sqsubseteq D$) and disjointness ($C \sqsubseteq \neg D$), simple existential quantification in the 'all-some' pattern ($C \sqsubseteq \exists R.D$) and negation thereof ($C \sqsubseteq \neg\exists R.D$) and simple conjunction ($C \sqcap D$), based on the algorithms and patterns in [6] and extended with the patterns for part-whole relations [7]. Some of these patterns require the name of the OWL class—assumed to be in the singular, as by good design practices—to be pluralised, hence requiring the isiZulu *pluraliser* of [1]. The implementation of the algorithms in Python is such that the corresponding functions can be linked to a variety of source files as well as individual statements in the interpreter for quick generation of a single sentence. For instance, for some axiom of the pattern $C \sqsubseteq \exists R.D$, e.g., uSolwazi $\sqsubseteq \exists$fundisa.Isifundo, then the input would be

```
>>> exists_zu('uSolwazi','fundisa','isifundo')
```

that will instantly generate *Bonke oSolwazi bafundisa isifundo esisodwa* 'all professors teach at least one course'.

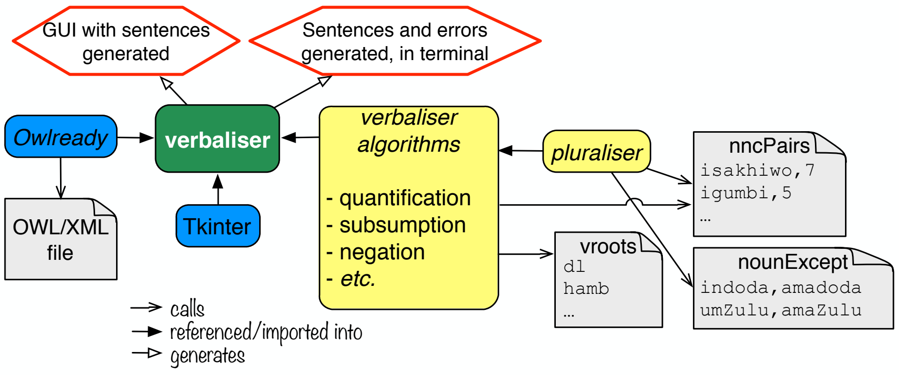


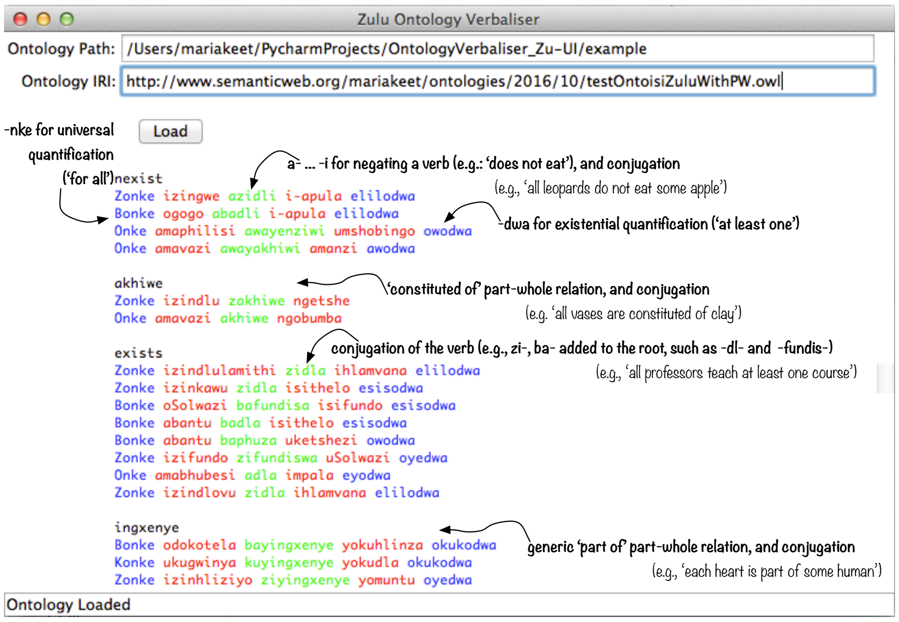
Fig. 1. Principal components of the OWL verbaliser.

To make it truly Semantic Web enabled, we have linked the verbalisation module to the novel OWL API for Python, Owlready [8], which works with OWL/XML serialisations. Using Owlready, the verbaliser fetches automatically the knowledge from the ontology and passes it on to the verbalisation module so as to compute the sentences and output the generated sentences in batch to the terminal for possible further processing. Consider, e.g., named class subsumption, whose serialisation in OWL/XML is `<SubClassOf> <Class IRI="..."/> <Class IRI="..."/> </SubClassOf>`, which is mapped to the `isa_zu(sub,super)` function in the .py file. For instance, the serialisation of impala $\sqsubseteq$ isilwane,

```
<SubClassOf>
    <Class IRI="http://www.example.org/ex.owl#impala"/>
    <Class IRI="http://www.example.org/ex.owl#isilwane"/>
</SubClassOf>
```

is fetched and passed on and processed as `isa_zu('impala','isilwane')` to generate *impala yisilwane* 'impala is an animal'. This holds likewise for the other supported types of axioms, with one category of exceptions: part-whole relations. There is no single string for the 'has part' object property in isiZulu. Therefore, a stub is used that is mapped to a specific part-whole relation and corresponding function; e.g., 'has portion' is realised with an object property named `eeee` in the OWL/XML file, which is mapped to the Python function `wp_solid_p(whole,part)` that, when used in an axiom, will generate the correct isiZulu surface realisation of 'has portion'. For instance, in shorthand notation, isinkwa $\sqsubseteq$ $\exists$eeee.ucezu_isinkwa generates, with the 'has portion' underlined, *Sonke isinkwa <u>sino</u>cezu lwesinkwa olulodwa* yet igazi $\sqsubseteq$ $\exists$eeee.isampula_igazi generates *Lonke igazi <u>line</u>sampula legazi elilodwa*, where the difference is due to the conjugation determined by the noun class (*si-* for the noun *isinkwa* in noun class 7 and *li-* for *igazi* in noun class 5) and phonological conditioning (*na-* + *ucezu* = *nocezu* and *na* + *isampula* = *nesampula*).

Although most ontology-driven information system use cases have an ontology 'in the background' rather than as end product for users, a GUI was deemed useful both for the common purpose of validation of an ontology's content as well as a better understanding of the sentence components from a language learning and linguistics viewpoint. To this end, the Python module Tkinter[4] was used to create a GUI with colour-coded elements. A screenshot of the GUI is shown in Fig. 2, which has been annotated for clarity.



```
Zulu Ontology Verbaliser
Ontology Path: /Users/mariakeet/PycharmProjects/OntologyVerbaliser_Zu-UI/example
Ontology IRI: http://www.semanticweb.org/mariakeet/ontologies/2016/10/testOntoisiZuluWithPW.owl

nexist
Zonke izingwe azidli i-apula elilodwa
Bonke ogogo abadli i-apula elilodwa
Onke amaphilisi awayenziwi umshobingo owodwa
Onke amavazi awayakhiwi amanzi awodwa

akhiwe
Zonke izindlu zakhiwe ngetshe
Onke amavazi akhiwe ngobumba

exists
Zonke izindlulamithi zidla ihlamvana elilodwa
Zonke izinkawu zidla isithelo esisodwa
Bonke oSolwazi bafundisa isifundo esisodwa
Bonke abantu badla isithelo esisodwa
Bonke abantu baphuza uketshezi owodwa
Zonke izifundo zifundiswa uSolwazi oyedwa
Onke amabhubesi adla impala eyodwa
Zonke izindlovu zidla ihlamvana elilodwa

ingxenye
Bonke odokotela bayingxenye yokuhlinza okukodwa
Konke ukugwinya kuyingxenye yokudla okukodwa
Zonke izinhliziyo ziyingxenye yomuntu oyedwa
Ontology Loaded
```

Load
-nke for universal quantification ('for all')
a- ... -i for negating a verb (e.g.: 'does not eat'), and conjugation
(e.g., 'all leopards do not eat some apple')
-dwa for existential quantification ('at least one')
'constituted of' part-whole relation, and conjugation
(e.g. 'all vases are constituted of clay')
conjugation of the verb (e.g., zi-, ba- added to the root, such as -dl- and -fundis-)
(e.g., 'all professors teach at least one course')
generic 'part of' part-whole relation, and conjugation
(e.g., 'each heart is part of some human')

Fig. 2. Section of the GUI interface of the Semantic Web-enabled isiZulu verbaliser. Colour coding: existential and universal quantification is shown in blue, the classes (nouns) in red, and the object properties and simple subsumption (verbs) in green.

Evaluation. Evaluation of the tool consisted of internal verification of correctness of encoding and external validation in the sense of testing it with more examples as described in [6], i.e., with more axioms. We represented in OWL all test cases of [6, 7], and the axioms used in their respective user evaluations, which were based on existing ontologies and selected to ensure coverage of permutations for noun classes, verbs, and part-whole relations. This totalled to 82 logical axioms. The tool, source code, sample ontology, and screencast video showing the working code are available from `http://www.meteck.org/files/geni/`.
Benefits of the chosen design and implementation The principal benefits are: 1) the ease with which the *verbaliser algorithms* file can be swapped for an analogous file in another language (e.g., isiXhosa, which is similar to isiZulu), 2) the reusability of the algorithms beyond OWL files when needed, and 3) the two

[4] `https://wiki.python.org/moin/TkInter`

modes of operation for users (GUI) and further processing in ontology-driven information systems (terminal output).

3 The Demo

The main aim of the demo is to present the functioning proof-of-concept OWL verbaliser. Given that isiZulu is not a familiar language to most people, an English-isiZulu dictionary will be available so that attendees can select terms and declare axioms that then will be verbalised on the fly. It is also an opportunity to discuss details of the implementation of the verbalisation patterns that present challenges to other existing OWL verbalisers and ontology editor tools.

Acknowledgments This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Number 93397).

References

1. Byamugisha, J., Keet, C.M., Khumalo, L.: Pluralising nouns in isiZulu and similar languages. In: Gelbkuh, A. (ed.) Proceedings of CICLing'16. p. in print. Springer (2016)
2. Chavula, C., Keet, C.M.: Is lemon sufficient for building multilingual ontologies for Bantu languages? In: Proc of OWLED'14. CEUR-WS, vol. 1265, pp. 61–72 (2014), Riva del Garda, Italy, Oct 17-18, 2014
3. Coetzer, W., Moodley, D., Gerber, A.: A case-study of ontology-driven semantic mediation of flower-visiting data from heterogeneous data-stores in three south african natural history collections. In: Semantics for Biodiversity (S4BioDiv'13) (2013), 27-5-2013, Montpellier, France
4. Dalvit, L., Gunzo, F.T., Maema, M.K.V., Slay, H.: Exploring the Use of Ontologies in Creating Learning Platforms: HIV and AIDS Education at a South African University. In: Proc. of ICCSSE'08. vol. 5, pp. 407–410 (Dec 2008)
5. Dombeu, J.V.F.: A conceptual ontology for e-government monitoring of development projects in sub saharan africa. In: IST-Africa 2010. pp. 1–8 (May 2010)
6. Keet, C.M., Khumalo, L.: Toward a knowledge-to-text controlled natural language of isiZulu. Language Resources and Evaluation in print (2016)
7. Keet, C.M., Khumalo, L.: On the verbalization patterns of part-whole relations in isiZulu. In: Proc. of INLG'16. pp. 174–183. ACL (2016), 5-8 September, 2016, Edinburgh, UK
8. Lamy, J.: Ontology-oriented programming for biomedical informatics. Studies in Health Technology and Informatics 221, 64–68 (2016)
9. Lo, M., Camamra, G., Niang, C.A.T., Ndiaye, S.M., Sall, O.: Towards an ontology-based framework for data integration: application to agriculture and health domains in Senegal. In: Gamatié, A. (ed.) Computing in Research and Development in Africa: Benefits, Trends, Challenges, and Solutions. pp. 41–57. Springer (2015)
10. McCrae, J., et al.: Interchanging lexical resources on the semantic web. Language Resources and Evaluation 46(4), 701–719 (2012)
11. SNOMED CT: (last accessed: 27-1-2012), `http://www.ihtsdo.org/snomed-ct/`