Toward an NLG System for Bantu languages: first steps with Runyankore (demo)

Joan Byamugisha and **C. Maria Keet** and **Brian DeRenzi**
Department of Computer Science, University of Cape Town, South Africa,
{jbyamugisha,mkeet,bderenzi}@cs.uct.ac.za

Abstract

There are many domain-specific and language-specific NLG systems, which are possibly adaptable across related domains and languages. The languages in the Bantu language family have their own set of features distinct from other major groups, which therefore severely limits the options to bootstrap an NLG system from existing ones. We present here our first proof-of-concept application for knowledge-to-text NLG as a plugin to the Protégé 5.x ontology development system, tailored to Runyankore, a Bantu language indigenous to Uganda. It comprises a basic annotation model for linguistic information such as noun class, an implementation of existing verbalisation rules and a CFG for verbs, and a basic interface for data entry.

1 Introduction

Natural Language Generation systems require content planning and format for the selected subject domain as input and specifics about the natural language in order to generate text (Staykova, 2014), of which the latter tend to be bootstrappable for related languages (de Oliveira and Sripada, 2014). Our NLG system uses ontologies to represent domain knowledge. As for language, we are interested in Runyankore, a Bantu language indigenous to south western Uganda. The highly agglutinative structure and complex verbal morphology of Runyankore make existing NLG systems based on templates inapplicable (Keet and Khumalo, 2017). There have been efforts undertaken to apply the grammar engine technique instead (Byamugisha et al., 2016a; Byamugisha et al., 2016b; Byamugisha et al., 2016c), which resulted in theoretical advances in verbalization rules for ontologies, pluralization of nouns, and verb conjugation that address the text generation needs for Runyankore. We present our implementation of these algorithms and required linguistic annotations as a Protégé 5.x plugin.

2 Linguistic Annotations for NLG

Most NLG systems for ontology verbalization require some annotations to the ontology's vocabulary so as to make the generated sentence sound more natural language-like. For instance, the *lemon* model for ontologies (McCrae and others, 2012). However, it has been shown to be insufficient for covering grammar constructs for Bantu languages, most notably due to the noun class system and complex morphological rules (Chavula and Keet, 2014). Other systems use tailor-made annotation schemata, e.g. (Androutsopoulos et al., 2013; Keet and Chirema, 2016). While they differ in number of linguistic annotation properties, what they share in common is the separation of annotation from ontology, as proposed in (Buitelaar et al., 2009), and storing that annotation in a separate XML file for further processing. We thus also annotated using an XML-based model, but limited our structure to our information of interest: NC, part-of-speech, and translation. The annotation functionality was implemented as a view tab in the Protégé 5.x plugin, so that it can be used during either multi-lingual or mono-lingual ontology development, or validation of the represented knowledge in an easily accessible way. The interface also ensures no typographical errors are made in the XML file. These annotation fields are mandatory, and we allowed for the use of 0 as the NC for the POS which is not a noun. These restrictions to input were achieved using document filters. The XML file is queried during the verbalization process so as to obtain the required annotations that are needed for the algorithms.

3 Implementation of the Grammar Engine

We implemented the algorithms for verbalization and pluralization presented in (Byamugisha et al., 2016a; Byamugisha et al., 2016c) as a Java application. The CFG specified in (Byamugisha et al., 2016b) was implemented using the CFG Java tool (Xu et al., 2011). We used this tool for three main reasons: our grammar engine implementation was done in Java, so we wanted a Java tool as well; we wanted a small CFG implementation for reasonable performance; and their tool extended Purdom's algorithm to fulfill Context-Dependent Rule Coverage (CDRC), which generates more and simpler sentences. A sample of the generated text is presented below:

- **Buri** rupapura rwamakuru **n'**ekihandiiko ekishohoziibwe, (generated from: Newspaper $\sqsubseteq$ Publication)
- **Buri** ntaama nerya ebinyaansi byo*ona*, (generated from: Sheep $\sqsubseteq \forall$ eats.Grass)

The generated text is saved in a text file, which ensures that the text can be linked to other application scenarios. We are working on a better design to present the sentences within the tool, for interaction during multi-modal ontology development. The grammar engine can be launched through the 'Runyankore>Verbalize' submenu under the 'Tools' menu in Protégé 5.x. The jar file is available from `https://github.com/runyankorenlg/RunyankoreNLGSystem`.

4 Conclusion

We briefly presented the core components of the Runyankore grammar engine Protégé 5.x plugin. It implements algorithms for verbalization patterns, noun pluralization, and verb conjugation. To make this work, the grammar engine requires linguistic information about each noun and verb (OWL class and object property) in the ontology in order to generate text. This linguistic information is stored in as separate XML file. The demo will show the working system and further details of the architecture.

Acknowledgements This work is based on the research supported by the Hasso Plattner Institute (HPI) Research School in CS4A at UCT and the National Research Foundation of South Africa (Grant Number 93397).

References

I. Androutsopoulos, G. Lampouras, and D. Galanis. 2013. Generating natural language descriptions from OWL ontologies: The NaturalOWL system. *JAIR*, 48:671–715.

P. Buitelaar, P. Cimiano, P. Haase, and M. Sintek. 2009. Towards linguistically grounded ontologies. In *ESWC'09, LNCS 5554*, pages 111–125.

J. Byamugisha, C. M. Keet, and B. DeRenzi. 2016a. Bootstrapping a Runyankore CNL from an isiZulu CNL. In *CNL 2016*, Aberdeen, Scotland. Springer LLNCS.

J. Byamugisha, C. M. Keet, and B. DeRenzi. 2016b. Tense and aspect in Runyankore using a context-free grammar. In *INLG 2016*, Edinburgh, Scotland.

J. Byamugisha, C. M. Keet, and L. Khumalo. 2016c. Pluralizing nouns in isiZulu and related languages. In *CICLing 2016*, Konya, Turkey.

C. Chavula and C. M. Keet. 2014. Is lemon sufficient for building multilingual ontologies for bantu languages? In *OWLED'14*, volume 1265, pages 61–72, Riva del Garda, Italy.

R. de Oliveira and S. Sripada. 2014. Adapting simplenlg for brazilian portuguese realisation. In *Proc of INLG'14*, pages 93–94. ACL.

C. M. Keet and T. Chirema. 2016. A model for verbalizing relations with roles in multiple languages. In *Poc. of EKAW'16*, volume 10024 of *LNAI*, pages 384–399, Bologna, Italy. Springer.

C. M. Keet and L. Khumalo. 2017. Towards a knowledge-to-text controlled natural language of isizulu. *LRE*, 51:131–157.

John McCrae et al. 2012. Interchanging lexical resources on the semantic web. *LRE*, 46(4):701–719.

K. Staykova. 2014. Natural language generation and semantic technologies. *Cybern. and Info. Tech.*, 14.

Z. Xu, L. Zheng, and H. Zhen. 2011. A toolkit for generating sentences from context-free grammars. *Int. J. of Software and Informatics*, 5:659–676.