UNIVERSITY OF CAPE TOWN

DOCTORAL THESIS

Ontology Verbalization in Agglutinating Bantu Languages: A Study of Runyankore and Its Generalizability

Author:
Joan BYAMUGISHA

Supervisor:
Assoc. Prof. C. Maria KEET
Dr. Brian DERENZI

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Computer Science Department

October 25, 2019

Declaration of Authorship

I, Joan BYAMUGISHA, declare that this thesis titled, "Ontology Verbalization in Agglutinating Bantu Languages: A Study of Runyankore and Its Generalizability" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed by candidate

Date:24-10-2019

Abstract

Joan BYAMUGISHA

Ontology Verbalization in Agglutinating Bantu Languages: A Study of Runyankore and Its Generalizability

Natural Language Generation (NLG) systems have been developed to generate text in multiple domains, including personalized patient information. However, their application is limited in Africa because they generate text in English, yet indigenous languages are still predominantly spoken throughout the continent, especially in rural areas. The existing healthcare NLG systems cannot be reused for Bantu languages due to the complex grammatical structure, nor can the generated text be used in machine translation systems for Bantu languages because they are computationally under-resourced. This research aimed to verbalize ontologies in agglutinating Bantu languages.

We had four research objectives: (1) noun pluralization and verb conjugation in Runyankore; (2) Runyankore verbalization patterns for the selected description logic constructors; (3) combining the pluralization, conjugation, and verbalization components to form a Runyankore grammar engine; and (4) generalizing the Runyankore and isiZulu approaches to ontology verbalization to other agglutinating Bantu languages. We used an approach that combines morphology with syntax and semantics to develop a noun pluralizer for Runyankore, and used Context-Free Grammars (CFGs) for verb conjugation. We developed verbalization algorithms for eight constructors in a description logic. We then combined these components into a grammar engine developed as a Protégé5X plugin. The investigation into generalizability used the bootstrap approach, and investigated bootstrapping for languages in the same language zone (intra-zone bootstrappability) and languages across language zones (inter-zone bootstrappability). We obtained verbalization patterns for Luganda and isiXhosa, in the same zones as Runyankore and isiZulu respectively, and chiShona, Kikuyu, and Kinyarwanda from different zones, and used the bootstrap metric that we developed to identify the most efficient source—target bootstrap pair. By regrouping Meinhof's noun class system we were able to eliminate non-determinism during computation, and this led to the development of a generic noun pluralizer. We also showed that CFGs can conjugate verbs in the five additional languages. Finally, we proposed the architecture for an API that could be used to generate text in agglutinating Bantu languages.

Our research provides a method for surface realization for an under-resourced and grammatically complex family of languages, Bantu languages. We leave the development of a complete NLG system based on the Runyankore grammar engine and of the API as areas for future work.

Acknowledgements

I owe a lot to many people and institutions for enabling me to undertake my PhD studies at the University of Cape Town (UCT). I would like to thank the Hasso Plattner Institute (HPI) for awarding me the funding that made my studies possible.

I am very grateful to the research guidance, support, and direction that I received from my supervisors, Assoc. Prof. Maria Keet and Dr. Brian DeRenzi. Thank you very much for opening my mind to what research really is, seeing the potential in me even when I could not, and accommodating my special needs throughout the duration of my PhD. Thanks to your efforts, I have obtained a great level of independence! Thank you very much!

I would also like to thank Prof. Hussein Suleman, the head of the Computer Science Department during my time at UCT, for taking a keen interest into how the activities at the department could be made accessible for me. I appreciated this greatly!

I extend a Lot of gratitude to the Disability Services department at UCT, whose staff and volunteers spent hours converting documents into accessible formats. To the late Reinette Popplestone, thank you very much for your friendship, guidance, and support throughout my studies. Knowing you as a person opened my mind to the positive impact that one can have on the world in spite of one's disability. How I wish you could have lived to see me graduate! To Denise and your awesome team of volunteers, thank you for taking the time and care to format documents for me, which, due to my area of research, sometimes required you to retype entire books—as was the case with the Runyankore dictionary. Without the resources you made accessible for me, my research journey would have been much much harder. Finally, to Edwina, Nafisa, and DJ, many thanks for always being accommodating and willing to assist.

My family has always been my strong foundation, on which I stand and dare to reach out to the world. You have all made direct inputs into this research, and I honestly do not know what I would ever do without you. To my parents, Francis and Goreth Byamugisha, thank you for trusting and supporting me as I left to start my studies at UCT. Thank you father for all the time spent translating texts, evaluating test results, and proof-reading results and identifying errors. To my mother, thank you for your input into the Luganda translations and in moderating the evaluation carried out in Mbarara. Your fluency in speaking and reading Runyankore provided me with assurance that the study participants understood what the study was about.

To my siblings, many thanks for the support and patience in assisting me, especially during times of urgency. Thank you Jean for proof-reading and printing out all the study materials for 100 study participants, and for your company and input into the activities of that day. Thanks Gilbert for always providing a listening ear whenever I wanted to vent out my frustrations or share my excitement, and needed the view of a fellow computer scientist. Thanks a lot Jema for helping me to draw the architecture diagrams, read inaccessible content to me over the phone for hours on end, and for those memorable long chats that I valued greatly. Thank you Julian for the crash-course in data analysis in Excel which enabled me to analyze my data, helping me through a very emotionally difficult time last year, and for the hours spent recording voice-notes of inaccessible documents.

I am also grateful to my labmates whose accepting and helpful nature enabled me to take part in all department activities, social and otherwise. I have grown very close to many of you, and formed long-lasting friendships. Special thanks go to Christine Wangiru Mburu, Juliet Nagawa, Naissa Mafoni, Ngoni Choga, and Zola Mahlaza for providing the Kikuyu, Luganda, Kinyarwanda, chiShona, and isiXhosa translations used in this research.

I am very grateful to the friendship and support of Michelle, Bennedict, Pumeza, Clare, Felina, Christa, and Consolata for everything crazy that we have done together. Special thanks go to the Botha and Richfield families for providing me with a home away from home in Cape Town.

Finally, I am grateful to God for all the blessings of love, support, and care that I have received throughout my life.

Contents

List of Figures

List of Tables

List of Abbreviations

CFG	Context-Free Grammar
FV	Final Vowel
NC	Noun Class
NCS	Noun Class System
Neg	Negation
NLG	Natural Language Generation
OWL	Web Ontology Language
POS	Part-Of-Speech
SC	Subject Concord
SVO	Subject Verb Object
TnA	Tense and/or Aspect
VC	Vowel Coalescence
VE	Vowel Elision
VH	Vowel Harmony
VR	Verb Root

This is dedicated to my family.

Chapter 1

Introduction

1.1 Background

Throughout the world today, patients generally receive information about their diagnosis, treatment, side effects and care verbally during the patient–doctor consultation. However, Di Marco et al. [48, 49] identified that patients consistently retain only a small fraction of the information after the consultation, and that the information which is not directly addressed to the patient is highly likely to be discounted or ignored, possibly resulting in improper compliance to medical instructions.

On the other hand, it has been found that personalized messages increase the likelihood for a patient to be more engaged, and to read, comprehend, and act upon the information received [52]. Such personalized information was found to enable patients to make more informed decisions about their prescribed treatment when they understood their medical condition [50, 52]. It was also noted to be of great value to the patient's friends and family in helping them to understand the patient's illness and care [48, 49].

Several studies in health communication have supported the importance of personalization of health information by showing that such information can be more effective if customized for the individual patient according to their specific disease, literacy level, dialect, and cultural and cognitive dispositions [52]. However, personalizing patient information results in the problem of customizing, where the differences among the characteristics of patients can easily be in the tens or hundreds of thousands [48, 49]. Despite this difficulty, the importance of presenting patient information as natural language instead of a structured patient file is that it is more understandable, as patients will want to receive information in a form that will enable them to understand their illness or disease, treatment options and health outcomes [52].

Natural Language Generation (NLG), the production of understandable texts in a selected human language from an underlying non-linguistic representation of information [22, 63, 160], has successfully been applied to the problem of producing personalized healthcare messages [31, 46, 53, 81, 117, 120]. Examples of these include systems that generate personalized diagnosis and treatment information for patients with cancer [31], mental health intervention messages [81], summaries for parents with babies in neonatal care [120], health education materials [51, 53], and drug prescription explanations [46].

However, all existing NLG systems for healthcare [31, 46, 53, 62, 74, 81, 117, 120] generate text in English, which limits their usefulness in African countries, where multiple indigenous languages are still predominantly spoken, especially in rural areas. Additionally, the lack of computational resources (such as labeled and unlabelled corpora and tools) for most Bantu languages limits the use of machine translation from the generated English text. Further, though medical professionals are expected to communicate in the languages of the areas where they work, few actually do [187]. In addition, given that the presentation of medical information in English exacerbates literacy difficulties already prevalent in situations of health [52], it is important to localize patient information. For most African countries, where patients still receive information about their diagnosis, treatment, side effects, and care verbally during the patient–doctor consultation,

there is a need to also provide written personalized patient information so as to improve patient compliance with prescribed treatments.

We therefore investigated how NLG can be used to generate text in the Bantu language family, spoken in 27 of the continent's 54 countries by about 240 million speakers, and estimated to comprise between 300 to 680 languages [146]. We limited our scope to drug prescription explanations because enabling patients to make more informed decisions about their prescribed treatments has been associated with promoting greater compliance, achieving better patient outcomes, and reduced healthcare costs [31, 52, 54, 77, 186].

Healthcare-based NLG systems rely on a patient's Electronic Health Record (EHR) [31, 46, 53, 62, 74, 81, 117, 120], and the entities in EHRs as well as the relations between them are very well modeled as ontologies (chief among them is SNOMED-CT [83]). We therefore took a knowledge-to-text approach to text generation, where Description Logic (DL) represented the knowledge in the ontology.

1.2 Problem Statement

Some of the existing NLG techniques cannot currently be used for Bantu languages. Templates [63, 160] were found to be inapplicable to the complex grammatical structure of Bantu languages [100], while statistical corpus-based and deep learning NLG approaches [13, 58, 110, 111, 185] require large amounts of data, yet most Bantu languages are computationally under-resourced.

Our first problem space thus centered around finding the most appropriate NLG technique for generating text in a Bantu language. We used a single Bantu language, Runyankore, to research this. Our next problem space was concerned with how to apply the knowledge gained from text generation in a single Bantu language to generate text in other languages in the same family.

1.2.1 Research Questions

The main objectives of this research were two fold: (1) research the use of a grammar engine to verbalize ontologies in Runyankore, and (2) investigate the generalizability of this text generation approach to other Bantu languages with an agglutinating morphology that results from words formed by adding affixes to the root word.

RQ1: How can noun pluralization and verb conjugation be achieved in Runyankore?

Entities in an ontology are expressed as concepts and roles. The names of DL concepts are usually denoted with a noun, and it has been shown that users prefer the plural in the verbalization of some of the axiom types [69, 93, 99]; for example, 'all doctors' instead of 'each doctor'. The DL roles in an ontology are usually labeled as verbs, thus requiring a means of conjugating the verb in the appropriate tense and aspect represented by the roles in the ontology. For example, *teaches.Course* is in the simple present tense, while *takenWith.Milk* involves the passive. For these reasons, we first investigated how to achieve noun pluralization and verb conjugation as prerequisites to verbalization in Runyankore. Unlike in English, noun pluralization in Runyankore is performed on the prefix not the suffix of the noun, and requires taking into account the class to which a noun belongs. Verb conjugation requires accounting for concordial agreement that is also based on the class of the noun, as well as agglutination and the placement of morphemes for tense, aspect, negation, and extension.

RQ2: What are the Runyankore verbalization patterns for the selected $\mathcal{ALCQ}$ constructors?

The degree to which an ontology can specify a conceptualization depends on the expressiveness of the logic used to represent a domain [71]. Given the richness of the healthcare domain, we

first needed to identify the appropriate formal language that is expressive enough for the relationships and constraints found in drug prescription explanations. We found the description logic Attributive Concept Language with Complement and Cardinality ($\mathcal{ALCQ}$) to be appropriate for this purpose. Next, the relevant constructors in the selected language, those which when verbalized result in the required healthcare messages, needed to be identified. Verbalization patterns could then be obtained for the selected constructors.

RQ3: How can the Runyankore verbalization components be combined to form the grammar engine?

This question is concerned with how to combine the noun pluralizer, verb conjugater, and verbalization patterns to form a grammar engine that generates text. There were three main considerations here: how to account for phonological conditioning, which is required during pluralization, conjugation, and verbalization; the kind of implementation to associate verbalization with ontology creating, modifying, and reasoning; and generalizing the output of the surface realizer to other domains besides healthcare.

RQ4: How can the Runyankore and isiZulu approaches to ontology verbalization be generalized to other agglutinating Bantu languages?

The question of generalizability is important because, if possible, the existing Runyankore and isiZulu text generation approaches may be extended to other Bantu languages without requiring the same extensive and time-consuming research. There were two areas of investigation here: the appropriate method by which to investigate generalizability; and how to demonstrate generalizability beyond the languages under investigation, given that there are hundreds of Bantu languages. We investigated whether the bootstrap approach, which had been successfully applied to obtain some Runyankore verbalization patterns from the existing isiZulu ones, could also be used for other agglutinating Bantu languages. The approaches taken for noun pluralization and verb conjugation were also assessed for generalizability. We relied on the Guthrie zone classification [72, 123] of Bantu languages to reach conclusions on the generalizability to other Bantu languages that were not directly involved in this investigation.

1.3 Research Approach

We carried out this research in two phases: (1) investigating verbalization for a single Bantu language, and (2) investigating whether that approach was generalizable to other agglutinating Bantu languages. In Phase 1 we investigated ontology verbalization in Runyankore, where the processes of noun pluralization, verb conjugation, and verbalization were addressed separately and later combined into a grammar engine. It was important that the approach taken for each process would be applicable to other domains as well as healthcare. For noun pluralization, we developed a pluralizer that combines morphology with syntax and semantics, which can handle simple, mass, and compound nouns, as well as singular-only nouns and prefix exceptions. For verb conjugation, we found Context-Free Grammars (CFGs) [91] to be sufficient to conjugate verbs in Runyankore. For verbalization, we found the Description Logic (DL) $\mathcal{ALCQ}$ to be sufficient for our scope (that is, generating drug prescription explanations), selected eight constructors, and developed verbalization patterns for each constructor. We then combined the pluralizer, CFGs, and algorithms resulting from the verbalization patterns to form the grammar engine. Finally, we evaluated the quality of the generated text in two ways: (1) a rate test to assess the grammatical correctness and understandability of the generated sentences, and (2) a human-based comparison test to assess whether the generated text was distinguishable from human-authored text, using both linguists and non-linguists in our evaluation.

In Phase 2, we investigated how the approaches used for Runyankore and isiZulu could be generalized to other agglutinating Bantu languages. We used the bootstrap approach because the Runyankore verbalization patterns of some constructors were obtained by bootstrapping from the existing isiZulu ones. Due to Runyankore and isiZulu belonging to different language zones (zone J.10 and zone S.40 [123] respectively), we investigated bootstrapping among languages in different zones (inter-zone bootstrappability), and bootstrapping among languages in the same zone (intra-zone bootstrappability). We verbalized five constructors in five different languages in this investigation. When investigating the use of the bootstrap approach, we used both Runyankore and isiZulu as source languages, and due to the presence of multiple source languages, also developed a measure of bootstrappability to determine the most efficient source-target language bootstrap pair. Further, by studying the noun class systems of these languages, we identified the need to regroup the class prefixes in Meinhof's noun classification, regarded as a standard among linguists, in order to enable generalizability. We were thus able to develop a noun pluralizer with language-independent rules based on the same combination of morphology with syntax and semantics as we used for Runyankore and isiZulu. Further still, by studying their verbal morphologies, we found that CFGs can be applied to verb conjugation in the five additional languages under investigation. Finally, having understood the generic and language-specific processes and resources required to generalize the text generation approach, we propose an architecture for an API to verbalize ontologies for agglutinating Bantu languages.

1.4 Research Contributions

This research has resulted in the following contributions:

(1) A domain-independent language-specific noun pluralizer for Runyankore that can pluralize simple, mass, singular-only, and compound nouns, as well as prefix exceptions and noun phrases;

(2) Evidence for the use of CFGs to conjugate verbs across domains in Runyankore—for both the standard verbs and for the copulative and auxiliary;

(3) Algorithms for verbalizing subsumption ($\sqsubseteq$), conjunction ($\sqcap$), negation ($\neg$), existential quantification ($\exists$), universal quantification ($\forall$), maximum cardinality ($\leq$), minimum cardinality ($\geq$), and exact cardinality ($=$) in Runyankore across domains;

(4) An architecture for domain-independent ontology verbalization in Runyankore that shows how to account for the noun class system, noun pluralization, verb conjugation, verbalization, and phonological conditioning;

(5) A tool for verbalizing ontologies in various domains in Runyankore in the form of a Protégé plugin;

(6) A regrouping of class prefixes in Meinhof's noun classification, increasing the number of noun classes from 29 to 54, in order to ensure a deterministic output during computational processes and thus enable generalizability;

(7) A language-independent noun pluralizer for agglutinating Bantu languages based on an approach that combines morphology with syntax and semantics;

(8) A metric, based on edit distance and linear gap penalty, for assessing the efficiency of bootstrappability when there is more than a single source language from which to bootstrap;

(9) Evidence for the use of CFGs to conjugate verbs in the simple present tense with negation for chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda; and

(10) An architecture for a domain-independent API to verbalize ontologies in agglutinating Bantu languages, showing how to account for generic and language-specific processes and resources.

The above contributions provide methodological benefits in the field of computational linguistics for complex, under-resourced languages like Runyankore. The application of the approaches and software tools we develop is not restricted to the healthcare domain, but we use examples from healthcare throughout this thesis in order to demonstrate how the output is applicable to this domain. In chapters 3, 4, 5, and 6, we provide details on the processes required for ontology verbalization for Runyankore. This information is helpful for understanding how to deal with the complex grammatical structure of Bantu languages.

The results on using the bootstrap approach to obtain verbalization patterns for Runyankore from isiZulu's is novel in this context, and we provide more evidence for its use by bootstrapping from Runyankore and isiZulu to chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda. Our investigation extends verbalization patterns to these five languages, providing a selection of source languages from which to bootstrap a new target language. By quantifying the bootstrap process through our bootstrappability metric, we contribute a means of assessing the most efficient source-target language bootstrap pair, thus providing a means of developing such language resources faster.

The regrouping of the class prefixes in Meinhof's noun class system [136] is a major contribution that enables a deterministic outcome during noun pluralization. We practically show how our regrouping applies to seven languages and, based on literature on the noun classes of other Bantu languages [122], also present results on its applicability beyond these seven languages. This is an important resource for computational linguistics of Bantu languages, as it enables researchers to overcome the non-determinism when Meinhof's noun classification is applied to computational tasks, while maintaining the concord system for each class in the new classification.

Though the Guthrie zone classification method has been criticized as not always being accurate or consistent as more linguistic knowledge is revealed about individual languages, it is nonetheless regarded as less problematic to establish than genetic classification [123]. Our results on inter-zone versus intra-zone bootstrappability show that for most constructors, intra-zone bootstrappability is usually more efficient than inter-zone bootstrappability, and that inter-zone bootstrappability is almost always possible. This implies that when bootstrapping to a new language, source languages in the same zone as the target language should be considered first. However, given that there are over 80 Guthrie zones [123], and Bantu languages are generally computationally under-resourced, the probability of having a target language from a zone with no source languages is quite high. In this case, results from our research show that it is possible to obtain efficient results on bootstrappability through inter-zone bootstrappability.

The development of a language-independent noun pluralizer (that achieved high levels of accuracy), evidence for the use of CFGs for verb conjugation, and architecture for an API are all resources that we contribute to enable the development of computational resources for other agglutinating Bantu languages. Research into computational resources for this class of languages is not common, and solutions are complex to implement. Our work thus contributes to the growing body of work in this area along side existing resources such as: a morphological analyzer for isiZulu [19] that was bootstrapped to isiXhosa, seSwati, and isiNdebele [21]; morphological analyzer for Kiswahili [80]; morphological generator for isiZulu [19]; ontology verbalizer for isiZulu [29, 96, 98, 99, 100]; word-form recognizer and morphological tagger for isiZulu [20]; part-of-speech tagging for Kiswahili, Ciluba, Northern Sotho, and isiZulu [43]; parallel corpora for machine learning, such as the SAWA English-Kiswahili corpus [44]; as well as all the research presented on [1].

1.5 Thesis Structure

This thesis contains nine chapters, which can be categorized into four parts:

1. **Introduction and Background**:

   Chapter 1: We provide a background to the research area, and introduce the problem statement, research questions, and research contributions.

   Chapter 2: We present the literature relevant to this research, specifically, Natural Language Generation (NLG), Bantu languages, ontologies, and NLG in personalized health information. We show how the reviewed literature guided the decisions we made in this research.

2. **Ontology Verbalization in Runyankore**:

   Chapter 3: We present how one of the prerequisites to verbalization, noun pluralization, was achieved in Runyankore. We explain the use of a combined morphology with syntax and semantics approach to pluralize nouns, starting with the standard noun classification rules, and then extending to exceptional cases such as compound, mass, and singular-only nouns.

   Chapter 4: We present how the other prerequisite to verbalization, verb conjugation, was achieved in Runyankore. We first provide details on the Runyankore verbal morphology, and then explain the analysis used to select an appropriate tense and aspect, as well as the use and evaluation of Context-Free Grammars for verb conjugation.

   Chapter 5: We investigate verbalization for subsumption ($\sqsubseteq$), conjunction ($\sqcap$), negation ($\neg$), existential quantification ($\exists$), universal quantification ($\forall$), maximum cardinality ($\leq$), minimum cardinality ($\geq$), and exact cardinality ($=$) in Runyankore, as well as phonological conditioning for the nasal compound *-nk-*.

   Chapter 6: We explain the details of the implementation of the grammar engine as a Protégé plugin, including its annotation model and architecture. The evaluation of the generated text is also presented here.

3. **Generalizability and Bootstrappability**:

   Chapter 7: We present the methodology and results on the investigation into generalizability and bootstrappability, resulting in: a regrouping of the class prefixes in Meinhof's noun classification; verbalization patterns for chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda; and a metric for measuring bootstrappability.

   Chapter 8: We provide details on how an API to generate text in agglutinating Bantu languages can be developed. We first describe the development of a generic pluralizer, then show that Context-Free Grammars are adequate to conjugate verbs in other agglutinating Bantu languages, and finally propose an architecture for an API to generate text in agglutinating Bantu languages.

4. **Conclusion**:

   Chapter 9: We conclude this thesis by restating the research questions and showing how we answered them. We also show the remaining gaps in our work, which have been left for future work.

Chapter 2

Literature Review

2.1 Overview

This chapter presents the relevant literature about the key areas of the research on text generation in agglutinating Bantu languages. In Section 2.2, we present literature on NLG, its definition, application, tasks, approaches, and architectures. Section 2.3 provides information on the key linguistic features of Bantu languages: noun class system, agglutinative structure, verbal morphology, and classification into Guthrie zones [123]. We also present some details on a particular Bantu language, Runyankore. Section 2.4 presents details on ontology verbalization, including the use of controlled natural languages during text generation. Lastly, we describe the domain of personalized patient information and the role of NLG in it in Section 2.5. In order to avoid having a fragmentary chapter by trying to present all the literature on every aspect of this research, we have decided to limit the content in this chapter to the overarching areas of our research, and present details on specific methodologies, approaches, or formalisms in the relevant chapters where they are discussed.

2.2 Natural Language Generation

Natural Language Generation (NLG), a subfield of computational linguistics and artificial intelligence, is concerned with the construction of computer systems which can produce understandable texts in a selected human language from an underlying non-linguistic representation of information [22, 160, 161]. Precisely defining NLG is rather difficult because its inputs vary substantially, and can include: flat semantic representations, graph structures, numeric data, structured knowledge bases, images, and even video [63]. On the other hand, there is agreement that the output from NLG is text, and therefore, NLG maps from data or meaning to text [63]. This may involve filtering and abstracting the input into a set of preverbal messages [63], which are then output as a collection of text fragments, whose length can range from a single word to several pages [65].

The aim of NLG is to satisfy some communicative goal [22, 65, 91, 161], which could include:

(1) Generating summaries from data on events like football games [113], or from patient data [62, 74];

(2) Generating reports, such as weather reports [153, 156], reports on current affairs [114], feedback reports [183], and financial reports [149];

(3) Generating personalized information, such as personalized patient information [46, 53, 120], or personalized environmental information [184]; and

(4) Generating informative text; for example, descriptions of artifacts in a museum [4, 170].

The desired or expected impact of the generated text depends on the communicative goal. The text might be intended to persuade, motivate behaviour modification, or inform [63]. The generated text must therefore be coherent, accurate, valid, informative, understandable, and relevant in order to achieve the intended communicative goal [63].

An NLG system is thus a translator which converts a computer-based non-linguistic representation of language into natural language text [65]. NLG systems are mostly used to present information to non-expert users in natural language (a representation that they find easy to understand)), instead of the computer's straightforwardly manipulatable internal representations of data [160]. Because the inputs to an NLG system are generally unambiguous, well specified, and well formed, the main concern of NLG is choice, regarding: (1) selecting the appropriate content to linguistically express the inputs in accordance with the communicative goal, (2) selecting the most appropriate lexical item to express a particular concept, and (3) selecting the most appropriate way to aggregate the selected content into phrase, clause, and sentence-sized blocks [91]. These choices can be categorized as NLG tasks.

2.2.1 NLG Tasks

The choices made when mapping from some input data, information, or knowledge to an output text are categorized into six NLG tasks [22, 63, 160, 161]:

(1) Content determination: deciding which information to include from the underlying data sources;

(2) Discourse planning or text structuring: determining the order and structure of the set of texts to be created in order to ensure that it is not presented as a random collection of pieces of text, but rather in a coherent manner;

(3) Sentence aggregation: deciding how to group the text together into sentences;

(4) Lexicalization: deciding which specific words and phrases to use to express information;

(5) Referring expression generation: choosing the words and phrases to identify domain entities; and

(6) Linguistic realization: combining the words and phrases to form grammatically correct sentences.

Content Determination

Content determination is primarily about deciding which information to include, and which to exclude [22, 63, 160], because the data is very detailed, and typically not all the information contained in the data is required for generation [63]. The choices made during content determination are also associated with the communicative goal [22, 160] (as discussed above) and the approaches applied during this task are typically related to the application domain [63, 160, 161]; for example, there may be a different approach for healthcare, where the data on physiological parameters is continuously collected, and thus needs to be filtered and abstracted into a set of preverbal messages; and a different approach for sports, where data on certain events (such as a pass or foul in football) may be omitted in favor of other events (such as each goal scored) [63].

Discourse Planning

After determining the content from which to generate text, the next decision involves in what order to present the information to the user [22, 63, 160]. This decision can be based on the temporal sequence of events (for example, in the order in which goals were scored [113]), importance (such as stating the most important medical results first [120]), or relatedness (grouping data describing the same medical outcome together [63, 120]. This results in a discourse plan, which is a structured and ordered representation of the information [22, 63, 160]. The above examples of discourse planning methods also show how the application domain constrains ordering preferences, as was the case with content determination [63].

Sentence Aggregation

The structure and order imposed during discourse planning do not imply that a single sentence is used for each message [63]. Readability and coherency can sometimes be achieved by combining multiple sentences into a single one [63]. This process of grouping related messages together into sentences is referred to as sentence aggregation [22, 63, 160], and can be used for reducing or eliminating redundancy in the messages, or combining similar linguistic structures together [63, 160]. The early work on sentence aggregation, like content determination and discourse planning, was strongly domain and application specific, and comprised handwritten rules [63]. However, recent approaches are data-driven, where the rules are acquired from either corpus data by identifying similarities between entries in a parallel corpus of sentences and corresponding database entries; or global optimization based on the pairwise similarity of pairs of database entries [63].

The tasks of content determination, discourse planning, and sentence aggregation involve choices centered around the data, concerned with which information to convey to the audience [63], and are, as such, language-independent. But this is not to say that they have no bearing on linguistic aspects. In fact, MacDonald [119] argues that these early tasks contain linguistic knowledge as they represent the articulation, reasoning, and comprehension components, with some linguistic processing at every stage, and thus regards them as substantial tasks for generation. On the other hand, the following three tasks are concerned with which words to use to convey the desired message, and how to output them using the correct grammar; thus being heavily linguistically grounded [63].

Lexicalization

This task begins the conversion from abstract representation to natural language, and is concerned with deciding the words and phrases to use during generation [22, 63, 160]. Natural language is ambiguous, with the possibility of expressing a single event in various ways, and the complexity of the lexicalization task is brought about by the number of alternatives that the NLG system can handle [63]. The decisions made during lexicalization depend on: the need for variation in the generated text, stylistic constraints, the attitude towards the event, presenting numerical information (dates, time, temperature, etc.), and choosing adjectives to make comparisons (larger, taller, younger, more, etc.) [63]. Key considerations when making comparisons are: the reasoning that the system needs to do on the dimensions of an entity (for example, the time units to determine 'before' versus 'after'), and the vagueness that can inadvertently appear in the generated text [63] (for example, stating that an entity is younger than another does not state by how much).

Referring Expression Generation

Reiter and Dale [160] define 'referring expression generation' as 'the task of selecting words or phrases to identify domain entities'. It differs from lexicalization by being a discrimination task, concerned with distinguishing one domain entity from another [160]. It is domain-independent, with several existing stand-alone solutions [63]. The decisions made regarding how the NLG system refers to different domain entities depend on whether a particular entity has already been mentioned (in which case, its name can be replaced with a pronoun or definite description); and the need to distinguish between similar entities (based on properties such as time, size, colour, pluralization, position, etc.) [63]. However, a balance between too much and too little information needs to be made [63]. For example, in the Gre3D corpus for objects in a visual domain [181], providing too much in the description, such as 'the small blue ball before the large green cup', can lead to misleading or boring text, while too little information, such as 'the ball', can hinder identification of the entity being referred to [63].

Linguistic Realization

Linguistic realization involves putting together the selected words and phrases to form a grammatically correct sentence, including the correct morphology, verb agreement, prepositions, and punctuation [22, 63, 160]. The main difficulty of this task is that the output requires various linguistic components that are not present in the inputs [63], such as tense, aspect, and plurals.

The main techniques used for linguistic realization are: canned text, templates, statistical approaches, and grammar engines [63, 79].

Canned Text Canned text consists of preprocessed sentences or paragraphs which are selected and output [79]. Canned text is typically easy to implement, and involves having a list of sentences or phrases, which can be output either without modification, or concatenated with some glue text [91, 167]. Examples of the use of canned text include systems which generate customized form letters by inserting a person's name in the appropriate places [91].

Canned text is unable to adapt to new situations without human intervention [91, 167]. An attempt to use canned text to generate personalized health promotion letters resulted in unwieldy, incoherent, and poorly generalized text [31].

Templates Templates are selected predefined structures represented as slots of text which can be filled by inserting values into the slots [160]. These slots or blank spaces usually have associated requirements specifying what kind of content-specific information can fill them [79, 160]. The values inserted into the slots sometimes undergo further linguistic processing and are annotated with linguistic and formatting information [63, 160].

Template filling is more flexible than canned text [91, 179], because templates can express their non-linguistic input in varying degrees of directness [179]. They can thus be used in complex domains such as sports [113] and healthcare [120], and in the latter case have been successfully applied to generate customized patient information [48, 49, 52, 53, 54, 120].

Templates are advantageous because they provide some control for the quality of the output, preventing the generation of ungrammatical structures [63]. This can be achieved through the use of syntactically structured templates that enable the recursive filling of gaps, minimal templates, or grammars to aid linguistic realization [179].

They however have two main disadvantages: (1) they are labour-intensive when handwritten, and (2) they do not scale well to applications that require considerable linguistic variation [63]. The first disadvantage has recently been solved by learning templates from corpus data, for example in Kondadadi, Howald, and Schilder [103], but the second still persists, and extends to grammatically complex languages, such as Bantu languages. In fact, Keet and Khumalo [100] showed templates to be inapplicable to Bantu languages unless used with a fully-fledged grammar engine.

Statistical Approaches The statistical approaches are recent, and they aim to acquire probabilistic grammars from large corpora [63]. This both reduces the amount of manual effort required by handwritten grammars, and increases the coverage of the resulting grammars [63]. There are three main approaches: the first, a forest representation of alternative realizations, obtained from a small handwritten grammar, from which the optimal candidate is selected by a stochastic reranker [63, 110, 111]. The second approach involves directly generating the optimal realization from the statistical information in the corpora (for example, the PCRU system that uses Context-Free Grammars to generate the most likely derivation of a sentence given a corpus [13, 63]. Both of these approaches rely on a handwritten generator on which the realization task is based, and statistical information is used to select the optimal output [63].

The third and most recent approach is to use fully data-driven grammars obtained from treebanks, thus also relying on statistical information for the base generator [63]. In this approach, a

classifier or particular grammar formalism is used with a treebank corpus, and statistical language models are used for reranking for the optimal output [63]. For example, the OpenCCG framework [58, 185] uses a corpus of the Combinatory Categorial Grammar (CCG) formalism [169] derived from the Penn Treebank [78] in its broad coverage English realizer [63]. However, the statistical approach is resource-intensive, and therefore cannot be used by under-resourced languages.

Grammar Engine A grammar is a definition of a language as a set of strings and the analyses or rules associated with these strings [155]. A grammar engine thus considers grammatical categories (such as sentence, noun phrase, and verb phrase) and rules which implement the categories as objects with complex sets of properties (such as tense and grammatical form) associated with them [91]. Realization systems that rely on a grammar engine make some or all of the decisions based on the grammar of the selected language [63]. The Systemic Unification Realization Grammar of English (SURGE), for example, is a reusable, domain-independent realization component that comprises a comprehensive computational grammar of English; it has been applied to generate text in several NLG systems with differing architectures [56]. The grammar can be manually written (such as the Functional Unification Grammar [91]), in which case they require very detailed input, or it can be modeled by traversing a network and making choices based on grammatical and semantic information [63]. The Systemic Unification Realization Grammar of English (SURGE), for example, is a comprehensive computational grammar of English that has been applied to generate text in several NLG systems with different architectures [56].

Advantages of handwritten grammars guided by linguistic knowledge are: enabling precise, logical analysis of content; predictability and easy programmability; support for detailed error analysis and explanations; and not requiring a lot of data [155].

On the other hand, handwritten grammars suffer from limited coverage and an high initial cost when they are being developed [155]. The former leads to the difficulty of how to make choices among related options in grammar-based systems, where handwritten rules with the right sensitivity to context and input are difficult to design [63].

One of the most well-known grammar formalisms is the Grammatical Framework (GF), which is based on type theory and functional programming [154, 155]. It has successfully been used as a platform for multilingual applications such as translation, localization, and information retrieval, and currently supports over 30 languages [154, 155]. However, none of these languages are Bantu languages, and an attempt to add Kiswahili (regarded as a Bantu language) managed to only introduce some parameter types [143].

GF requires a large resource grammar, referred to as the Resource Grammar Library (RGL), that has a core abstract syntax made up of 86 categories, 216 functions, and a test lexicon of 524 word senses [155]. Additionally, the abstract syntax of the RGL was originally designed for European languages, though support for other language families (Thai, Japanese, and Chinese) has been achieved [155]. This is done by 'tweaking' the grammar, but if the grammar of the new language needs more parameters than currently supported by the abstract syntax, then the code becomes more complex [155]. This partly explains why there is still no GF implementation for the grammatically complex Bantu languages, and why the attempt in [143] to add Kiswahili did not succeed in creating a resource grammar. Ranta, Tian, and Qiao [155] present the use of a chunking grammar as an alternative to modifying the RGL, while increasing coverage. But they also admit that this results in lower quality output, due to an absence of grammatical dependencies between chunks, resulting in agreement errors and incorrect word order.

Grammar-based Linguistic Realizers Grammar engines have been used to develop realization engines. One such engine is SimpleNLG, which offers direct control over the processes of building and combining phrases [64]. It comprises: (1) a lexical component that defines a lexicon, morphological rules, and lexical items that correspond to major grammatical categories (noun, verb, adjective, etc.); and (2) a syntactic component that handles different phrasal types defined

by several grammatical features (tense, number, person, and mood) [64]. It was initially developed for English [64] and has since been ported to French [180], German [18], Italian [1], Mandarin [34], Brazilian Portuguese [42], and Spanish [152]. However, to the best of our knowledge, there is currently no implementation of SimpleNLG for any Bantu language.

Another multilingual NLG system is KPML, which is a graphical-based platform for large-scale grammar engineering for multi-lingual text generation [12]. It supports the construction, maintenance, and use of large-scale, broad-coverage grammars based on the generation architecture developed from the sentence generation component of the Penman text generation system [12, 127]. KPML has been used to develop grammars for Bulgarian, Chinese, Czech, Dutch, English, German, Russian, and Spanish[2]. To the best of our knowledge, there is currently no KPML grammar for any Bantu language.

For Bantu languages, the pattern-grammar-based approach taken by Keet and Khumalo [98, 99] is one that has been used for linguistic realization through the verbalization of logical theories in isiZulu, a Bantu language indigenous to South Africa. Mahlaza [121] also developed grammars that were aimed at generating weather bulletin verbs in isiZulu and isiXhosa, another South African Bantu language. The grammar rules developed by Keet and Khumalo [95] and Keet and Khumalo [98, 99] account for: (1) the classes of nouns in isiZulu and the concordial agreement these classes govern; (2) the isiZulu complex verb, covering negation, tense, aspect, mood, and extensions; and (3) the agglutinative structure. These patterns were used to develop a surface realizer for isiZulu [97]. To the best of our knowledge, these are the existing cases of surface realization in Bantu languages.

Having presented the different NLG tasks and what they entail, we next discuss how these six tasks can be organized to form an NLG system.

2.2.2 NLG Architectures and Approaches

Gatt and Krahmer [63] broadly distinguish among three dominant NLG architectures: (1) modular approaches which, by design, involve fairly clear divisions and significant variations among the NLG tasks; (2) planning-based approaches which avail a more integrated perspective on the NLG tasks by viewing text generation as planning; and (3) global approaches which are heavily reliant on statistical learning and correspondences between inputs and outputs, and therefore cut across the divisions inherent in the NLG tasks.

Modular Approaches

Reiter and Dale [160] and [158] introduced a pipeline architecture where different modules incorporate the different NLG tasks described in Section 2.2.1. This approach is typically a three-tier architecture comprising the text or document planner, the sentence planner or microplanner, and the linguistic realizer [63, 160].

The first tier, the text or document planner, combines the first two tasks of content determination and discourse planning [63, 160] (see Section 2.2.1) in order to model the choice of 'what to say' [63]. The output of this tier is a structured representation of messages referred to as the text plan [63]. This then serves as the input to the next tier, referred to as the sentence plan or microplanner, which combines sentence aggregation, lexicalization, and referring expression generation [63, 160] in order to model the choice of 'how to say it' [63]. The third tier is composed of the task of linguistic realization, where the final text is generated in a grammatically correct way [63, 160].

[1]The Italian implementation of SimpleNLG is available from https://github.com/alexmazzei/SimpleLEX-IT

[2]See http://www.fb10.uni-bremen.de/anglistik/langpro/kpml/kpml-description.htm for details of existing grammars in KPML, and how to access them.

This modular approach clearly separates the tasks regarded as strategic generation (content determination and discourse planning), concerned with the 'what', from those regarded as tactical generation (sentence aggregation, lexicalization, referring expression generation, and linguistic realization), which are concerned with the 'how' [63]. However, in practice, this division is not universally accepted [63], with the organization of tasks into different tiers varying from system to system, and in some cases with the same tasks split across tiers instead of being wholly contained in a single tier [131]. Mellish et al. [131] thus proposed an alternative formalism that accommodates different types of information flow between the NLG tasks, as opposed to ordering a specific architecture. Their formalism, the 'object-and-arrows' framework, was intended to specify high level descriptions of different architectures, whilst retaining the principle that the tasks are well-defined and distinguished from each other, regardless of how they are organized [63, 131].

Reiter [157] also recognized the need for the pipeline architecture to accommodate raw, unstructured inputs that require some preprocessing before undergoing text planning. Examples of application domains with such inputs include generating weather reports and generating patient summaries [63]. In these cases, some data abstraction followed by data interpretation is required before the processes of text generation can begin. Reiter [157] therefore extends the pipeline backwards so as to account for processes carried out before text planning.

The two main limitations of a pipeline architecture are: (1) mismatches between strategic and tactical components, where decisions made earlier in the pipeline lead to unforeseen consequences further downstream; and (2) generating under constraints, where the output is restricted to certain requirements, such as the length of the sentence [63]. In the first case, referred to by Meteer [133] as the generation gap, the ordering of sentences determined by an NLG system during sentence planning may, for example, result in ambiguous output during linguistic realization [63]. For the second case, it might be possible during linguistic realization to formalize restrictions in terms of words or characters, but it is much harder earlier in the pipeline where it is unpredictable to match prelinguistic representations to the final text [63].

These problems, among others, have led to the development of alternative NLG architectures, those that blur the boundaries between modules and the NLG system [63]. These are discussed further below.

Planning-based Approaches

Planning-based approaches regard text generation as the execution of planned behavior, where the planning problem is the process of identifying a sequence of one or more actions that satisfy the communicative goal [63]. The overarching communicative goal can be subdivided into sub-goals that are themselves satisfied by actions, with each action having its preconditions and effects [63]. In principle, therefore, there are no restrictions on what types of actions can be included in a plan, thus making planning-based approaches able to cut across the NLG tasks that are split in to tiers in Section 2.2.2 [63]. In this way, planning-based approaches combine both strategic and tactical components by viewing 'what to say' and 'how to say it' as part of the same set of operations [63]. Planning can be achieved based on the grammar, or stochastically using reinforcement learning [63].

Planning through the grammar is done by interpreting linguistic formalisms as planning operators, and requires grammar formalisms that integrate multiple levels of linguistic analysis, from pragmatics to morpho-syntax [63], such as the lexicalized tree adjoining grammar [90] and the combinatory categorial grammar [169]. Once the NLG goal is formulated using an appropriate plan description language, it becomes possible, in principle, to then use any planner to generate text [63]. However, the biggest drawback of planning-based approaches is that they incur considerable computational expense because they require highly expressive formalisms to capture deep reasoning about beliefs, desires, and intensions [63]; Koller and Petrick [102] found that planning-based systems spend a significant amount of time on preprocessing.

Stochastic planning, instead of regarding a planned action and its consequences as having a fixed relationship, rather views the planning of a good solution to achieve the communicative goal as a stochastic optimization problem [63]. The use of reinforcement learning in this context is a recent development, where generation can be modeled as a Markov decision process with states paired with possible actions, and each state-action pair having a probability of being moved by action a from a state at a particular time t to a new state at time $t+1$ [63]. The transitions between states are associated with a reinforcement signal in the learning algorithm, and this is done by a reward function that uses simulations to quantify the optimality of the generated output through associating rewards with different generation plans (possible paths through the states) [63]. This approach was used to optimize the choice of selecting information during referring expression generation [84], and it was shown to be effective for optimizing information presentation for generating restaurant recommendations [162].

Global Approaches

This is a broad category used to encompass several other integrated approaches to NLG which rely on statistical methods, and thus take a global rather than a modularized view of the NLG process, and also require large amounts of training materials that are pairs of inputs (data) and outputs (text) [63]. There are several ways to model the NLG process, given the alignment between data and text; these include modeling NLG as a sequential stochastic process, as a classification task, as a parsing problem, or as a representation from deep neural networks [63].

When modeled as a sequential, stochastic process, the division between the strategic and tactical choices (see Section sec:pipe:arch) is maintained, but statistical alignment between data and text is used to determine content selection, while different NLG techniques (such as templates and grammars) can be used for sentence planning and realization [63]. Stochastic planning can also be used to form a single probabilistic model by combining strategic choices (database records and fields) with tactical choices (word sequences), thus pairing data to text based on a sequential, Markov process [116]. Further, an end-to-end architecture that maintains a division between the three tiers in Section 2.2.2 (content selection, sentence planning, and linguistic realization) models each process as a sequence of decisions in a log-linear framework [5]. This is done by using different sets of features to inform the decisions at each stage of text generation, such as basing sentence planning and surface realization decisions on templates acquired from a corpus, and a template is selected based on its likelihood given the database fields selected during content selection [5]. They can thus handle long-range dependencies more flexibly by conditioning choices on arbitrarily long histories of previous decisions [5]. Finally, a global solution to generation can be achieved without distinguishing between strategic and tactic components, but rather using a tree representation where the root represents a type of dialogue act, the leaves represent words, and the non-terminals represent associations between pieces of input and words [63, 124]. Therefore, by searching for the optimal sequence of word-input associations for a particular dialog act, content selection and realization can be solved jointly [63, 124].

When NLG decisions are regarded as a classification problem, the generation process is modeled using a cascade of classifiers that construct the output incrementally, with the output of a previous classifier serving as input into the next classifier [63]. For example, Marciniak and Strube [126] divided the process of generation into classification subproblems (such as determining linear precedence of discourse units and determining lexical forms for verbs), and then applied corpus-trained classifiers in a machine learning algorithm to generate instructional text. Zarrieß and Kuhn [189] also used a sequence of classifiers to perform referring expression generation and linguistic realization, starting with a corpus annotated with a dependency representation. Their ranking model (based on Support Vector Machines) performs realization by mapping the input to a shallow syntactic tree, and inserts referring expressions, for any given input dependency representation that is extracted from the corpus [189]. The main limitation associated with using a cascade of classifiers for NLG is error propagation (similar to the generation gap discussed in

Section 2.2.2), where errors from earlier classifiers affect classification further downstream [63]. Zarrieß and Kuhn [189] solved this problem by using a revision-based architecture, where syntactic mapping is followed by the insertion of referring expressions, and this in turn is followed by the syntax being revised.

NLG can also be regarded in terms of parsing, where probabilistic context-free grammar formalisms are used to select which rules to expand during generation, based on the probabilities derived from a corpus [63]. The base generator is hand-written, though extracting rules or templates from corpora is also possible [63]. This type of generation can also be used to produce text from structured knowledge bases (expressed in formalisms such as the Resource Description Framework (RDF)) by extracting pairings of textual descriptions with lexicalized grammars or templates [57]. Chen and Mooney [33] took a statistical approach to generate text by identifying the maximally probable sentence given a meaning representation, thus working from meaning to text. But unlike their approach that separates content determination and linguistic realization [33], Konstas and Lapata [104, 105] use a global approach that starts by aligning text with database records, and models the generation process as grammar rules that implicitly incorporate different types of decisions.

The most recent approach used for NLG is deep neural networks, commonly referred to as deep learning methods [63]. Gatt and Krahmer [63] explain that neural networks are designed to learn dense, low-dimensional, distributive representations in increasing levels of abstraction, and that these representations are well-suited to capturing grammatical and semantic generalizations. There are two general categorizations in which different models can be placed: (1) encoder-decoder architectures, and (2) conditioned language models [63]. Encoder-decoder architectures use a neural network to encode the input into a vector representation, and this in turn serves as the input into a decoder neural network [171]. They were used by Dušek and Jurcicek [55] to generate text without the need to align inputs and outputs, with the encoder producing deep syntax trees that the decoder then uses as inputs to generate text. The second category, conditioned language models, generate text by sampling words or characters from a distribution conditioned on input features [63]. The input features may consist of semantic, contextual, or stylistic attributes [63]. For example, semantic and sentiment attributes are the input features that are conditioned on in order to generate product reviews in [118], where as it is the input context (both discrete and continuous information) that is conditioned on in order to also generate product reviews [173].

Having presented the main aspects of the field of natural language generation, we next explain the grammatical structure of Bantu languages.

2.3 Bantu Languages

Bantu languages are a group of languages indigenous to Africa [146]. They extend from the south, below Nigeria, to most of central, east, and southern Africa, as shown in Figure 2.1 [146]. There are Bantu-speaking communities in 27 of the continent's 54 countries, with about 240 million speakers [146]. The exact number of languages classified as Bantu ranges from 300 to 680, based on different criteria by different authors [146].

The default phrasal structure across Bantu languages is Subject-Verb-Object (SVO), and the noun precedes its modifiers within a noun phrase [146].

2.3.1 Noun Classification

Noun class systems are a strong areal feature in Africa, with an estimated two-thirds of the languages on the continent having noun classes [94]. Bantu languages assign all nouns to a class; and there are over 20 noun classes, though some NCs have fallen into disuse in most languages [122, 136, 146]. The largest number of noun classes in a single language seems to be 21 [94], as is the

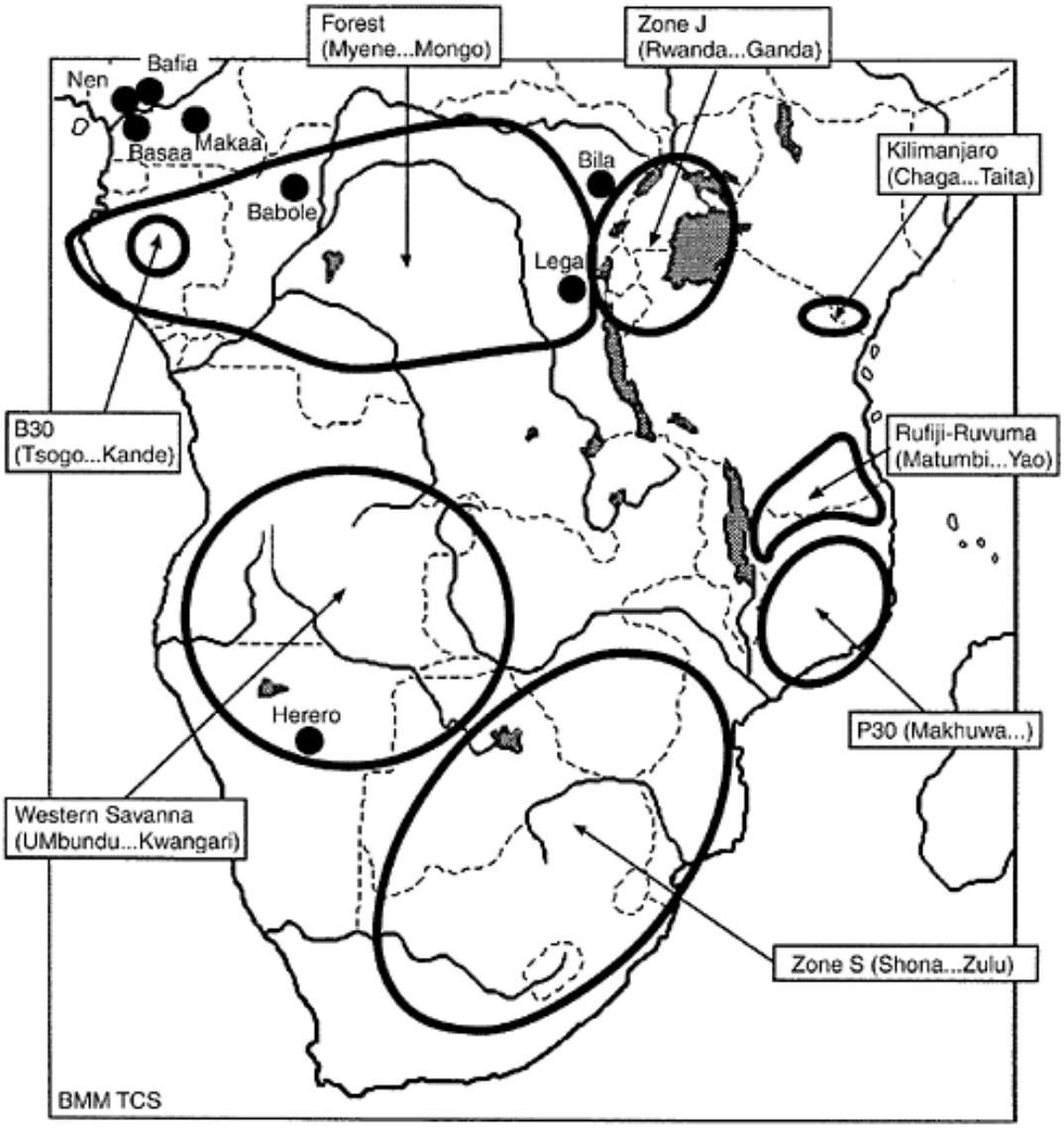

FIGURE 2.1: The spread of Bantu languages across Africa [146]

case in Luganda [11]. The semantic generalizations of the types of nouns in each class are shown in Table 2.1 [11, 88, 94, 99, 101, 122, 136, 174, 190].

The simple noun comprises a prefix and a stem [94]; for example, *omuntu* 'person' in Runyankore, which can be analyzed as the prefix *o-mu-* and stem *-ntu*. Therefore, in addition to the semantic categorization of nouns shown in Table 2.1, nouns are also syntactically categorized according to the prefixes they take [94]. This is the hallmark of Bantu nominal morphology [94].

In many Bantu languages, the class prefix may be preceded by a formative referred to as the augment, pre-prefix, or initial vowel [94, 122]. In the above example of *omuntu*, the class prefix *o-mu-* possesses the augment *o*. The augment is not found in all Bantu languages [94, 122]. For instance, isiXhosa, isiZulu, Kinyarwanda, Luganda, and Runyankore nouns possess the augment [8, 11, 101, 174], while chiShona and Kikuyu do not [88, 190]. However, the augment can be eliminated in certain grammatical constructions, such as when used together with proper names, kinship terms, vocative and predicative constructions, and when followed by a negative verb [94, 122].

TABLE 2.1: Classification of Bantu nouns into noun classes (the 'and' indicates that the two classes are a singular/plural pairing) [11, 88, 94, 99, 101, 122, 136, 174, 190]

Noun Class	**Description of Associated Nouns**
1 and 2	People and kinship
3 and 4	Plants, nature, and some parts of the body
5 and 6	Fruits, liquids, some parts of the body, and paired things
7 and 8	Inanimate objects
9 and 10	Tools and animals
11	Long thin stringy objects, languages, and inanimate objects
12 and 13	Diminutives
14	Abstract concepts
15	Infinitives and parts of the body
16, 17, and 18	Locative classes
19	Diminutives
20, 21, and 22	Augmentatives
23	Locative class

Noun classes do not only classify nouns, but are at the heart of an extensive system of concordial agreement [94], that is, the change in the morphology of a word brought about by grammatical agreement[3]. Each class determines the agreement with: concord patterns; nominal prefix in nouns, locatives, and adjectives; numeral prefix; pronominal prefix for substitutives, connectives, possessives, demonstratives, and determinatives; initial prefix in absolutive verb forms; and the verbal infix [94, 122, 146, 175, 190]. Table 2.2 shows the noun class system of Runyankore (a Bantu language indigenous to south-western Uganda [8]); the augment is depicted for each class prefix, as well as the subject, adjective, and possessive concords for each class.

TABLE 2.2: The Runyankore noun class system, showing the subject, adjective, and possessive concords for each class (the dashes between the letters in the prefix show separation between the initial vowel (augment) and prefix)

Number	**Class Prefix**	**Grammatical Number**	**Subject Concord**	**Possessive Concord**	**Adjective Concord**
1	o-mu-	singular	-a-	owa	o-mu-
2	a-ba-	plural	-ba-	aba	a-ba-
3	o-mu-	singular	-gu-	ogwa	o-mu-
4	e-mi-	plural	-gi-	eya	e-mi-
5	ei-/e-ri-	singular	-ri-	erya	e-ri-
6	a-ma-	plural	-ga-	aga	a-ma-
7	e-ki-	singular	-ki-	ekya	e-ki-
8	e-bi-	plural	-bi-	ebya	e-bi-
9	e-n-/e-m-	singular	-e-	eya	e-n-
10	e-n-/e-m-	plural	-zi-	eza	e-n-
11	o-ru-	singular	-ru-	orwa	o-ru-
12	a-ka-	singular	-ka-	aka	a-ka-
13	o-tu-	plural	-tu-	otwa	o-tu-
14	o-bu-	plural	-bu-	obwa	o-bu -
15	o-ku-	singular	-ku-	okwa	o-ku-
16	a-ha-	locative	-ha-	aha	a-ha-
17	o-ku-	locative	-ha-	-	a-ha-
18	o-mu-	locative	-ha-	-	a-ha-
20	o-gu-	singular	-gu-	ogwa	o-gu-
21	a-ga-	plural	-ga-	aga	a-ga-

[3]Definition of 'concordial' and 'agreement' according to the online version of the Merriam Webster dictionary available at https://www.merriam-webster.com/dictionary/concordial.

The concords are also used to distinguish between classes with the same class prefix (such as classes 1, 3, and 18 in Table 2.2) because their concords differ [94, 122].

Throughout this research, we applied Meinhof's noun classification, a standard among linguists for defining noun classes, where the plural form of a noun is considered to belong to a different class from its singular form [136]. Though Taljard and de Schryver [172] found through a corpus analysis that the noun classes and genders should be regarded as dynamic instead of fixed, we nonetheless use Meinhof's noun classification in our research because it makes computationally defining grammar rules more articulate. The most widespread singular/plural class pairings are classes 1/2, 3/4, 5/6, 7/8, 9/10, 11/10, 12/13, and 14/6, while the less common pairings are classes 3/6, 7/10, and 19/13 [94, 122]. Loan words are also usually found in classes 9/10 [122].

2.3.2 Agglutinative Morphology

We are particularly interested in Bantu languages with an agglutinating morphology, where words are formed by adding affixes to their bases, and each affix carries meaning such as tense and aspect [60, 88, 128, 136, 140, 146, 174, 177]. The example below shows the agglutinative nature of Runyankore:

Runyankore: *Tibakumureeba*.

Morphemes: ti-ba-ku-mu-reeb-a

English: They do not see him/her.

The above Runyankore sentence consists of affixes, each with a separate meaning: *ti*, the negation morpheme; *ba*, noun class (NC) 2 subject concord for the third-person plural pronoun 'they'; *ku*, the infinitive; *mu*, the NC 1 third-person pronoun for 'him/her'; *reeb*, the verb-root for 'see'; and *a*, the indicative final vowel.

2.3.3 Verbal Morphology

The morphological and phonological structure of Bantu verbs is very regular in most languages [145, 165], with a typical verbal form consisting of: one or more bound morphemes, a verb-root, and one or more extensions [165]. The general structure of the verb is as below [145]:

<Initial> <Subject> <Negative> <Tense and/or Aspect> <Object> <Root> <Extension> <Final>

The morphemes preceding the verb-root specify the person, noun class, aspect, time, negation, etc. [165]. The 'initial' usually expresses negation [145], while the subject and object depend on the noun class [146]. The extensions specify valency-changing categories, which can be as many as eleven [165], but the most common are: causative, applicative, stative, reciprocal, reversive, and passive [145, 165]. The final usually contains morphemes associated with mood (indicative or subjunctive) or aspect, but also tense in some languages [145].

Tense and Aspect

Nurse [145] defines tenses as representations of the time that contains a specific event, and locates events in universe time. Aspects, on the other hand, are different representations of the time within a specific event [145]. The widespread semantic features of both tense and aspect in most languages are: (1) tense reference occurs before that of the aspect; and (2) a single verbal word can express a single tense, several aspects, or both tense and aspect [145]. In most Bantu languages, tense and aspect are encoded by inflection of the verb, tone, and extra verbs preceding the main verb; and this is usually done using a single marker [145].

The number of tenses varies among languages, because different languages divide up the timeline of events differently [145]. From an analysis of 120 languages from all but thirteen of over 80

language zones, Nurse [145] concluded that, with a few exceptions, all the examined languages either had one, two, three, or four past tenses, as well as one, two, or three discrete future tenses. Aspect is however, more stable: the perfective, imperfective, and interior aspects are widespread across languages in the world; the progressive and habitual are common, but less so; and, in addition to possessing the five above, the persistive is characteristic of Bantu languages [145].

2.3.4 Language Zone Classification

Due to the large number of Bantu languages in existence, linguists have tried to group several similar languages together as a means of classifying them. There are two main classification strategies that have been applied: referential and genetic [164]. Referential classification groups together languages with similar linguistic features (such as phonetic, semantic, and syntactic) that are geographically colocated, without presupposing their historical relatedness, while genetic classification aims to create a 'family tree' of languages through the cross-linguistic comparison of historical developments of lexical, grammatical, tonal, morphological, and phonological themes on a very large scale [164].

The most well-known referential classification is that done by Malcolm Guthrie, resulting in language classes referred to as 'Guthrie Zones' [72, 123]. It has been criticized as not always being accurate or consistent as more linguistic knowledge has been revealed about individual languages, but Maho [123] argues that there are fewer problems associated with establishing a referential classification than a genetic one. Additionally, genetic classifications have not yet been standardized and are constantly changing, and, despite some detailed genetic analysis on Bantu migration offering origins for some language groups [67], the current state of some knowledge about the internal subgrouping of Bantu languages, on which genetic classifications depend, is still fragmentary [164]. On the other hand, referential classification is regarded as useful precisely because it is independent of the differing opinions associated with genetic classification [164].

Guthrie zones categorize Bantu languages into 16 geographic zones, which are labeled using the letters A, B, C, D, E, F, G, H, J, K, L, M, N, P, R, and S [123]. These are further subdivided into decades, totaling about 80 zones [123, 146]. Zone J.10, for example, contains individual languages from J.11 to J.19, while J.20, J.30, etc. represent different zones [123, 164]. Figure 2.2 shows how this classification covers the Bantu languages throughout the African continent.

The inherent complexity of Bantu languages thus stems from the combination of the noun class system, agglutinative structure, and verbal morphology; and is further compounded by their under-resourced state.

2.3.5 Runyankore

Runyankore is a Bantu language spoken in the south-western part of Uganda by over 2 million people, making it one of the top five most widely spoken languages in the country [8, 175, 177]. As shown in Section 2.3.2, Runyankore is a highly agglutinative language [8, 175] to the extent that a word can be composed of more than five constituents [32]. Like all Bantu languages, Runyankore has several noun classes–20 according to Meinhof's noun classification [8] as shown in Table 2.2.

Additionally, Runyankore possesses the verbal morphology presented in Section 2.3.3, where the past and future tenses are further subdivided to express the degree of remoteness from the present [177]. The past tense is subdivided into the immediate past, near past, and remote past, while the future tense is subdivided into the near future and remote future [177]. This, when combined with their participial forms, results in fourteen tenses [177]. Further, in addition to the six common categories of extensions presented in Section 2.3.3, Runyankore expresses the repetitive, intensive, and instrumental extensions [177].

This complex grammatical structure of Runyankore and other Bantu languages has to be adequately captured during text generation in order to produce grammatically correct text. Chavula

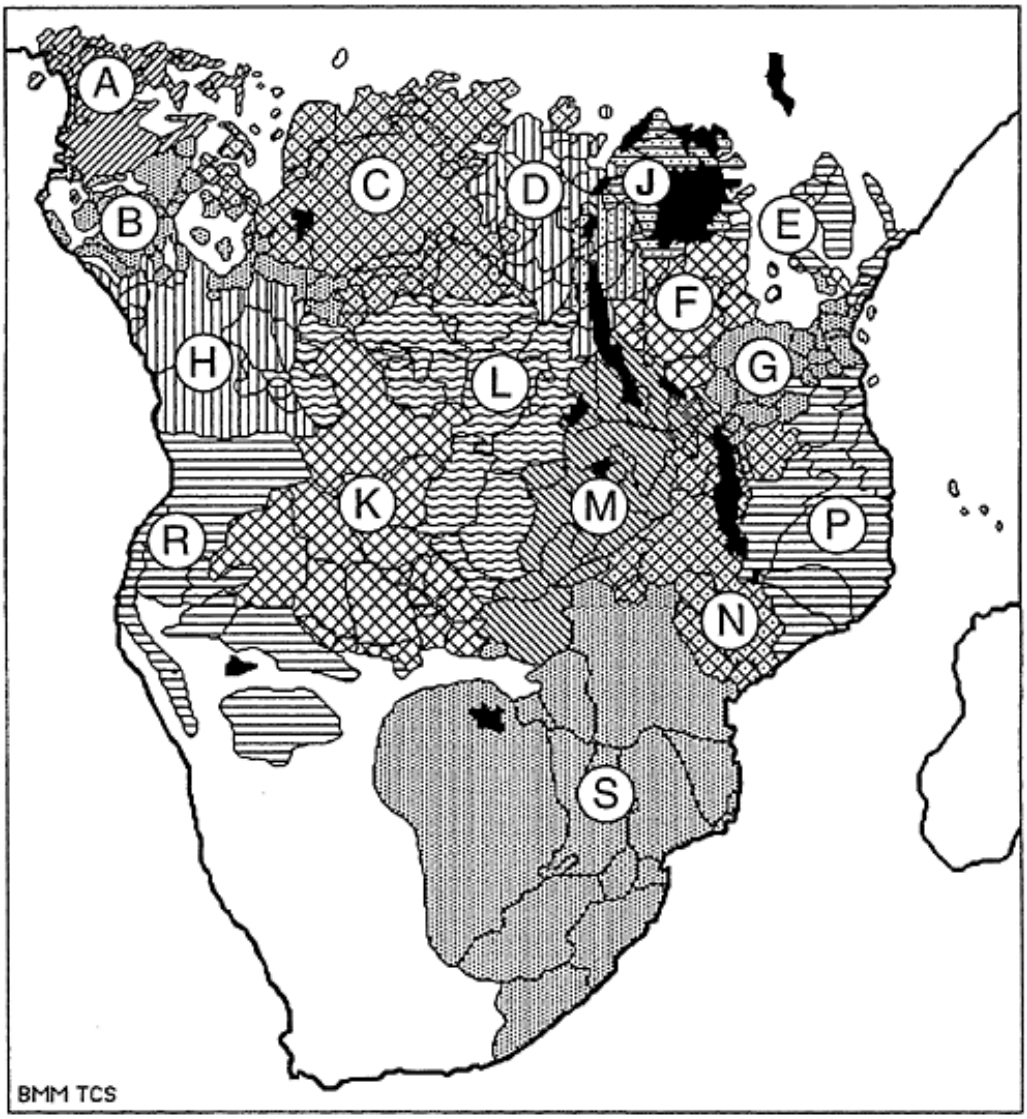

FIGURE 2.2: The spread and classification of Bantu languages by Malcolm Guthrie [123]

and Keet [32] identified that a grammar engine should contain the following morphosyntactic data that is important for the development of ontology-based applications for Bantu languages:

(1) A noun class system, its associated prefixes, and its association with lexical entries need to be defined;

(2) Rules for verbs and adjectives which ensure agreement with the noun class need to be defined; and

(3) Rules for the agglutination process need to be written.

2.4 Ontology

An ontology is an explicit formalism of the semantics of information in a domain, and the relationships and constraints that hold among the information [30]. Ontologies aim to support knowledge-sharing between humans and/or machines in a structured way, by: (1) enabling key

concepts in a domain to be identified and defined in an unambiguous way, (2) facilitating the integration of different perspectives, and (3) capturing key distinctions in a given perspective [30]. Therefore, ontologies describe a domain [163].

An ontology can therefore be referred to as a set of statements which make up the essential knowledge of the domain, that is, the knowledge which must always hold if the theory is to be coherent [59]. Coherence is ensured through the explicit modeling of the domain knowledge as a logical theory using a formal language, and thus is independent of natural language [23, 71].

The degree to which an ontology specifies depends on: the richness of the domain of discourse, the richness of the vocabulary of the formal language chosen, and the axiomatization, which in turn depends on the expressiveness of the language [71]. The simplest ontology describes a hierarchical subsumption relationship between concepts, while more complex ontologies include axioms which express more relationships between concepts and restrict their intended interpretation [30, 70].

Reasons for developing ontologies include [71, 144, 166]:

(1) To share a common understanding of the structure of information in a particular domain among people or software agents;

(2) To facilitate the reuse of domain knowledge;

(3) To make domain assumptions underlying an implementation explicit, so as to allow for modifications to the assumptions as the domain changes;

(4) To separate domain knowledge from the operational knowledge; and

(5) To use a declarative specification of the terms in a domain to analyze domain knowledge.

An ontology has to be represented in terms of a well defined language, in order to provide the mechanisms by which to model the knowledge in a domain [30]. Logics that have been specifically designed to represent and reason upon structured knowledge are called description logics [30].

2.4.1 Description Logics

Description logics (DLs) are knowledge representation languages which can be used to represent the knowledge of an application domain in a formally structured way [9]. A DL has a formal syntax which specifies how to construct well-formed sentences, and formal semantics which relates those sentences to a model [10]. In DLs, concepts represent categories being modeled in the domain, are universal notions which can be instantiated, and denote sets of individuals; and roles denote binary relationships between individuals [9, 10]. There are several DL language features available, each offering a different level of expressiveness [10, 30]. The details on the specific DL applied in our research are presented further in the thesis when presenting our methodology.

In Section 2.2.1, when discussing grammar-based linguistic realizers for Bantu languages, we presented the case of isiZulu where text was generated by expressing logical theories (DL statements) as natural language, and more details are provided in the next section. Safwat and Davis [163], Schwitter [166], and Vinu and Kumar [182] stated that expressing DL statements in a Controlled Natural Language (CNL) is necessary in order to ensure unambiguous interpretation. This is a requirement for ontology verbalization [69].

2.4.2 Ontology Verbalization

The verbalization of an ontology is the generation of natural language descriptions from the model [182]. Expressing the semantics captured in an ontology is done in a pseudo-natural language composed of fixed-syntax sentences, which are a highly restricted syntactic subset of natural language, referred to as Controlled Natural Language (CNL) [68, 163]. According to Kuhn [108], a

language is called a controlled natural language if and only if: (1) it is based on exactly one natural language, its base language; (2) the key difference between it and its base language is that it is more restrictive in lexicon, syntax, and/or semantics; (3) it reserves most of the natural properties of its base language, to the extent that it is still understandable to speakers of the base language; and (4) it is explicitly and consciously defined.

CNLs have to ensure the deterministic interpretation of the statements, and therefore some statements are likely not to conform to a fully correct subset of natural language because obtaining a predictable interpretation is the desired outcome, and this can only be ensured by having a strict syntactic subset of natural language [68, 69, 108]. However, Hallett, Scott, and Power [73] developed a method based on conceptual authoring that was able to overcome these limitations and result in the composition of fluent and complex natural language questions. This is because, in conceptual authoring, all editing operations are defined directly on the underlying ontology thus making interpretation unambiguous, and users interact with the ontology in natural language thus fostering usability [73].

Attempto Controlled English (ACE) is a CNL that has been used in ontology editors, reasoners, and for rule-based machine translation [108]. ACE can express complex noun phrases, plurals, anaphoric references, subordinated clauses, modality, and questions [108]. Examples of ACE statements are shown below [108]:

A customer owns a card that is invalid or that is damaged.

Every continent that is not Antarctica contains at least two countries.

The benefits of obtaining CNLs from verbalizing ontologies include [68, 69, 85, 93, 163]:

(1) Providing a common understanding of the semantics in a domain since natural language is understandable to any speaker of that language;

(2) By bridging the communication gap between domain experts and logicians, CNLs enable the involvement of domain experts in the conceptual modeling and validation phases of an ontology (such as the knowledge editing method by [150] which enables a domain expert to reliably edit an ontology and obtain feedback through its natural language descriptions);

(3) The use of plain text eliminates the need for dedicated ontology editors, since plain text can be viewed and modified using any text editor;

(4) Plain text is easy to store and search; and

(5) Natural language hides the formal syntax of the ontology.

Ontology verbalization has even been the focus of NLG challenges, mainly WebNLG, which focused on generating text from a training dataset composed of pairs of DBPedia data in the Resource Description Framework (RDF) format and their verbalizations [37, 61]. This challenge aimed to promote the development of RDF verbalizers and microplanners that can handle a wide range of linguistic constructions[4] [37].

Ontology Verbalization Implementations

Safwat and Davis [163] categorized several CNL-based tools according to whether they are used for ontology engineering, ontology querying, or other purposes. Here, we only present tools that generate CNLs through ontology verbalization. CNL-based ontology engineering tools are targeted at enabling domain experts to create ontologies in a manner which they are familiar with, that is, natural language, and they include: What You Say Is What You Mean (WYSIWYM) [150], Guided Input Natural language Ontology Editor (GINO) [14], Roundtrip Ontology Authoring

[4]The details on the WebNLG challenge can be found at `http://webnlg.loria.fr/pages/challenge.html`.

(ROA) [40], ACEView [92], and ACEWiki [107]. The CNL-based tools that are used to query an ontology include: Pseudo Natural Language (PNL) [125], Guided Input Natural language Search Engine (GINSENG) [15], and OWLPath [178].

NaturalOWL is a template-based tool that verbalizes ontologies in English and Greek [4]. It can generate fluent and coherent multi-sentence text describing the semantics in an ontology; the text is targeted towards ordinary internet uses, such as customers at a retail website [4]. They used a domain expert in their system to add high quality linguistic information required for verbalization, and the generation of high quality text is thus domain dependent. NaturalOWL uses a three-tier NLG architecture: document planning, where the axioms to be verbalized are selected and the order of sentences is determined; micro-planning, where the NLG tasks of lexicalization, sentence aggregation, and referring expression generation are performed; and linguistic realization, where internal sentence specifications are converted to text [4].

Gruzitis, Nespore, and Saulite [69] found that the methods used to verbalize ontologies in analytical languages which share fundamental characteristics (like english) cannot be directly reused for highly synthetic languages (like Baltic languages). They therefore used Topic-Focus Articulation (TFA) in order to capture the correct word order patterns in the sentence structure [69]. They were able to verbalize subsumption (one concept is a sub-class of another), conjunction (the intersection of sets of concepts), disjointness (the compliment of a set of concepts with respect to the universal set of a domain), existential quantification (the set of concepts, each of which in exclusively satisfies a role), and universal quantification (the set of concepts whose features are only concepts of a given role) in Latvian and Lithuanian [68, 69]. An evaluation of their verbalizations in controlled Baltic languages showed a preference for the verbalization in the singular for all constructors except subsumption, because plural statements (such as 'all professors') are more intuitive than singular ones (such as 'each professor') when making generalizations [69].

The verbalization tool by Keet, Xakaza, and Khumalo [96] generates text in isiZulu. They developed a Python-based verbalizer that verbalizes named class subsumption, disjointness, existential quantification and its negation, universal quantification, and conjunction. The advantages of this tool are that it allows for verbalization in other languages, which can be done by a language-specific verbalization file, and the algorithms can be reused beyond OWL files if necessary [96]. The verbalization is based on patterns from [98, 99], which account for the need for the noun class during verbalization. Ann experimental evaluation of the verbalization patterns by five linguists and seven non-linguists showed that the singular verbalization patterns were preferred for most constructors except universal quantification [99].

Ontology Lexicalization

When verbalizing ontologies, richer models capturing how concepts and relations are linguistically realized are needed to associate linguistic information (such as part-of-speech, morphological decomposition, inflection, sub-categorization frames, etc.) with ontology elements (concepts, roles, and individuals) [23, 35]. This association has been achieved through the use of annotations. Annotation models were used by Androutsopoulos, Lampouras, and Galanis [4] to provide a lexicon entry specifying the inflectional form for each noun, verb, or adjective, while Keet and Chirema [97] annotated the ontology with the noun class, gender, case, tense, and prepositions.

Cimiano et al. [35] argued for more expressive models beyond simple RDF and OWL labels in order to associate linguistic information with an ontology at any level of linguistic description and expressivity. McCrae, Spohr, and Cimiano [130] developed just such a model, Lexicon Model for Ontologies (Lemon), which supports the sharing and linking of terminological and lexicon resources with ontologies. The structure of Lemon is as follows: a lexical object has a number of terms in a lexicon (lexical entries) which comprise several inflectional variants (lexical forms), and each form may have multiple representations (such as, written and phonetic) [130]. Lemon also has senses that represent the correspondence between the lexical entry and the ontology entity; and the lexical entry is linked to the semantic description of the ontology through a reference that

specifies the meaning of the lexical entry [130]. This core model has been extended to form the OntoLex-Lemon model by adding modules on syntax and semantics, decomposition, variation and translation, and metadata [129].

Despite Lemon becoming the primary mechanism for the representation of lexical data on the semantic web [129], it was found to be insufficient for use with Bantu languages, because it does not account for noun class information [32]. On the other hand, an XML-based annotation model was successfully used in the verbalization of two Bantu languages, isiZulu and ChiShona, where the ontology was annotated with NC, tense, and prepositions [97].

2.5 Personalized Patient Information

There has been a growing trend towards patient-centric healthcare, which aims to directly involve patients in the decision-making process by providing them with the information they need to understand their medical condition [31, 49, 54, 186]. This has extended to the development and use of technological interventions whose goal is to aid and motivate patients to adopt behaviors that help to either promote better health or manage existing diseases [147]. The latter includes enabling patients to make more informed decisions about their prescribed treatment [52, 54], which should in turn promote greater compliance and satisfaction with their therapeutic regimens, resulting in better patient outcomes and reduced healthcare costs [31, 52, 54, 77, 186]. This information can serve to complement and reinforce the information communicated during the patient-doctor consultation [31, 48, 49].

Studies in health communication have shown that health education material is likely to be more effective if it is personalized according to the patient's medical condition, demographic, and personality profile [31, 48, 49, 52, 54, 77]. Also, patient educators support the notion that customized patient information can enhance the uptake of information, bring about increased receptivity to behavioral change, and involve patients in their own healthcare decision-making [54]. Additionally, because patients consistently retain a rather small fraction of the verbal information provided during the patient-doctor consultation [48, 49, 50], it is essential that educational materials are given to the patient to complement and augment the face-to-face session [48, 50].

The problem with most health education and patient information material is its limited effectiveness when applied to a wide audience [48, 49, 53]. What is usually produced is either a generic document with minimal information common to everyone [31, 49, 186], or a large document which tries to provide the maximum amount of information considered relevant to someone (and hence mostly irrelevant to many) [48, 49, 50, 53]. Such material, which omits relevant information, or contains irrelevant information, or is not addressed to a particular patient, is likely to be discounted or ignored [48, 50, 53].

The fundamental complexity in the customization of patient information is the number of different combinations of factors or characteristics, which can easily be in the tens or hundreds of thousands [48, 50, 51, 53, 54]. On the other hand, methods from artificial intelligence and computational linguistics have been applied to develop automated systems for personalizing health information to individual patients [46, 48, 50, 52, 54], with the goal of providing more relevant, patient-centric health content [54].

Several NLG systems have been developed for this purpose, and they generate personalized text about: diagnosis and treatment information for cancer patients [31], mental health interventions [81], oral medicine [117], affective messages for parents with babies in neonatal care [120], health education materials [51, 53], and prescription explanations [46]. However, none of these can be reused for Bantu languages because they are template-based and, as explained by Keet and Khumalo [100], templates cannot handle the noun class system, agglutinative structure, and complex verbal morphology of Bantu languages. Additionally, most Bantu languages are too under-resourced for the use of machine translation algorithms on the text generated by these NLG systems.

2.6 Summary

In this chapter, we presented the literature on the main research areas in our research. In the field of Natural Language Generation (NLG) described in Section 2.2, we described the six NLG tasks, the four main NLG techniques, and the three main NLG architectures. The literature shows a move towards corpus-driven statistical-based global approaches because they handle long-range dependencies better and produce representations that are well-suited to capturing grammatical and semantic generalizations. They do however require very large training corpora, and for the purposes of Bantu language linguistics, the application of these approaches is still greatly limited by the under-resourced state of these languages. On the other hand, the literature presented in Section 2.3 shows that the grammar of Bantu languages is well documented generally, and specifically for Runyankore, which draws our work towards the grammar-based NLG technique, where the details on the noun class system, agglutinative structure, and verbal morphology can be formalized as a grammar. And though this literature on the grammatical structures of Bantu languages is informative about their nature, it also highlights the complexity of their grammar, which, coupled with their under-resourced state, explains why the development of computational tools is difficult. Despite this, the importance of CNLs and their limited scope presented in Section 2.4 provides a means of generating text in a manner that reduces the overall grammatical scope by only considering a subset of natural language instead of the full grammar. In deed, we have seen that this method has successfully been used to generate tex in a Bantu language. This therefore offers a natural starting point for how to generate text in Runyankore.

The next chapter describes how one of the prerequisites to verbalization, noun pluralization, was investigated.

Chapter 3

Noun Pluralization in Runyankore

3.1 Overview

Our first research question, 'How can noun pluralization and verb conjugation be achieved in Runyankore' in Section 1.2.1, is concerned with how to address the prerequisites to verbalization of noun pluralization and verb conjugation. In this chapter, we present the findings from our research on the first part of this question, RQ1 (a), regarding noun pluralization. In Section 2.4.2, we explained that an axiom can be verbalized either in the singular or plural. This requires the pluralization of the concepts to be addressed as part of the verbalization process.

At the start of this research, we found that there was no computational approach for pluralizing Runyankore nouns. The approaches used to pluralize nouns of better resourced languages like English and German rely on regular expressions for the endings of the nouns, and are based on extensive linguistic resources that are easily used for the specification of the rules computationally [38, 141]. Pluralizing nouns in Arabic, a more complex language that is computationally well resourced, focuses more on comparing techniques, such as the focus on the lexeme as a central morphological concept, the use of a multi-tier Finite State Morphology (FSM), and modeling form-based morphology [3] and refinements such as 'broken' plurals and irregular gender [2]. None of these methods can be applied to Runyankore because they are resource intensive and/or incompatible with its grammatical structure.

On the other hand, the use of the noun classes in pluralization, by providing singular/plural pairings, has been well documented for Bantu languages generally [94, 122] and Runyankore specifically [8], providing information on the rules governing noun pluralization. However, the pluralization of nouns in isiZulu (a Bantu language indigenous to South Africa) identified some exceptions to the standard singular/plural pairings [29]. The term 'exceptions' in the context of this research refers to nouns whose process of pluralization requires different computational rules from, or additional computational rules to the approach taken for basic rules stated in the singular/plural pairings. Here, we explain the decisions and methodology we applied to develop a noun pluralizer for Runyankore. We started with basic rules based on the standard singular/plural pairings according to the noun classes (NC) presented in Table 2.2 in Section 2.3.1 in Chapter 2, and investigated the following questions:

RQ1 (a.1). How well does the 'standard' noun class table of prefixes work for computationally pluralizing nouns in Runyankore?; and

RQ1 (a.2). Do the same exceptions to the standard identified for isiZulu exist in Runyankore, and are there also rules among them?

Section 3.1 states the research questions about noun pluralization in Runyankore and the resulting research contributions. In Section 3.2, we explain the rule-based approach we used to pluralize nouns. Section 3.3 details how the standard NC singular/plural pairings formed the pluralization algorithm. Section 3.4 explains the causes of exceptions to the standard pluralization (mass nouns, compound nouns, prefix exceptions, singular-only nouns, and noun phrases), and their solutions. In Section 3.5, we explain the phonological conditioning required during noun

pluralization. We present the results on the evaluation of the pluralizer in Section 3.6. The main contribution here is: a rule-based Runyankore noun pluralizer.

In this chapter, we extend the work published at the 17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016) [29] by presenting more detailed pluralization algorithms in Sections 3.2, and 3.4, as well as the details on handling phonological conditioning in Section 3.5, and the results from the evaluation of the pluralizer in Section 3.6.

3.2 Methodology

The design of a pluralizer based on the rules on noun pluralization can be done in two ways: through a purely syntactic/morphological analysis that examines the characters of the string of text; or through a semantic analysis using the meanings of the nouns. The first approach entails an analysis of the patterns of the characters of the noun using a regular expression applied to the beginning of the noun. However, there are several nouns in Runyankore that have similar prefixes but are pluralized differently [1], which would result in some errors in pluralization; for example, ***omu**ntu* 'person' is pluralized as ***aba**ntu*, whereas ***omu**ti* 'tree' in NC 3 is pluralized as ***emi**ti*.

In the second approach, some encoding of the meaning of the noun, or its noun class (NC), is stored with the noun. The noun is then associated with its NC, which enables the identification of the correct pluralization for nouns with the same prefixes but belonging to different NCs. Once the correct plural NC has been identified, a morphological or syntactic analysis is required to perform the pluralization.

Based on the benefits and drawbacks of each approach, we applied an approach that combines morphology with syntax and semantics. This involves storing each noun with its NC to cater for the semantics, and then syntactically or morphologically pluralizing using the appropriate plural prefix. The semantic approach, which involves storing the NC with the noun enables: (1) the identification of nouns with the same prefix but belonging to different NCs, such as *omuntu* 'person' in NC 1 and *omuti* 'tree' in NC 3; (2) differentiation among nouns that have the same spelling but different meanings and are in different noun classes, for example *omubazi*, which can either refer to 'accountant' (NC 1) or 'medicine' (NC 3); and (3) categorizing loan words into two groups, humans (NC 1) and others (NC 9), with the former pluralized as NC 2 and the latter pluralized as NC 10.

3.3 Standard Noun Class Pluralization

We started with a pluralizer based on the Runyankore noun class system singular/plural pairings shown in Table 3.1 (NC 14 is not included because it is a plural class, and NCs 16, 17, and 18 are also left out because they are locative classes that have no plural forms).

From the singular/plural pairings in Table 3.1, Algorithm 3.3.1 was developed to add them as rules in the pluralizer.

Algorithm 3.3.1 shows the pluralization rules for all singular NCs in Runyankore, including how classes with multiple prefixes, such as NC 5, are accounted for. However, not all nouns in Runyankore conform to the standard pluralization rules; these are treated as exceptions.

3.4 Pluralizing Exceptions

As explained in Section 3.1, the term 'exceptions' in the context of this research refers to nouns whose process of pluralization requires different computational rules from, or additional computational rules to the approach taken for basic rules stated in the singular/plural pairings. While

[1] For a list of NCs with the same class prefixes, see classes 1, 3, and 18; and 15 and 17 in Table 2.2 in Section 2.3.1.

TABLE 3.1: Standard singular/plural NC pairings for Runyankore

Singular NC	Plural NC
1 (*o-mu-*)	2 (*a-ba-*)
3 (*o-mu-*)	4 (*e-mi-*)
5 (*e-i-/e-ri-*)	6 (*a-ma-*)
7 (*e-ki-*)	8 (*e-bi-*)
9 (*e-n-/e-m-*)	10 (*e-n-/e-m-*)
11 (*o-ru-*)	10 (*e-n-*)
12 (*a-ka-*)	13 (*o-tu-*)
15 (*o-ku-*)	6 (*a-ma-*)
20 (*o-gu-*)	21 (*a-ga-*)

Algorithm 3.3.1 accounts for all singular noun classes in Runyankore, we had two reasons to suspect that the algorithm might not be able to pluralize all nouns. First, Maho [122] stated that a traditional singular/plural distinction is not sufficient for all nouns. Second, work on an isiZulu pluralizer identified exceptions from the standard pairings [29]. The exceptions identified were found to be either due to nouns that do not conform to the standard rules of pluralization, or due to the need for further preprocessing before pluralization [29]. The former group contains mass nouns, prefix exceptions, or nouns with no plurals; while the latter comprises compound nouns [29]. We also identified two types of noun phrases that can result from the name of the concept in an ontology: definitions (which contain a conjugated verb), and those with adjectives. Definitions occur whenever there is a concept in the ontology that cannot be directly translated to Runyankore, and a definition is used instead. The presence of adjectives on the other hand mainly results from the nature of the concept's name in the ontology. In the following sections, we explain our investigation into the rules to pluralize exceptions in Runyankore.

3.4.1 Compound Nouns

A compound noun is a noun composed of two or more nouns, where the main noun is modified by the other noun, such as 'schoolteacher' or 'cat food'. Compound nouns in Runyankore use the possessive concord to relate the main noun to the modifier noun, for example, *omwegyesa w'eishomero* 'teacher of school' and *ekyokurya ky'enjangu* 'food of cat', with possessive concords *wa* and *kya* respectively. The main challenge for pluralization here is the extra processing required, where the main nouns, *omwegyesa* 'teacher' and *ekyokurya* 'food', are first pluralized and then the plural possessive concord used with the modifier nouns. As explained in Section 2.3.1, the possessive concord is dependent on the NC[2], and pluralization thus requires extra steps to obtain it. We designed a novel algorithm to pluralize compound nouns in Runyankore, shown in Algorithm 3.4.1.

In the singular and plural forms of a compound noun whose modifier noun starts with a consonant, the possessive concord retains its ending vowel, and the three parts (main noun, possessive concord, and modifier noun) are written as three separate words. This is shown in lines 7 and 8 in Algorithm 3.4.1. If, however, the modifier noun starts with a vowel, the possessive concord drops its ending vowel and is assimilated into the modifier noun through a process called vowel assimilation (lines 9, 10, and 11 in Algorithm 3.4.1). When this happens, the ending vowel of the possessive concord is replaced with an apostrophe.

3.4.2 Prefix Exceptions

There are some singular nouns whose plural forms deviate from the standard NC pairings. These are referred to as prefix exceptions, and are true exceptions because they belong in a singular class A with singular/plural pairing A/B, yet their correct plural does not use the prefix of class B.

[2]The list of possessive concords is shown in Table 2.2 in Chapter 2.

Algorithm 3.3.1 Pluralization according to the standard singular/plural pairings

Variables: a_1, a_1'; Functions: $getNC(a_1)$, $dropSingularPrefix(a_1, prefix)$
$n \leftarrow getNC(a_1)$ ▷ Get the NC n of the noun
if $N == 1$ **then**
    $a_1' \leftarrow dropSingularPrefix(a_1, \text{'omu'})$ ▷ Drop the singular prefix from the stem of a_1,the noun
    Result $\leftarrow$ "abaa_1'" ▷ Pluralize with *aba*, the plural prefix in NC 2
else if $N == 3$ **then**
    $a_1' \leftarrow dropSingularPrefix(a_1, \text{'omu'})$ ▷ Drop the singular prefix from the stem
    Result $\leftarrow$ "emia_1'" ▷ Pluralize with *emi*, the plural prefix in NC 4
else if $N == 5$ **then**
    if $a_1.startsWith(\text{"}ei\text{"})$ **then**
        $a_1' \leftarrow dropSingularPrefix(a_1, \text{'ei'})$ ▷ Drop the singular prefix from the stem
        Result $\leftarrow$ "amaa_1'" ▷ Pluralize with *ama*
    else
        $a_1' \leftarrow dropSingularPrefix(a_1, \text{'eri'})$ ▷ Drop the singular prefix from the stem
        Result $\leftarrow$ "amaa_1'" ▷ Pluralize with *ama*
    end if
else if $N == 7$ **then**
    $a_1' \leftarrow dropSingularPrefix(a_1, \text{'eki'})$ ▷ Drop the singular prefix from the stem
    Result $\leftarrow$ "ebia_1'" ▷ Pluralize with *ebi*, the plural prefix in NC 8
else if $N == 9$ **then**
    if $a_1.startsWith(\text{"}en\text{"})$ **then**
        $a_1' \leftarrow dropSingularPrefix(a_1, \text{'en'})$ ▷ Drop the singular prefix from the stem
        Result $\leftarrow$ "ena_1'" ▷ Pluralize with *en*
    else
        $a_1' \leftarrow dropSingularPrefix(a_1, \text{'em'})$ ▷ Drop the singular prefix from the stem
        Result $\leftarrow$ "ema_1'" ▷ Pluralize with *em*
    end if
else if $N == 11$ **then**
    $a_1' \leftarrow dropSingularPrefix(a_1, \text{'oru'})$ ▷ Drop the singular prefix from the stem
    Result $\leftarrow$ "ena_1'" ▷ Pluralize with *en*, the plural prefix in NC 10
else if $N == 12$ **then**
    $a_1' \leftarrow dropSingularPrefix(a_1, \text{'aka'})$ ▷ Drop the singular prefix from the stem
    Result $\leftarrow$ "otua_1'" ▷ Pluralize with *otu*, the plural prefix in NC 13
else if $N == 15$ **then**
    $a_1' \leftarrow dropSingularPrefix(a_1, \text{'oku'})$ ▷ Drop the singular prefix from the stem
    Result $\leftarrow$ "amaa_1'" ▷ Pluralize with *ama*, the plural prefix in NC 6
else if $N == 20$ **then**
    $a_1' \leftarrow dropSingularPrefix(a_1, \text{'ogu'})$ ▷ Drop the singular prefix from the stem
    Result $\leftarrow$ "agaa_1'" ▷ Pluralize with *aga*, the plural prefix in NC 21
end if
return Result

Table 3.2 shows examples of some exceptions, their incorrect plurals according to the NC, and their correct plurals that deviate from the standard pairings.

These 'true' exceptions occur among nouns from different NCs, and their plurals, too, belong to different NCs. Due to the lack of rules that can be generalized among them, we hard-coded their plurals into a novel algorithm shown in Algorithm 3.4.2. More rules can be added to this algorithm as new true exceptions are identified.

Algorithm 3.4.1 Pluralization of compound nouns

Variables: N, a_1, a_1', g, g', g'', a_2, n'; Functions: $split(N)$, $pluralize(a_1, n')$, $getPluralNC(n)$, $getpossessiveconcord(n')$, $dropIV(g)$, $dropLV(g)$
$a[] \leftarrow split(N)$ ▷ Split the compound noun into its constituents: the main noun, possessive concord, and modifier noun
$n' \leftarrow getPluralNC(n)$ ▷ Get the plural NC
$a_1' \leftarrow pluralize(a_1, n')$ ▷ Pluralize the main noun
$g \leftarrow getpossessiveconcord(n')$ ▷ Get the possessive concord of the pluralized noun
$g' \leftarrow dropIV(g)$ ▷ The possessive concord drops its initial vowel
if $a_2.startsWithConsonant()$ **then**
    Result $\leftarrow$ "$a_1'\ g'\ a_2$" ▷ The pluralized compound noun contains the plural main noun and plural possessive concord
else
    $g'' \leftarrow dropLV(g')$ ▷ The plural possessive concord drops its ending vowel
    Result $\leftarrow$ "$a_1'\ g''' a_2$" ▷ The pluralized compound noun is composed of the plural main noun, and vowel assimilation between the plural possessive concord and modifier noun
end if
return Result

TABLE 3.2: Examples of prefix exceptions, showing expected and actual plural forms

English	Singular	Incorrect NC Plural	Deviant Correct Plural
Home	*Eka* (NC 9)	*Eka* (NC 10)	*Amaka* (NC 6)
Shop	*Eduka* (NC 9)	*Eduka* (NC 10)	*Amaduka* (NC 6)
Adolescent	*Omunyeeto* (NC 1)	*Abanyeeto* (NC 2)	*Eminyeeto* (NC 4)
Rabbit	*Orume* (NC 11)	*Enme* (NC 10)	*Obume* (NC 14)
Onion	*Orutunguru* (NC 11)	*Entunguru* (NC 10)	*Obutunguru* (NC 14)
month	*Okweezi* (NC 15)	*Ameezi* (NC 6)	*Emyeezi* (NC 4)

Algorithm 3.4.2 Pluralizing true exceptions

Variables: a_1; Functions: $pluralize(a_1, N)$
if $a_1.equals$('Eka') **then**
    Result $\leftarrow$ "Amaka" ▷ Pluralize with *Amaka*
else if $a_1.equals$('Eduka') **then**
    Result $\leftarrow$ "Amaduka" ▷ Pluralize with *Amaduka*
else if $a_1.equals$('Omunyeeto') **then**
    Result $\leftarrow$ "Eminyeeto" ▷ Pluralize with *Eminyeeto*
else if $a_1.equals$('Orume') **then**
    Result $\leftarrow$ "Obume" ▷ Pluralize with *Obume*
else if $a_1.equals$('Orutunguru') **then**
    Result $\leftarrow$ "Obutunguru" ▷ Pluralize with *Obutunguru*
else if $a_1.equals$('Okweezi') **then**
    Result $\leftarrow$ "Emyeezi" ▷ Pluralize with *Emyeezi*
else
    Result $\leftarrow pluralize(a_1, N)$ ▷ Pluralize according to the NC
end if
return Result

3.4.3 Singular-only Nouns

There are several nouns in Runyankore that belong to a singular NC, but by their nature have no plural. These include nouns that belong to a locative class (for example, *aheekiikire* 'secret place' in NC 16); abstract nouns such as *eiriho* 'thirst' in NC 5, *ekyanda* 'drought' in NC 7, or *omururu* 'greed'

in NC 3; and seasons, such as *akanda* 'autumn' in NC 12; as well as translations for proper nouns such as the days of the week (for example, *Orwokubanza* for 'Monday') that are all in NC 11.

Though there are no rules among these nouns, they can be generalized as nouns that should not be pluralized. We thus placed all such identified nouns in a separate lookup file that is searched before the pluralization process begins. A noun found in this list is returned without being pluralized.

3.4.4 Mass Nouns

Mass nouns refer to those entities that are only countable in certain quantities, but not as discreet objects. These nouns should not be pluralized, but left as they are. This is also true in Runyankore, for which there are two categories of mass nouns: those belonging to a singular NC, and those belonging to a plural NC. The latter present no challenge to pluralization because, by virtue of belonging to a plural NC, cannot be pluralized. Examples of such nouns include: *amaizi* 'water' and *amate* 'milk' in NC 6, or *obushera* 'porridge' in NC 14. A rule stating that any noun belonging to a plural NC should be returned as it is was added to the pluralizer (lines 2 and 3 in Algorithm 3.4.3).

The mass nouns belonging to singular NCs present the challenge of identifying if and when they should be pluralized. This is because they can be further subdivided into two groups: (1) singular-only mass nouns; and (2) mass nouns that can be pluralized in certain senses. Examples of nouns found in group (1) are *orwondo* 'mud' in NC 11 or *omugati* 'bread' in NC 3. Group (2) contains nouns such as *eitaka* 'soil' or 'land' and *eigufa* 'bone' in NC 5, which are in some senses pluralizable (to *amataka* and *amagufa* respectively). In these senses, they are singular nouns that can be pluralized according to the NC singular/plural pairings, but in some senses are mass nouns and should not be pluralized. For both singular and plural groups, we used the same identification strategy as used for the isiZulu pluralizer [29] marking all mass nouns that should not be pluralized with an '**m** on the noun class (NC), and a new rule using the '**m** as a marker was added to the pluralizer (lines 4 and 5 in Algorithm 3.4.3).

Algorithm 3.4.3 Pluralizing mass nouns

1: Variables: a_1, N'; Functions: $pluralize(a_1, N')$
2: **if** $N == 2 \mid 4 \mid 6 \mid 8 \mid 10 \mid 13 \mid 14 \mid 16 \mid 17 \mid 18 \mid 21$ **then**
3: Result $\leftarrow$ "a_1" ▷ Return original noun if NC is plural
4: **else if** $N.contains(\text{'m'})$ **then**
5: Result $\leftarrow$ "a_1" ▷ Return original noun if NC is marked with an 'm'
6: **else**
7: Result $\leftarrow$ $pluralize(a_1, N')$ ▷ Pluralize according to the NC
8: **end if**
9: **return** Result

3.4.5 Pluralizing Noun Phrases

Noun phrases are a special case in pluralization because they contain other grammatical categories, which are also affected by the process of pluralization. Table 3.3 shows examples of the two types of noun phrases we identified: those with verbs and those with adjectives.

We thus categorized these two kinds of noun phrases separately because they are handled differently. When pluralizing definitions, the subject concord of the pluralized noun is required to conjugate the verb in order to obtain the correct plural noun phrase. However, when adjectives are present, the adjective concord of the pluralized noun is required to additionally pluralize the adjective. In both cases, the NC is needed to obtain the subject and adjective concords. As there is currently no morphological analyzer for Runyankore, a similar identification strategy as for mass

TABLE 3.3: Examples of noun phrases resulting from definitions and adjectives

Source	Example	Runyankore
No direct translation	Pet	***En**yamaishwa **e**rikutungwa abantu kuzaanisa*
	Opiate	***Omu**bazi **gu**kusinza ogukwejunisibwa kukyendeza obusaasi*
Nature of concept's name	OldLady	***Omu**kazi **mu**kuru*
	BroadSheet	***Eki**papura **ki**hango*

nouns was used to differentiate between definitions and adjective noun phrases. This involved adding a '**v** or '**j** to the NC of a definition or adjective noun phrase respectively. We developed Algorithm 3.4.4 to handle this.

Algorithm 3.4.4 Identifying noun phrases with verbs or adjectives

Variables: a_1, a_1', N, N'; Functions: $getNC(a_1)$, $pluralizeWithAdjective(a_1, N')$, $pluralizeWithVerb(a_1, N')$
if $N.contains(\text{'v'})$ **then**
    $N' \leftarrow getNC(a_1)$ ▷ Obtain the NC from the noun
    Result $\leftarrow pluralizeWithVerb(a_1, N')$ ▷ Pluralize as definition that contains verb
else if $N.contains(\text{'j'})$ **then**
    $N' \leftarrow getNC(a_1)$ ▷ Obtain the NC from the noun
    Result $\leftarrow pluralizeWithAdjective(a_1, N')$ ▷ Pluralize as noun phrase with adjective
else
    Result $\leftarrow pluralize(a_1, N)$ ▷ Pluralize according to the NC
end if
return Result

The details of the pluralization process undertaken for each category are presented below.

Definitions

A definition is composed of the subject and the conjugated verb. The conjugated verb requires the subject concord[3] of the subject for its correct form. For example, in the definition of 'pet' in Table 3.3 as 'an animal obtained by people for leisure', the subject *enyamaishwa* 'animal' in NC 9 and the conjugated verb ***e**rikutungwa* 'obtained' are the focus for pluralization. The subject concord of the singular subject in the conjugated verb (***e***) is from NC 9. After pluralization, however, the subject concord of the plural subject (NC 10 -*zi*- in this case) is used instead, thus making the conjugated verb ***zi**rikutungwa*. Algorithm 3.4.5 shows the steps taken to pluralize definitions.

Algorithm 3.4.5 Pluralizing noun phrases containing a conjugated verb

Variables: A, n, n', a_1, c'; Functions: $split(A)$, $getPluralNC(n)$, $pluralize(a[0], n')$, $getPluralSC(n')$
$a[] \leftarrow split(A)$ ▷ Split the noun phrase into its constituents
$n' \leftarrow getPluralNC(n)$ ▷ Obtain the plural NC
$a_1 \leftarrow pluralize(a[0], n')$ ▷ Pluralize the noun (subject)
$c' \leftarrow getPluralSC(n')$ ▷ Use the plural nc to obtain the plural subject concord
Result $\leftarrow$ "$a_1\ c'a[1]\ a[2]$ ..." ▷ The pluralized noun phrase contains the plural noun, and verb conjugated with the plural subject concord **return** Result

[3]For a list of subject concords and their respective NCs, see the column for 'Subject Concord' in Table 2.2 in Chapter 2.

Noun Phrases Containing Adjectives

When pluralizing a noun phrase that contains one or more adjectives, for example *omukazi mukuru* 'old lady' in NC 1, first the noun (*omukazi*) is pluralized (to *abakazi* in NC 2), and then the plural adjective concord is obtained and used to pluralize the adjective (from *mukuru* to *bakuru*). These steps are shown in Algorithm 3.4.6.

Algorithm 3.4.6 Pluralizing noun phrases containing adjectives

1: Variables: A, n', a_1, d', a_2; Functions: $split(A)$, $getPluralNC(n)$, $pluralize(a[0], getPluralAdj(n'), pluralizeADJ(d')$
2: $a \leftarrow split(A)$ ▷ Split the noun phrase into its constituents
3: $n' \leftarrow getPluralNC(n)$ ▷ Obtain the plural NC
4: $a_1 \leftarrow pluralize(a[0], n)$ ▷ Pluralize the noun
5: $d' \leftarrow getPluralAdj(n')$ ▷ Obtain the plural adjective concord
6: $a_2 \leftarrow pluralizeAdj(d')$ ▷ Pluralize the adjective
7: Result $\leftarrow$ "a_1 a_2" ▷ The pluralized noun phrase contains the plural noun, and plural adjective
return Result

If the noun phrase contains more than 1 adjective (for example, 'tall, beautiful tree'), then line 6 in Algorithm 3.4.6 is repeated for each adjective in the noun phrase.

3.5 Phonological Conditioning

Phonological conditioning is the change of a morpheme because of the phonological features of the morphemes surrounding it, such as 's' versus 'es' or 'in' versus 'im' in English [142]. In Runyankore, during pluralization, the plural prefix is concatenated with the stem. In some cases (for example, when the stem starts with a vowel, or an 'n' followed by an 'r' or 'p') the resulting plural text contains letter combinations that do not exist in Runyankore phonology. We used phonological rules to make the required changes that reflect the sound change. This is referred to as phonological conditioning [122].

We found the need for phonological conditioning evident across NCs, and it can be achieved either through vowel coalescence (adding an extra vowel), vowel elision (deleting a vowel), or by deleting or adding a consonant. Table 3.4 shows examples of how the need for phonological conditioning arises, and how it is corrected for.

TABLE 3.4: Examples of the need for, and solutions to phonological conditioning ('VC' and 'VE' refer to vowel coalescence and vowel elision, respectively)

Singular	Plural	Cause	Solution	Result
Omwegi (student)	*Abaegi*	*ae* in *ab**ae**gi*	VE on *a*, then VC of *e*	*Abeegi*
Omworo (poor person)	*Abaoro*	*ao* in *ab**ao**ro*	VE on *a*, then VC of *o*	*Abooro*
Omwaka (year)	*Emiaka*	*ia* in *Em**ia**ka*	VE on *i*, replaced with *y*	*Emyaka*
Oruha (intestine)	*Enha*	*nh* in *e**nh**a*	*nh* replaced with *nd*	*Enda*
Orupapura (paper)	*Enpapura*	*np* in *En**p**apura*	*np* replaced with *mp*	*Empapura*
Ororimi (language)	*Enrimi*	*nr* in *e**nr**imi*	*nr* replaced with *nd*	*Endimi*
a Akaisho (tiny eye)	*Obuisho*	*ui* in *ob**ui**sho*	VE on *u*, replaced with *y*	*Obwisho*

We developed Algorithm 3.5.1 which contains the rules for phonological conditioning.

Note that the noun is first pluralized, and then corrected for using phonological conditioning, if necessary.

Algorithm 3.5.1 Performing phonological conditioning

Variables: I, P, v, n, n'; Functions: $pluralize(a)$, $replace(p_1, p_2)$
$I = \{$'ae', 'ao', ', 'au', ('iv', 'ua', 'ue', 'ui', 'uo', 'nh', 'nr', 'np', 'nb', 'nm'$\}$ ▷ An array of all the incorrect orthography, where v is all vowels
$P = \{$'ee', 'oo', ', 'uu', ('yv', 'wa', 'we', 'wi', 'wo', 'mp', 'nd', 'mp', 'mb', 'm'$\}$ ▷ An array of all the phonologically conditioning corrections
$a' \leftarrow pluralize(a)$ ▷ First pluralize the noun
if $a'.contains(I(i))$ **then**
    Result $\leftarrow replace(I(i), P(i))$ ▷ For each element i, correct with its corresponding correction
end if
return Result

3.6 Evaluation of Pluralizer

We developed the Runyankore pluralizer as a Java application[4]. In order to evaluate how well the algorithms developed capture the rules we identified, we evaluated the correctness of the output with a Runyankore first-language speaker who was not involved in the development of the model or the software. We continued to evaluate as we identified and added new rules to the pluralizer. The details of the evaluation process are explained below.

3.6.1 Materials and Methods

After the development of the first version of the pluralizer based on Algorithm 3.3.1, we manually compiled three word lists and used them to test the pluralizer. The first, **Set1**, was composed of 92 nouns, obtained by translating a list of random English words collected from multiple ontologies. This list was created to test the isiZulu pluralizer [29], and initially comprised 101 nouns. However, only 88 were translatable to Runyankore, and we added four to include NC 20 and special orthographical cases. This word list thus accounted for all the singular NCs in Runyankore.

The second word list, **Set2**, was composed of 2,543 nouns that we obtained by attempting to automatically extract every noun from Tailor's Runyankore dictionary [176]. We used this second set to more comprehensively test the pluralizer after testing with **Set1** achieved 100% accuracy. **Set2** accounts for all NCs except 20 and 21, because the placement of nouns in these classes is common in speech rather than writing, since they are augmentative classes [8].

We used **Set3** to test the rules for the pluralization of noun phrases, and it was composed of 20 noun phrases: 10 made up of definitions, and 10 made up of noun phrases with adjectives. Unlike **Set1**, which we obtained by translating from English to Runyankore, we compiled **Set2** and **Set3** from Runyankore sources. The accuracy of the pluralizer was calculated as the percentage of correctly pluralized nouns for each test set.

We first used **Set1** to test the pluralizer, and as we identified incorrect plurals, we added more rules to the pluralizer, and then tested again. After achieving 100% accuracy with this dataset, we tested further with **Set2**, with the identification of incorrect plurals resulting in adding more rules and retesting. Finally, we targeted **Set3**–testing the pluralization of noun phrases.

3.6.2 Results and Analysis

Table 3.5 shows the results of testing the pluralizer with the 3 datasets, calculated as a percentage of words in the list that were pluralized correctly. The improvement in the accuracy of the pluralizer as new rules were added can be seen, and they are presented in the table in the order in which they were added.

[4]The Runyankore noun pluralizer can be accessed at https://github.com/ThesisResources/RunyankorePluralizer

TABLE 3.5: Results from testing the pluralizer on the three datasets ('PC' refers to phonological conditioning, and 'NP' refers to noun phrase)

Pluralizer Version	**Set1**	**Set2**	**Set3**
Whole words	88	-	-
Words + NC	92	-	-
Words + NC + loan words	97	-	-
Words + NC + loan words + PC	98	-	-
Words + NC + loan words + PC + compound nouns	99	67	-
Words + NC + loan words + PC + compound nouns + plural exceptions	100	71	-
Words + NC + loan words + PC + compound nouns + plural exceptions + singular-only	100	74	-
Words + NC + loan words + PC + compound nouns + plural exceptions + singular-only + prefix exceptions	100	74.2	-
Words + NC + loan words + PC + compound nouns + plural exceptions + singular-only + prefix exceptions + mass	100	81	-
Words + NC + loan words + PC + compound nouns + plural exceptions + singular-only + prefix exceptions + mass + proper nouns	100	85	-
Words + NC + loan words + PC + compound nouns + plural exceptions + singular-only + prefix exceptions + mass + proper nouns + more PC	100	94	-
Words + NC + loan words + PC + compound nouns + plural exceptions + singular-only + prefix exceptions + mass + proper nouns + more PC + NP	100	98	100

The results from the evaluation of the pluralizer shown in Table 3.5 demonstrate the importance of the new rules identified during pluralization. Associating a noun with its NC during pluralization improved the accuracy of **Set1** from 88% to 92%, due to correctly pluralizing nouns with the same prefix but belonging to different NCs. Additionally, for **Set2**, which was a more comprehensive dataset, the inclusion of proper nouns in the lookup file containing nouns that should not be pluralized increased the accuracy from 81% to 85%. The proper nouns in the dataset can be categorized as: names of the days of the week (for example *Orwokubanza*, in NC 11, for 'Monday'); names of languages (for example *Orujungu*, in NC 11, for 'English'); names of the months in the year (for example *Biruuru*, in NC 9, for 'January'); and names of countries (for example *Bungyereza*, in NC 9, for 'Britain'). Further, the effect of the phonological conditioning described in Section 3.5 is clearly depicted in the results of **Set2**, where accuracy improved from 85% to 94%.

One of the most notable findings of this evaluation was that the preferred plural for NC 12 is NC 14, instead of NC 13 as stated according to Table 2.2 in Chapter 2. This was first raised by the evaluator, and we further confirmed that pluralizing NC 12 as 13 is so irregular that no NC 13 plurals exist in the Runyankore dictionary [176]. Instead, for every noun in NC 12, its associated plural is given in NC 14 [176]. While both NC 12 and 13 contain diminutive nouns, pluralizing NC 12 (for example *akasiisi* for 'small ant') as NC 13 (*otusiisi* for 'very very tiny ants') is highly irregular, because, semantically, it diminishes the object even further, as compared to pluralizing with NC 14 (*obusiisi* for 'small ants'). Consequently, throughout our research, we use NC 14 as the plural for NC 12 for Runyankore. Adding this rule to the pluralizer improved the accuracy of the results of **Set2** from 67% to 71%.

Further, adding the rules for the pluralization of noun phrases described in Sections 3.4.5 and 3.4.5 also had an impact on the results of **Set2**, though **Set3** was deliberately compiled for this purpose. The accuracy of the pluralizer increased from 94% to 98% for **Set2**.

Algorithm 3.4.4 was first applied to the datasets in order to distinguish between noun phrases with verbs, and those with adjectives. Table 3.6 shows illustrative results from different NCs on pluralizing definitions when Algorithm 3.4.5 was used on **Set3**.

Table 3.7 shows some results of pluralizing **Set3** using Algorithm 3.4.6, including those noun phrases with several adjectives.

TABLE 3.6: Results from pluralizing definitions

English	Singular	Plural
Medicine that intoxicates as a means of pain reduction	***Omu**bazi **gu**kusinza ogukwejunisibwa kukyendeza obusaasi*	***Emi**bazi **gi**kusinza ogukwejunisibwa kukyendeza obusaasi*
A lid that is brightly colored	***Eki**fundikiso **ki**rikutukura*	***Ebi**fundikiso **bi**rikutukura*
An animal obtained by people for leisure	***En**yamaishwa **e**rikutungwa abantu kuzaanisa*	***En**yamaishwa **zi**rikutungwa abantu kuzaanisa*
A lecturer taking 4 courses	***Omu**shomesa **o**rikushomesa amashomo ana*	***Aba**shomesa **ba**rikushomesa amashomo ana*

TABLE 3.7: Results from pluralizing noun phrases containing adjectives

English	Singular	Plural
Old person	***Omu**ntu **mu**kuru*	***Aba**ntu **ba**kuru*
Big beautiful seed	***Eki**juma **ki**hango **ki**rungi*	***Ebi**juma **bi**hango **bi**rungi*
Thin tall tree	***Omu**ti **mu**kye **mu**raingwa*	***Emi**ti **mi**kye **mi**raingwa*
Thin tall tiny tree	***Aka**ti **ka**kye **ka**raingwa*	***Obu**ti **bu**kye **bu**raingwa*
Big tall man	***Omu**shaija **mu**hango **mu**raingwa*	***Aba**shaija **ba**hango **ba**raingwa*

In Table 3.7, it can be seen that the NC of the noun changes the adjective both in the singular and plural forms, for example, 'big ', 'tall', and 'small'. The algorithm is also able to pluralize a noun phrase with several adjectives, as is the case with 'seed', 'tree', 'tiny tree'[5], and 'man'.

While the pluralizer was able to achieve 100% on both **Set1** and **Set3**, there were still 49 incorrect pluralizations with **Set2**. These remaining cases can be placed in two categories: (1) noun phrases with definitions that require more morphological analysis than is currently supported; and (2) some hyphenated and compound words that take several forms and the rules for their pluralization are not clear[6].

Results from Pluralizing Exceptions

Table 3.8 shows some examples of the results of applying Algorithm 3.4.1 to singular compound nouns in the dataset[7].

TABLE 3.8: Examples of singular compound nouns and their plurals (the class prefix and its possessive concord are bolded)

English	Singular	Plural
President	***Omw**ebembezi **w**'eihanga*	***Abe**ebembezi **b**'eihanga*
Newspaper	***Oru**papura **rw**'amakuru*	***Em**papura **z**'amakuru*
Cup handle	***Omu**kono **gw**'ekikopo*	***Emi**kono **y**'ekikopo*
Eye sore	***Aka**ronda **ky**'erisho*	***Obu**ronda **bw**'erisho*
Rainmaker	***omw**igi **w**'enjura*	***aba**igi **b**'enjura*

[5]Diminutive objects are placed in NC 13, and so there is no separate adjective for 'tiny' in 'tiny tree'. Rather, the stem for 'tree', *-ti*, is used with the prefix of NC 12, *aka-*, instead of the prefix of NC 3, *omu-*. The translation for 'tiny tree' thus becomes *akati*. This would be the same if the noun referred to a 'tiny girl, cup, or car', in which case the translations would be *akaishiki*, *akakopo*, *akamotoka*, respectively, all in NC 12, instead of *omwishiki* (NC 1), *ekikopo* (NC 7), and *emotoka* (NC 9), respectively.

[6]Hyphenated nouns in Runyankore take many forms, and the general rules for their pluralization are not clear, but rather assessed on a case by case basis. For example, some are proper nouns, such as *omubamba-njobe* 'small prickly tree', and so are not pluralized; while others, like *ow'omu-ruganda* 'kinsman', start with the possessive concord *ow'*, and this is the only part of the noun which is pluralized, resulting in *ab'omu-ruganda* 'kinsmen'; and others still, like *omuhwa-nshoni* 'a person with no shame', can be pluralized, based on the prefix of the first word, to *abahwa-nshoni*.

[7]Note the effect of phonological conditioning when pluralizing ***omw**ebembezi*, which is changed from ***aba**ebembezi* to ***abe**ebembezi*.

Prefix exceptions seem to be very rare in Runyankore, as only six were identified from a test set of 2,543 nouns.

3.7 Discussion

The identification of rules for pluralization addresses one of the prerequisites for verbalization in RQ1 in Section 1.2.1 in Chapter 1. Question RQ1 (a.1) in Section 3.1 was concerned with assessing how well the standard noun class (NC) pairings work for pluralization. We have shown that, based on the kinds of nouns likely to be found in the ontologies to be verbalized, the NC is crucial to the pluralization process, but, in order to be applied in a computational context, the standard NC pairings require more rules. The same result was found for isiZulu [29].

The development of this pluralizer required the use of a combined syntactic, morphological, and semantic approach: the semantic approach to identify the kind of noun and how it should be correctly pluralized; the morphological to replace the singular prefix with its corresponding plural for simple nouns; and the syntactic for compound nouns and noun phrases. The semantic approach was required because of the presence of nouns with the same singular prefix, but belonging to different NCs, and are therefore pluralized differently. This was also found to be the case for isiZulu [29].

Question RQ1 (a.2) in Section 3.1 was concerned with whether the same exceptions identified for noun pluralization in isiZulu are present in Runyankore, and whether there are rules among them. We found that the pluralization of compound nouns, prefix exceptions, singular-only nouns, and mass nouns in Runyankore requires additional rules, as was found for isiZulu [29]. In Section 3.4, we showed the algorithms that pluralize compound nouns by obtaining the possessive concord, using a separate lookup file for singular-only nouns, referring to a list of hard-coded correct plurals for prefix exceptions, and adding an identifier to the noun class to identify mass nouns.

We also developed algorithms to pluralize noun phrases with definitions and noun phrases with adjectives, as the name of a concept in an ontology can result in these two categories. In both cases, the NC is required to obtain the concords with which to pluralize adjectives and verbs. The final step in the pluralization process is phonological conditioning, to ensure correct orthography. The rules and algorithm are presented in Section 3.5. The improvement in the accuracy of the pluralizer as new rules are added to cater for all these cases is shown in Section 3.6.

When compared to the generation of noun plurals in English, our work is similar to early approaches such as that presented by Minnen, Carroll, and Pearce [134] in three main ways: (1) both start by using a morphological approach to pluralize nouns, and this usually works for many nouns; (2) phonological conditioning is required in both languages (for example, if a noun ends in 'f' or 'fe', then the plural form replaces it with 'ves', such as from 'wol**f**' to 'wol**ves**' and 'wi**fe**' to 'wi**ves**'); and (3) not all nouns can be correctly pluralized using the same rules, and these, also termed 'exceptions' (for example, from 'che**f**' to 'che**fs**' and 'stimul**us**' to 'stimul**i**'), are addressed using different rules or placed in a look-up file [134]. However, there are two significant differences between these languages: (1) pluralization in Runyankore is performed on the noun prefix while it is the suffix in English [134]; and (2) English does not have a complex system of noun classification that determines several grammatical processes including noun pluralization. Current approaches are statistical and thus data-driven, as explained in Section 2.2.1, but these have not been applied to Runyankore due to its under-resourced state.

On the other hand, in the work to pluralize nouns in isiZulu, the crucial role of the noun classes (NCs) during pluralization is evident in the improvement in accuracy from 53% for Set1 and 45% for Set2 to 92% and 78% respectively [29]. We therefore hypothesize that this approach to noun pluralization may be generalizable to other Bantu languages with an agglutinating morphology. We shall consider this more closely in Chapter 8.

3.8 Summary

In this chapter, we explained how we solved one of the prerequisites of verbalization, noun pluralization. We found that noun pluralization requires an approach that combines morphology with syntax and semantics, which we first based on the standard NC singular/plural pairings, and then extended to handle exceptions. These exceptions include compound nouns, prefix exceptions, singular-only nouns, mass nouns, and noun phrases. Additionally, we presented the rules for the phonological conditioning required after pluralization. We evaluated the pluralizer with three datasets, two of which achieved 100%, and the third, 98%.

Having completed and evaluated the pluralization component of verbalization, we next address the second prerequisite, verb conjugation.

Chapter 4

Verb Conjugation in Runyankore

4.1 Overview

Our first research question, RQ1, 'How can noun pluralization and verb conjugation be achieved in Runyankore' in Section 1.2.1 in Chapter 1 is concerned with how to address the prerequisites to verbalization of noun pluralization and verb conjugation. The previous chapter presented our findings on the first part of this question, RQ1 (a), regarding noun pluralization. In this chapter, we present our findings on the research into the second part of this question, RQ1 (b), regarding verb conjugation. In Section 2.4.1, we explained that in ontologies roles denote binary relationships between individuals [9, 10, 59]. These roles may be expressed as verbs, whose verbalization requires verb conjugation. In this chapter, we present the methodology undertaken to achieve verb conjugation in Runyankore. Our research was based on the following question: are Context-Free Grammars sufficient to conjugate verbs in Runyankore?

We begin this chapter by providing details on the verbal morphology of Runyankore in Section 4.2, in order to properly explain how tense and aspect are expressed in the language. In Section 4.3, we explain the corpus-based analysis that led to the selection of the progressive aspect, simple present tense, participial present continuous tense, and the passive as sufficient for our target domain. Section 4.4 then explains why we found Context-Free Grammars (CFGs) to be sufficient to conjugate verbs in Runyankore, and we show the CFGs for standard verbs and deviations from the standard. We provide details on phonological conditioning during verb conjugation in Section 4.5, and present the results of the evaluation of the CFGs by two linguists in Section 4.6. The main contributions in this chapter are:

- The identification of the appropriate tense and aspect for healthcare messages in Section 4.3; and
- The approach to verb conjugation for Runyankore using CFGs in Section 4.4.

In this chapter, we extend the work published in the 9th and 10th International Conference on Natural Language Generation [25, 26] by selecting the tense and aspect from a Runyankore corpus instead of an English one in Section 4.3, providing details on phonological conditioning during verb conjugation in Section 4.5, and providing details on evaluating the conjugated verbs in Section 4.6. Throughout this chapter, we rely on Turamyomwe's thesis on tense and aspect in Runyankore [177]. Additionally, though we use healthcare examples throughout this chapter to show how verbs can be generated for our domain of interest, the verb conjugation aspects are applicable to other verbs in other domains.

4.2 Runyankore Verbal Morphology

In Runyankore, and indeed other Bantu languages [145], the standard categories of tense–past, present, and future–are further subdivided for each category [177]. For example, the past and future tenses are subdivided to express the degree of remoteness from the present: the past tense,

into immediate past, near past, and remote past; and the future tense, into near future and remote future [177]. This, together with the participial forms, results in a total of fourteen tenses.

Aspect, on the other hand, focuses on the internal nature of events, as opposed to their grounding in time [145, 177]. There are two major aspects in Runyankore: the perfective and imperfective; the imperfective is further subdivided into the persistive, habitual, and continuous aspects, with the progressive as a subtype of the continuous [177]. It is also generally accepted that Runyankore expresses tense as prefixes and aspect as affixes, and is generally depicted by encoding tense to the left of the verb stem and aspect to the right, though not always [177].

Table 4.1 from Turamyomwe [177] shows the different 'slots' in Runyankore's verbal morphology, as well as the morphemes which occupy these slots[1].

TABLE 4.1: Verbal morphology of Runyankore; App: applicative, Cs: causative, Ps: passive, Rec: reciprocal, Rev: reversive, Stv: stative, Itv: intensive, Red: reduplicative, Ism: instrumental [177]

Slot	**Grammatical Category**	**Morpheme**
pre-initial	1. primary negative 2. continuous marker	1. *ti-* 2. *ni-*
initial	subject concord	depends on the NC as shown in Table 2.2
post-initial	secondary negative	*-ta-*
formative	tense	all tenses except near past
limitative	persistive aspect	*-ki-*
infix	object concord	depends on NC
extentions	App; Cs; Ps; Rec; Rev; Stv; Itv; Red; Ism	*-er-, -erer-, -ir-; zi-, -is-; -w-; -n-; -ur-, -uur-; -gur-; repeat the stem; -is + pre-initial*
final	1. final vowel (a) indicative (b) subjunctive 2. near past tense	1. (a) *-a* (b) *-e* 2. *-ire*
post-final	1. locative 2. emphatic 3. declarative	1. *-ho, -mu-yo* 2. -ga 3. *-nu*

The depiction of the verbal morphology shown in Table 4.1 does not imply that the tense and aspect markers are always limited to one slot [177]. Additionally, some of the slots are optional: some can co-occur, while others, like the continuous aspect and negation in the 'pre-initial', cannot [177]. The compulsory slots that must always be filled are the 'initial', 'formative' (except in the case of the universal and near past tenses), 'verb-root', and 'final' [177]. Table 4.1 shows that the 'initial' contains the subject concord of the subject in the sentence, the 'formative' encodes all tenses except the universal and near past, and the 'final' either holds the indicative or subjunctive vowel, or the near past tense.

An illustration of the complex structure of the Runyankore verbal morphology is presented in the example below, where 'neg' is negation, 'RM' is remote past, 'VR' is verb-root, 'App' is applicative, 'FV' is final vowel, 'Loc' is locative, 'Emp' is emphatic, and 'Dec' is declarative.

Runyankore: *titukakimureeterahoganu*

English: 'We have never ever brought it to him'

[1] According to Oxford Bibliographies Online, Oxford Research Encyclopedia of Linguistics, and Oxford Handbooks Online, the 'applicative' is used to express an indirect participant as a direct participant; the 'causative' is used to express an event and what or who caused it; the 'passive' is used to express the object as the subject; the 'Reciprocal' is used to express the notion of mutuality using 'each ... other' or 'one ... another'; the 'reversive' is used to express the overturning of an action; the 'stative' is used to express the state of the subject rather than the action being done; the 'intensive' is used to express more pressure of the action than is expressed by the root verb; the 'Reduplicative' is used to express meaning by repeating all or part of a word; and the 'Instrumental' is used to mark the semantic role of an instrument.

Morphemes: *ti-tu-ka-ki-mu-reet-er-a-ho-ga-nu*

Grammatical Units: neg-(NC2 SC)-RM-(NC7 SC)-(NC1 SC)-VR-App-FV-Loc-Emp-Dec

4.3 Tense and Aspect in Prescription Explanations

Due to the complex verbal morphology of Runyankore, we decided to narrow our scope to the tense and aspect that would be necessary for the domain of interest, healthcare. We further limited the scope to explanations of prescribed medications because, as explained in Section 2.5, providing patients with extra information leads to patients making more informed decisions about their prescribed treatments [52, 54].

To the best of our knowledge, there is no prior work specifically discussing tense and aspect for explanations of prescribed medications. In order to research this for Runyankore, we collected and manually analyzed text describing drug prescriptions from empirical studies and transcribed corpora, [16, 17, 41, 45, 46], in order to identify the tense and aspect used. We limited our analysis to these authors because other literature regarding drug prescriptions focused on presentation style, the effects of including information on side effects, and simplifying applied terminologies.

Berry, Gillie, and Banbury [16] and Berry et al. [17] looked into what kind of information patients wanted to know about their prescribed medication through a series of empirical studies. They had participants make lists of what kind of questions they would want to ask the doctor about a prescription, and then categorized the questions and used the categories to generate explanations, which were then presented to participants for validation [16]. In a follow-up study that also generated drug explanations from the doctors' perspectives [17], both patients and doctors were asked to rate which informational categories they preferred. Five categories were identified as important by both patients and doctors: side effects, what the medication does, lifestyle changes, how to take the medication, and interaction with prescribed medication [16, 17]. Below are examples from the corpora for each category:

(a) Fennodil has no known side effects, and is not addictive.

(b) It reduces the amount of acid in your stomach allowing your ulcer to heal, and relieves painful symptoms.

(c) It is safe to drink alcohol, drive, or operate machinery while taking this medication.

(d) You should always take Fennodil with a glass of milk.

(e) These tablets are safe to take with any other prescribed medication you may be taking.

De Carolis et al. [41] and De Rosis and Grasso [45] presented text transcribed from doctors' recordings of prescription explanations based on hypothetical cases. Some excerpts are shown below:

Prescription for Angina:

(a) All the tests show that you have three separate problems. The first is the chest pain, which we know to be angina, that is, heart pain, which is very similar to the cramp you get in your legs when you walk too far.

(b) This combination of problems necessitates treatment with some special tablets that I'm going to give you.

(c) The first tablet is simple aspirin, which has been shown to help people with angina, and reduce the risk of you having a heart attack.

Prescription for Tuberculosis:

(a) Mr. Smith, unfortunately, you have an infection of the chest, which is called tuberculosis.

(b) The problem with this infection is it takes a long time to eradicate it from the body, and therefore we have to undertake quite a long course of treatment, which is essential for you to fulfill for the full course.

(c) Therefore, initially, we are going to start off with three, and we may even add four drugs when we talk to the microbiologist. These drugs will have to be taken every day, and then we will reassess the situation in three months time.

From the corpus by De Carolis et al. [41] and De Rosis and Grasso [45], we observed that the focus was on explaining the diagnosis rather than the prescription. On the other hand, the corpora from Berry, Gillie, and Banbury [16] and Berry et al. [17] focused on the explanations of the prescribed medications, which is what we were interested in. We therefore did not include the corpus from De Carolis et al. [41] and De Rosis and Grasso [45] in the final analysis.

4.3.1 Analysis for Tense and Aspect

We considered 33 sentences from [16, 17] for our analysis of tense and aspect describing medication prescriptions. These were translated by a linguist in their entirety to Runyankore[2], though our analysis is limited only to the tense in the main clause, because it can provide complete meaning and contains the tense of the entire sentence. For example, from the above excerpts, only, 'It is safe to drink alcohol' (in sentence (c)) is considered, which in the translated corpus is *Tikiine kabi kunywa amaarwa*[3].

We then performed a manual analysis on the corpus, focusing on the form of the verb, in order to identify the tense and aspect used. Table 4.2 shows how often each verb form occured in each **unique sentence** in the corpus. 'Unique sentence' is emphasized here because some sentences are duplicated in different categorizations of the corpus, sometimes with slight modifications, for example: (1) 'Fennodil and Nozydril tablets are safe to take ...'; and (2) 'These tablets are safe to take ...' [17]. In such situations, these sentences were counted as one unique sentence, which resulted in a final analysis of 26 sentences: 18 from [17] and 8 from [16].

TABLE 4.2: Tense and aspect used in prescription explanations

Example	Constitution	Tense, Aspect	Incidence
tigwine	ti-gw-in-e	simple present subjunctive negation	6
neekyendeeza	n-ee-kyendeez-a	simple present indicative, progressive	3
nikiragiirwa	ni-ki-ragiir-w-a	simple present indicative + passive, progressive	5
kaine	ka-in-e	simple present subjunctive	3
gutakyendeeze	gu-ta-kyendeez-e	participial near future negation, perfective	1
bukozirwe	bu-koz-ir-w-e	participial near future (applicative + passive), perfective	2
otakoreise	o-ta-kore-is-e	participial near future + causative, perfective	1
tiburikurahuka	ti-bu-riku-rahuk-a ku-kor-a	participial present continuous negation	1
oyerinde	o-ye-rind-e	participial near future, perfective	2
tihariho	Ti-ha-ri-ho	simple present + locative, progressive	1
tibugumire	ti-bu-gum-ire	near past negation, progressive	1

During the analysis, we regarded verbs conjugated in the same tense and aspect but with different subject concords as having the same tense and aspect. For example, *ka-in-e*, *o-in-e*, and *e-in-e* are in the same tense and aspect (simple present subjunctive), with the difference in the conjugated verb originating from the subject concords, which thus resulted in a count of three.

[2]The initial analysis done on the English corpus can be found in [25].

[3]The Runyankore and English corpora can be accessed from https://github.com/ThesisResources/TNADatasets

From Table 4.2, the simple present tense is used in 69.23% of the corpus: in 30.77% of cases with the indicative, in 34.62% with the subjunctive, and in 19.23% with the passive. Additionally, it can be seen that with the simple present tense, the subjunctive in this corpus is only used when the verb-root is *-in-* 'has'. From these results, we selected the tense and aspect that had a cumulative count greater than five in the corpus, resulting in the selection of the simple present tense with the progressive aspect, the passive, and indicative and subjunctive final vowels where appropriate.

Comparison With English Corpus Analysis In [25], we carried out a similar manual analysis to identify the tense and aspect on the same corpora from [16, 17]. However, this analysis was carried out on the English, not Runyankore, corpus. We found that the simple present tense was used in 55.5% of the corpus, as compared to 69.23% for the Runyankore corpus. The indicative in the simple present tense was used in 48.2% of the English corpus, but 30.77% in the Runyankore one. The use of the subjunctive in the simple present tense is more evident in the Runyankore corpus (34.62%) as compared to 7.7% in the English corpus.

4.4 Context-Free Grammars for Verb Conjugation

Similar to Bantu languages, other agglutinative languages place importance on the placement of morphemes in a word and rules governing the combinations of morphemes to form semantic categories [87]. Additionally, the sequence of their morphemes can also express mood, tense, and aspect, as well as sense of assertion, negation, interrogation, emphasis, and reflection [87, 151, 177].

We therefore first investigated the existing tools for verb conjugation of agglutinative languages [7, 87, 132, 148, 168], like Tamil [151] and Kannada [168]. The existing approaches that handle verb conjugation during text generation in agglutinative languages can be classified as: corpus-based, paradigm-based, Finite-State Transducer (FST) based, rule-based, and algorithm-based [7].

Not all of these approaches are applicable for our purposes. Runyankore's under-resourced state renders the corpus-based approach impractical, while the paradigm-based approach is not compatible with Runyankore's verbal morphology. On the other hand, the FST, algorithm, and rule-based approaches can be applied to Runyankore's verbal morphology. However, while rule-based approaches are typically implemented independently, the FST and algorithm-based approaches are typically combined with other approaches in an implementation, for example, the two-level morphology [80, 106], the combined FST and rule-based [132], FST and paradigm-based [148], or the FST and algorithm and paradigm-based [109]. This makes implementations based on the FST and algorithms more complicated to implement than a rule-based approach. Additionally, Context-Free Grammars (CFG), a rule-based grammar formalism, has been shown to handle complex verb conjugation in isiZulu [95].

We thus applied a rule-based approach, specifically, a Context-Free Grammar (CFG) to conjugate verbs for tense and aspect in Runyankore. A context-free grammar G is a tuple defined by four parameters: $G = N\ \sigma\ R\ S$, where N is a finite set of non-terminals, variables that express generalizations of words; σ is a set of terminal symbols corresponding to words in the language and are disjoint from N; R is a set of rules or productions expressing how symbols of the language can be grouped and ordered; and S is the designated start symbol [188]. Sentences are generated by recursively rewriting the start symbol using the productions until only terminal symbols remain [91].

CFGs are powerful enough to depict complex relations among words in a sentence, yet computationally tractable enough to enable efficient algorithms to be developed [188]. We thus applied a CFG based on the phrase-structure grammar [75], as a sentence generator through the derivation of a string of words [188]. In this case, some of the grammatical categories presented in Section 4.2

formed the non-terminals in the CFG, and only those associated with the identified tense and aspect categories (the simple present tense, progressive aspect, and the passive) are considered. The simple present tense has no special tense marker, and is sometimes called the 'null tense' [177].

The CFG comprised the following grammatical slots as non-terminals:

(a) **Pre-initial** for the negation (*ti*) and progressive aspect (*ni*) morphemes;

(b) **Initial** for all subject concords;

(c) **Formative** for the tense marker;

(d) **Verb-root**;

(e) **Extensions** for the passive;

(f) **Final** for the final vowel; and

(g) **Initial group** to represent the production to the 'pre-initial' and 'initial'.

We assigned these seven non-terminals the symbols: IG for initial group, PN for pre-initial (the progressive aspect), IT for initial, FM for formative (the tense marker), VR for verb-root, EX for extensions (the passive), and FV for final vowel (indicative and subjunctive). Additionally, throughout this chapter, we refer to the verbs whose conjugation adheres to the four 'compulsory' slots explained in Section 4.2 as 'standard', and those that omit one or more of these compulsory slots as 'deviations from the standard'.

The CFG below shows the production rules for conjugating standard verbs in Runyankore (verb-roots: *kyendeez* (reduce), *mir* (swallow), *vug* (drive), *reeb* (see), *shom* (read), *gamb* (speak), and *twar* (take)).

$S \rightarrow IG\ FM\ VR\ EX\ FV$
$IG \rightarrow PN\ IT$
$PN \rightarrow$ ti | ni
$IT \rightarrow$ a | o | n | tu | mu | ba | gu | gi | ri | ga | ki |
bi | e | zi | ru | tu | ka | bu | ku | gu | ga
$FM \rightarrow \emptyset$
$VR \rightarrow$ kyendeez | mir | vug | reeb | shom | gamb | twar
$EX \rightarrow$ w | er | erer | ir | zi | is | n | ur | uur | gur |
VR | isPN
$FV \rightarrow$ a | e | ire

The production of IT has several possible values, depending on the noun class of the subject of the sentence. The **formative** is $\emptyset$ because the simple present tense has no tense morpheme. Further, the **extensions** grammatical slot can take several values for the passive, applicative, causative, reciprocal, reversive, stative, intensive, reduplicative, and instrumental, as shown in Table 4.1. Finally, for all verbs, except 'has' and 'to be', FV will always be the indicative final vowel 'a'. The above CFG thus generates verbs like *nibatwara* 'they take', *tizikyendeeza* 'they do not reduce', and *nizivugwa* 'they are driven by'.

The CFG is responsible for verb conjugation of the roles in an ontology. The following examples show the derivation of several verbs, and are related to examples of drug prescription explanations in the corpus by [16, 17]. We only show the derivation of the verb here, and cover the entire verbalization process in the next chapter.

(a) Fennodil $\sqsubseteq \exists$ **reduces**.Pain

Verb-root: *kyendeez*]

Derivation: $S \Rightarrow IG\ FM\ VR\ EX\ FV$
$\Rightarrow PN\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow n\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow nee\ FM\ VR\ EX\ FV \Rightarrow neeVR\ EX\ FV$
$\Rightarrow neekyendeez\ EX\ FV$
$\Rightarrow neekyendeez\ FV \Rightarrow neekyendeeza$

(b) Fennodil $\sqsubseteq \neg \exists$ **causes**.Ulcer

Verb-root: *reet*, and requires negation
Derivation: $S \Rightarrow IG\ FM\ VR\ EX\ FV$
$\Rightarrow PN\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow t\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow te\ FM\ VR\ EX\ FV$
$\Rightarrow teVR\ EX\ FV$
$\Rightarrow tereet\ EX\ FV \Rightarrow tereet\ FV$
$\Rightarrow tereeta$

(c) Fennodil $\sqsubseteq \exists$ **taken**With.Milk

Verb-root: *twar*, and requires the passive
Derivation: $S \Rightarrow IG\ FM\ VR\ EX\ FV$
$\Rightarrow PN\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow n\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow nee\ FM\ VR\ EX\ FV \Rightarrow neeVR\ EX\ FV$
$\Rightarrow neetwar\ EX\ FV \Rightarrow neetwarw\ FV$
$\Rightarrow neetwarwa$

4.4.1 Deviations from Standard Conjugation

In the context of this research, we define 'deviations' as those verbs whose conjugation does not use one or more of the four compulsory slots explained in Section 4.2. We found two verbs which deviate from the standard Runyankore grammar: the auxiliary 'has' (verb-root *-in-*) and the copulative 'to be' (verb-root *-ri-*). They however deviate in different ways, and we thus used separate CFGs for each of them, in order to prevent the generation of words that do not exist in the language.

CFG for the Auxiliary

The auxiliary deviates from the standard grammar in four ways:

(a) the continuous marker is dropped;

(b) there is never a tense morpheme;

(c) not all the extensions apply; and

(d) the subjunctive final vowel *e* is used.

This results in the following CFG:
$S \rightarrow IG\ FM\ VR\ EX\ FV$
$IG \rightarrow PN\ IT$
$PN \rightarrow$ ti
$IT \rightarrow$ a | o | n | tu | mu | ba | gu | gi | ri | ga | ki |
bi | e | zi | ru | tu | ka | bu | ku | gu | ga
$FM \rightarrow \emptyset$

$VR \rightarrow$ in
$EX \rightarrow$ w | is | VR
$FV \rightarrow$ e

The above CFG can therefore generate verbs like *baine* 'they have', *tigwine* 'it does not have', or *ziinwe* 'they are with', expressing 'had' in the passive). Additionally, having a separate CFG for the auxiliary prevents the generation of verbs like *nibaine* or *tigurikwine*, which would otherwise be generated by the standard CFG.

An example derivation of this CFG, based on the corpus from [16, 17], is for the negation of 'have' in the axiom, Fennodil $\sqsubseteq \neg \exists$ **has**.SideEffect, would be: $S \Rightarrow IG\ FM\ VR\ EX\ FV \Rightarrow PN\ IT\ FM\ VR\ EX\ FV \Rightarrow t\ IT\ FM\ VR\ EX\ FV \Rightarrow te\ FM\ VR\ EX\ FV \Rightarrow teVR\ EX\ FV \Rightarrow tein\ EX\ FV \Rightarrow tein\ FV \Rightarrow teine$

CFG for Copulative

The copulative deviates even further from the standard grammar than the auxiliary. In addition to dropping the continuous marker and having no tense morpheme, the copulative has no extensions and does not apply *a* or *e* as its final vowel. The CFG below is thus used in this case:

$S \rightarrow IG\ FM\ VR\ EX\ FV$
$IG \rightarrow PN\ IT$
$PN \rightarrow$ ti
$IT \rightarrow$ a | o | n | tu | mu | ba | gu | gi | ri | ga | ki |
bi | e | zi | ru | tu | ka | bu | ku | gu | ga
$FM \rightarrow \emptyset$
$VR \rightarrow$ ri
$EX \rightarrow \emptyset$
$FV \rightarrow \emptyset$

This CFG can generate verbs like *gari* 'they are', *ari* 'he or she is', and *tiguri* 'it is not', and prevents the generation of verbs that do not exist in the language, such as *nibaria* from the standard CFG, and *gurie* from the auxiliary CFG.

The use of the copulative during verbalization is not immediately apparent from the axiom, as is the case for the standard and the auxiliary. Rather, it arises from the verbalization of axioms with complex roles, such as this below:
Axiom: Fennodil $\sqsubseteq \exists$ **hasSideEffect**.Diarrhea
English: 'Each Fennodil has at least one side effect, **which is** diarrhoea'
Runyankore: 'Buri Fennodil eine hakiri ekirikurugamu kitagyendereirwe kimwe ***kiri*** okwirukana'
Derivation: $S \Rightarrow IG\ FM\ VR\ EX\ FV$
$\Rightarrow PN\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow\ IT\ FM\ VR\ EX\ FV$
$\Rightarrow ki\ FM\ VR\ EX\ FV \Rightarrow kiVR\ EX\ FV$
$\Rightarrow kiri\ EX\ FV \Rightarrow kiri\ FV \Rightarrow kiri$

4.5 Phonological Conditioning

The phonological conditioning required after verb conjugation is due to agglutination. We used a different phonological conditioning module to account for the rules for verbs, separate from that for nouns in Section 3.5, because not all the rules identified for nouns were computationally applicable to verbs. In the case of verb conjugation, the corrective measures are also vowel coalescence, vowel elision, or deleting or adding consonants. Table 4.3 presents examples of the circumstances that create the need for phonological conditioning (source), the letter combinations that are affected (causes) and their corresponding corrective measures (solutions) for verbs. We

hard-coded them into the application, to check for and perform phonological conditioning of the verb returned by the CFG.

TABLE 4.3: The need for and addressing phonological conditioning during verb conjugation ('VC' is vowel coalescence, 'VE' vowel elision, 'CM' continuous marker, and 'SC' is subject concord)

Generated Verb	Cause	Solution	Result
Niagenda	*ia* resulting from CM followed by vowel-only SC	VE on *i*, and VC of SC (for vowel harmony)	*N**aa**genda*
Tierya	*ie* from negation and vowel-only SC	VE on *i*	*Terikurya*
Niaegyesa	*iae* from CM, vowel-only SC, and verb-root starting with vowel	VE on *i*, VC on SC, and *y* between SC and verb-root	*N**aay**egyesa*
nigucwwa	*ww* from verb-root ending in *w*, *s*, or *z* and the passive	Replace passive's *w* with *ibw*	*Nigucw**ibw**a*

It should be noted that vowel harmony is only required if the subject concord is a single vowel. This is the case for *a* in NC 1 and *e* in NC 9 (see Table 2.2 in Chapter 2), and can be seen in the example derivations from the standard and auxiliary CFGs. We required a separate phonological conditioning module for the case of verb conjugation because some of the rules for nouns in Section 3.5 can result in incorrect orthography for this case. For example, the correction for vowel harmony from *ni**ia**genda* to *n**aa**genda* would have incorrectly been done as *n**y**agenda* using the rules in Section 3.5 (because *i* followed by a vowel is always changed to *y*).

4.6 Evaluation of CFG Output

We evaluated the output from the three CFGs shown above with the assistance of two linguists. We purposefully selected linguists for this evaluation because verb conjugation requires knowledge about the Runyankore verbal morphology, which non-linguists typically do not have. The details of how the evaluation was done are presented below.

4.6.1 Materials and Methods

We used the CFG Java tool by Xu, Zheng, and Zhen [188] to implement the verb conjugation module as a Java application. Their CFG implementation extends Purdom's algorithm to fulfill Context-Dependent Rule Coverage (CDRC), with the benefits of generating more and simpler sentences [188]. The verb conjugation module comprises the three CFGs and the phonological conditioning module. We generated 117 verbs from the three CFGs for the evaluation. The following were taken into account when generating and selecting the verbs:

(1) The standard verbs, auxiliary, and copulative;

(2) Simple present tense with progressive aspect;

(3) Negation;

(4) The passive;

(5) All the subject concords, except the locative classes 16, 17, and 18; and

(6) The need for phonological conditioning.

We selected 99 verbs from the list of generated verbs by omitting verbs from NCs with the same subject concord that resulted in duplicates being generated. For example, both NC 6 and NC 21 have *-ga-* as their subject concord, while NC 3 and NC 20 have -gu- as their subject concord. This resulted in duplicate verbs, which were removed from the final list given to the linguists.

We presented the selected 99 verbs to two linguists in a questionnaire that can be found in Appendix A. The linguists were instructed to identify the output as either being: (1) grammatically incorrect verbs; (2) grammatically correct except for the need for phonological conditioning; or (3) correct verbs, but would be better presented in another form. They were instructed to use an asterisk (*) to flag these verbs, and to write an explanation after it. Any verbs not flagged were thus regarded as correct. Where there were discrepancies between the linguists' choices, we would inform them about this, and ask whether they agreed with the assessment from the other linguist. We only accepted a decision where there was consensus between the linguists. There was no case where they were still in disagreement after the clarification process.

4.6.2 Results and Analysis

From the first linguist, L1, 11 of the 99 verbs were explicitly flagged as falling into one of the three categories presented above; while the second linguist, L2, explicitly identified 24 as being erroneous. There were also some verbs that were not flagged, but the linguist instead noted the error at the bottom of the questionnaire in an explanatory paragraph. Table 4.4 shows the number of verbs flagged for each category, the reasons given, and the linguist who identified them.

TABLE 4.4: Results of the evaluation of conjugated verbs by linguists ('PC' is 'Phonological Conditioning' and 'NBNEI' is 'Noted, But Not Explicitly Identified')

Category	**Reason**	**Number by L1**	**Number by L2**
1. Incorrect Verb	Incorrect subject concord Incorrect phonological conditioning	1 -	1 1
2. Need for PC	PC through vowel coalescence for vowel harmony	8	NBNEI
3. Recommendations	PC with extra suffix for copulative and vowel SC Participial present continuous tense preferred for formal Runyankore	2 -	- 22

Several reasons were given by the linguists during the evaluation, some explicitly, and others in an explanation at the bottom of the questionnaire. Table 4.4 shows that there were some verbs flagged by one linguist but not another. For example, L1 flagged a verb as having an incorrect subject concord, but L2 did not. We contacted L2 for further clarification, and L2 explained that in some cases, where a category of an error was highlighted on one word, it was not repeated when found again on a different word, as it was assumed that we would understand that the error was true wherever it existed. We thus had agreement between both linguists in all cases.

Besides correcting errors like those for category 1 in Table 4.4, this evaluation is important because three main issues were made apparent: (1) the need for phonological conditioning for vowel harmony whenever the subject concord is a vowel; (2) the recommendation that whenever the subject concord is a vowel, the suffix *ya* should preferably be used with the copulative *ri*; and (3) the use of the participial present continuous tense (*riku*) during negation in formal written Runyankore, because, as explained by L2, the use of the participial present tense *ku* is colloquial.

The application of the required phonological conditioning rules was done as presented in Section 4.5. We also updated the standard CFG in Section 4.4 to account for the participial present continuous tense. This only required adding its tense marker *riku* as an option for the **formative** grammatical slot, as below:

$FM \rightarrow \emptyset \mid$ riku

The conjugation of the verb when verbalizing the axiom, Fennodil $\sqsubseteq \neg \exists$ **causes**.Ulcer, thus becomes:

Verb-root: *reet*, and requires negation

Derivation: $S \Rightarrow IG\ FM\ VR\ EX\ FV$

$\Rightarrow PN\ IT\ FM\ VR\ EX\ FV$

$\Rightarrow t\ IT\ FM\ VR\ EX\ FV$

$\Rightarrow$ *te FM VR EX FV* $\Rightarrow$ *terikuVR EX FV*
$\Rightarrow$ *terikureet EX FV* $\Rightarrow$ *terikureet FV* $\Rightarrow$ *terikureeta*

However, the auxiliary and copulative do not use this tense during negation, and there was therefore no need to update their CFGs.

4.7 Discussion

The question under investigation in this chapter was whether CFGs are sufficient to conjugate verbs in Runyankore. We found verb conjugation in Runyankore to be complicated, due to Runyankore's complex verbal morphology comprising 14 tenses, six aspects, and nine extensions (presented in Section 4.2). The complexity also results from the rules about how the grammatical slots are arranged during conjugation, as well as which morphemes can co-occur and which cannot [177]. We therefore narrowed our scope to the tense and aspect used in drug prescription explanations, in line with our domain of interest, healthcare. However, the CFGs presented above can be used to generate verbs in other domains. Additionally, a CFG can be applied to generate more complex verbs, as shown by Keet and Khumalo [95], who used a CFG to generate verbs in isiZulu, covering the subject and object concords, present tense, negation, aspect, mood, four extensions (causative, applicative, stative, reciprocal), politeness, wh-question modifiers, and aspect doubling. The resulting grammar was shown to be applicable for a range of applications, and a computational evaluation of the grammar showed that it was correctly specified [95].

We carried out a manual analysis of a corpus of drug prescription explanations in Section 4.3. From the corpus analysis, we identified the simple present tense, progressive aspect, and passive; while we used the participial present continuous tense during negation, based on the evaluation by linguists in Section 4.6.

The results on the identified tense and aspect are interesting because they reflect the tense and aspect used in roles in an ontology. These roles usually have names such as 'causes', 'treats', 'takenWith', or 'causedBy', which are either in the simple present tense or the passive, as shown in the axioms used to demonstrate the derivations of the CFG. The application of a similar tense and aspect in the verbalizations should thus correlate with the semantics expressed in the ontology, as shown in the derivations throughout Section 4.4.

We showed that some verbs in Runyankore can be generated from a CFG, and seven nonterminals were sufficient for the selected tense and aspect. CFGs have commonly been used to generate entire sentences in English [91, 188] and verbs in English are also conjugated for tense, aspect, mood, and agreement with the subject [64, 134]. Additionally, phonological conditioning is also performed in English; for example, the use of 'consonant doubling' in the past and present continuous tenses (such as, from 'submit' to 'submi**tt**ed' or 'submi**tt**ing') [134]. However, early rule-based English NLG systems such as the generator by Minnen, Carroll, and Pearce [134] used a look-up file for such verbs. But for Runyankore, due to our limited domain, we are able to generate the verbs, though a separate phonological conditioning module is used for verbs.

4.8 Summary

In this chapter, we undertook first steps towards solving the problem of verb conjugation in Runyankore, which is the second prerequisite to verbalization. In order to identify the appropriate tense and aspect for the healthcare domain, we undertook a manual analysis of a corpus of drug prescription explanations. From this analysis, we identified the simple present tense, progressive aspect, and passive; while the participial present continuous tense was recommended for verb negation by the linguists. We used a CFG for verb conjugation, where we limited the nonterminals to seven. We showed three CFGs: for the standard grammar, the auxiliary, and the copulative; the last two being deviations from the standard grammar. We further explained the

phonological conditioning rules specific to verb conjugation, and presented the results of the evaluation of the CFGs done by two linguists, which showed that most of the verbs generated by the CFG were grammatically correct.

Having solved both prerequisites to verbalization, noun pluralization in Chapter 3 and verb conjugation in this chapter, the next chapter explains how we proceeded to answer RQ2 by verbalizing the DL $\mathcal{ALCQ}$.

Chapter 5

Runyankore Verbalizations of $\mathcal{ALCQ}$ Constructors

5.1 Overview

In Chapters 3 and 4, we presented the research to answer RQ1 in Section 1.2.1 concerned with addressing noun pluralization and verb conjugation as prerequisites to verbalization. This chapter presents our work on the second research question, RQ2, 'What are the Runyankore verbalization patterns for the selected $\mathcal{ALCQ}$ constructors' in Section 1.2.1 in Chapter 1. The work in this chapter focuses on ontology verbalization, which, as explained in Section 2.4.1, involves expressing as natural language the semantics found in description logics (formal syntax specifying how to construct well-formed sentences based on formal semantics that relate those sentences to a model [10]). In order to undertake this, we needed to answer the following questions:

RQ2 (a). Which description logic constructors should be verbalized in order to generate healthcare messages in Runyankore?

RQ2 (b). What factors affect verbalization in Runyankore?

RQ2 (c). How can these factors be applied to develop verbalization patterns for the selected constructors?

In Section 5.2, we explain how we selected the constructors to verbalize based on the drug prescription corpus from [16, 17], and in Section 5.3, explain how we identified the factors affecting verbalization, which form the basis for the verbalization patterns identified and presented thereafter. In Section 5.6, we present the results on the evaluation done with non-linguists, who are also non-experts in ontologies, from whom we selected which verbalization to implement.

In this chapter, we extend the work published at the 5th Workshop on Controlled Natural Language (CNL 2016) [24] by presenting the verbalizations in Section 5.3 independent of bootstrapping, as well as providing more details regarding the verbalization of complex roles, negation of roles during verb conjugation in Section 5.3.5, and phonological conditioning of the nasal compound in Section 5.5. The main contributions in this chapter are:

- The identification of the factors affecting verbalization in Runyankore; and
- The verbalization patterns for the selected constructors.

5.2 Selected DL Constructors

To recap, in Section 2.4.1, we explained that there are several description logics, each specifying a different level of expressiveness. In order to identify which DL to verbalize, we developed logical theories from the same corpora used to identify the tense and aspect in Section 4.3. In order to do this, we first converted the sentences in the corpus by [16, 17] to a more restrictive form

called Controlled Natural Language (CNL). We specifically used Attempto Controlled English (ACE), which, as explained in Section 2.4.2, is a CNL used in ontology editors and reasoners [108]. We did this in order to avoid the ambiguity inherent in natural language, and thus ensure the deterministic interpretation of the resulting statements. We then removed repeating sentences, which left us with 16 sentences[1] that we used to form DL axioms. Table 5.1 shows examples of the CNL, the DL axiom formed, and the constructors in that axiom.

TABLE 5.1: The original, ACE, and DL axiom extracted from the drug prescription corpus

CNL	**DL Axiom**	**Constructor**
Fennodil is a well proven drug	Fennodil $\sqsubseteq$ WellProvenDrug	Simple Named Class Subsumption
Fennodil and Nozydril	Fennodil $\sqcap$ Nozydril	Conjunction
Each Nozydril is not addictive	Nozydril $\sqsubseteq \neg$ Addictive	Negation of Subsumption
All Fennodil reduce some stomach acid	Fennodil $\sqsubseteq \exists$ reduces.StomachAcid	Existential Quantification
Each Nozydril is not taken with some alcohol	Nozydril $\sqsubseteq \neg \exists$ takenWith.Alcohol	Negation of Roles
All Fennodil are only taken with milk	Fennodil $\sqsubseteq \forall$ takenWith.Milk	Universal Quantification
Each tablet has exactly 20 mg of Flavotine	Tablet $\sqsubseteq =$ 20 hasMGFlavotine	Exact Cardinality
All Fennodil act within a maximum of a week	Fennodil $\sqsubseteq \geq$ 1 actsWithIn.Week	Maximum Cardinality
All tablets are taken at least 8 hours apart	Tablet $\sqsubseteq \leq$ 8 taken.HoursApart	Minimum Cardinality

The constructors we found, shown in Table 5.1, are: subsumption ($\sqsubseteq$) and its negation ($\not\sqsubseteq$), conjunction ($\sqcap$), negation of roles ($\neg$), existential quantification ($\exists$), universal quantification ($\forall$), maximum cardinality ($\leq$), minimum cardinality ($\geq$), and exact cardinality ($=$). These eight constructors are from the DL $\mathcal{ALCQ}$ (see Section 2.4.1).

5.3 Verbalization of $\mathcal{ALCQ}$

Having identified the constructors to verbalize, we developed natural language descriptions from these constructors, from which we identified that verbalization is affected by six factors:

(1) The noun class of the concept's name;

(2) The concept's category;

(3) Whether the concept is atomic or an expression;

(4) The quantifier used in the axiom;

(5) The position of the concept in the axiom; and

(6) The number restriction in the axiom

Based on these factors, we observed patterns in the verbalizations. The following sections show, with examples and algorithms, the verbalization patterns for each constructor. Whenever there is a need for pluralization or verb conjugation, we simply state this as a step in the algorithm with no further illustration, as the details have already been presented in Chapters 3 and 4 respectively.

[1] All the 16 CNL statements can be found in Appendix B.

5.3.1 Simple Named Class Subsumption ($\sqsubseteq$)

Subsumption ($\sqsubseteq$) represents a taxonomic relationship, with the sub-class to the left, and the super-class to the right. Subsumption is verbalized in English as 'is a', which is *ni* in Runyankore. If however the name of the super-class starts with a vowel, then a process called vowel assimilation occurs, where the *i* in *ni* is dropped and replaced with an apostrophe, and the two are combined into a single word.

We voiced the universal quantification in the verbalization of $\sqsubseteq$, which is either verbalized as 'each' or 'all', which in Runyankore are *buri* or *-ona* respectively. *-ona* requires the possessive concord of the NC of the name of the sub-class, in order to form the full translation for 'all', and the NC of the name of the sub-class is required to obtain the possessive concord. On the other hand, *buri* is always *buri*, regardless of the NC of the name of the sub-class. It however can only be followed by a singular noun, which drops its initial vowel. We only used 'all' when the name of the sub-class and/or super-class was in the plural form or was a mass noun. The examples below show the verbalization of 'is a', 'each', 'all', and vowel assimilation (the underlined portions highlight were the possessive concord is required in Axiom2, which is not the case when *buri* is used in Axiom1):

Axiom1: Panado $\sqsubseteq$ Medicine

English: **Each** Panado **is a** medicine.

Runyankore: *Buri Panado **n'**omubazi*

Axiom2: Pain $\sqsubseteq$ Symptom

English: **All** pain **is a** symptom

Runyankore: *Obusaasi bwo**ona n'**emicucumo*

It can be seen that *buri* is placed before the sub-class, while *-ona* is placed after it. Additionally, in Axiom2, the possessive concord of *obusaasi* 'pain' is *bwa*. In the resulting concatenation, *bwa* + *ona* = *bw**ao**na*, and *ao* becomes *oo* as a result of the phonological conditioning described in Section 3.5. On the other hand, Axiom1 requires no possessive concord because *buri* is used.

In Algorithm 5.3.1, we show all the steps taken to implement the pattern identified in the verbalization of $\sqsubseteq$, which are: universal quantification, check for the grammatical form, negation, and pluralization. The details of the phonological conditioning are left out of this algorithm and presented later in this chapter.

If the subsumption is followed by negation ($\sqsubseteq \neg$), that is, disjointness, then it is verbalized as 'is not a' in English, which is *ti* in Runyankore. The other difference between the verbalization of subsumption and of its negation is that, if the name of the concept after $\neg$ starts with a vowel, instead of performing vowel assimilation, the name drops its initial vowel, and the two remain separate words. This follows with the literature on the initial vowel or augment that can be eliminated in certain grammatical constructions, such as when the noun follows a negative verb [94, 122]. The following examples show these two situations:

Axiom1: Diabetes $\sqsubseteq \neg$ Malaria

English: Diabetes **is not** Malaria

Runyankore: *Endwara ya shukari **ti** Malaria*

Axiom2: Pill $\sqsubseteq \neg$ Sedative

English: Pill **is not a** sedative

Algorithm 5.3.1 Verbalization of simple named class subsumption ($\sqsubseteq$)

1: Axiom A; Variables: a_1, n_1, a_2, n_2, d, d', g_1; Functions: $getSubClass(A)$, $getSuperClass(A)$, $dropLV(d)$, $pluralize(a_2, n_2)$, $getNC(a)$, $isSingular(n)$, $getSubClasspossessiveConcord(n_1)$
2: $a_1 \leftarrow getSubClass(A)$ ▷ Get the sub-class in the axiom
3: $a_2 \leftarrow getSuperClass(A)$ ▷ Get the super-class in the axiom
4: $n_1 \leftarrow getNC(a_1)$ ▷ Get the NC of the sub-class
5: $n_2 \leftarrow getNC(a_2)$ ▷ Get the NC of the super-class
6: **if** $checkNegation(A) == true$ **then**
7: Result $\leftarrow disjointness(a_1, a_2)$ ▷ This is a function call to the disjointness function in Algorithm 5.3.2
8: **else**
9: **if** $a_1.isSingular(n_1)$ **then**
10: **if** $a_2.isSingular(n_2)$ **then**
11: **if** $a_2.startsWithVowel$ **then**
12: $d' \leftarrow dropLV(d)$ ▷ *ni* drops its last vowel *i*
13: Result $\leftarrow$ "buri a_1 d''a_2" ▷ Verbalize $\sqsubseteq$ with *buri* and n'
14: **else**
15: Result $\leftarrow$ "buri a_1 d a_2" ▷ Verbalize $\sqsubseteq$ with *buri* and *ni*
16: **end if**
17: **end if**
18: **else**
19: **if** $a_2.isSingular(n_2)$ **then**
20: $a_2 \leftarrow pluralize(a_2, n_2)$ ▷ Pluralize the super-class
21: **end if**
22: $g_1 \leftarrow getSubClasspossessiveConcord(n_1)$
23: **if** $a_2.startsWithVowel$ **then**
24: $d' \leftarrow dropLV(d)$ ▷ *ni* drops its last vowel *i*
25: Result $\leftarrow$ "a_1 $g_1$ona d''a_2" ▷ Verbalize $\sqsubseteq$ with *-ona* and n'
26: **else**
27: Result $\leftarrow$ "a_1 $g_1$ona d a_2" ▷ Verbalize $\sqsubseteq$ with *-ona* and *ni*
28: **end if**
29: **end if**
30: **end if**
31: **return** Result

Runyankore: *Akajuma **ti** mubazi gurikugwejagisa*

In Axiom2, the translation for 'sedative' is a definition: *omubazi gurikugwejagisa* (medication that makes one fall asleep). Here *omubazi* drops its initial vowel because of negation, and becomes *mubazi*.

In Algorithm 5.3.2, we show the function for the pattern for negation of subsumption. It is called in line 6 of Algorithm 5.3.1 if line 4 is true.

The verbalization of $\sqsubseteq$ is thus affected by two factors: the NC of the concept's name and the category of the concept. There are two general categories into which the concepts can be placed: (1) both are singular, in which case *buri* is used; or (2) either one or both are plural or a mass noun, in which case *-ona* is used, and pluralization may be required. Additionally, we further subdivide Category 1 into concepts whose names start with a vowel, and those which do not. This is used to determine whether to use *ni* or n' for subsumption, or whether a concept's name should drop its initial vowel for the negation of subsumption. The need for the NC arises when *-ona* is used, as shown in the examples and algorithms. Further, checking whether a noun is singular or plural

Algorithm 5.3.2 Verbalization of the negation of subsumption ($\sqsubseteq \neg$)

```
1:  Variables: a_1, a_2, n, a_2'; Functions: dropIV(a_2)
2:  function DISJOINTNESS(a_1, a_2)
3:      if a_2.startsWithVowel then
4:          a_2' ← dropIV(a_2)          ▷ The concept to the RHS drops its initial vowel
5:          Result ← "a_1 n a_2'"       ▷ Verbalize ⊑ ¬ with ti
6:      else
7:          Result ← "a_1 n a_2"        ▷ Verbalize ⊑ ¬ with ti
8:      end if
9:      return Result
10: end function
```

(line 7 in Algorithm 5.3.2) is done by checking the NC of the noun[2]. If the sub-class is plural but the super-class singular (as is the case with: Pain $\sqsubseteq$ Symptom, above), then the super-class is first pluralized, followed by the verbalization with *-ona*. If both are plural, then the verbalization is more straightforward, with the possessive concord obtained, and *-ona* used.

5.3.2 Conjunction ($\sqcap$)

Conjunction ($\sqcap$) is verbalized as 'and'. In Runyankore, its verbalization depends on whether 'and' is used for a list of things or to connect clauses. For the former, *na* is used when the list comprises only nouns. In the latter case, *kandi* is used, and this also applies when a list contains adjectives. The examples below show the different circumstances under which *na* and *kandi* are used.

Simple Nouns: Disease $\sqcap$ Symptom

Runyankore: *Endwara* ***n'omucecemo***

Adjectives: Large $\sqcap$ Strong

Runyankore: *Bihango* ***kandi*** *by'amaani*

Clauses: $\ldots \exists$ reads.NewsPaper $\sqcap \ldots \exists$ drinks.Milk

Runyankore: *... hakiri naashoma orupapura rw'amakuru rumwe* ***kandi*** *... hakiri naanywa amate*

For the above examples, the verbalization of $\sqcap$ in English will always be 'and'. In Runyankore, however, we see how the verbalization changes based on the context of $\sqcap$: clauses and adjectives use *kandi* while nouns, regardless of form, use *na*. The examples on simple and compound nouns also show when vowel assimilation is necessary. In Algorithm 5.3.3, we show the steps in the pattern to verbalize conjunction:

The verbalization of $\sqcap$ is thus affected by two factors: (1) whether the name of the concept can be categorized as a noun or adjective, and (2) whether the concept is atomic or an expression. If the concepts are atomic, and both are nouns, then either *n'* or *na* is used depending on whether the name of the concept starts with a vowel or not respectively. If either of the concepts is an adjective, then $\sqcap$ is verbalized as *kandi*. Additionally, if the concept is an expression, then *kandi* is also used.

5.3.3 Existential Quantification ($\exists$)

Existential quantification ($\exists$) is verbalized as either 'some' or 'at least one'. In Runyankore, both 'some' and 'at least' are translated as *hakiri*, while 'one' is *-mwe*. The relative concord[3] of the

[2]The categorization of the singular and plural noun classes is presented in Table 3.1 in Chapter 3.
[3]For a list of the relative concords and their NCs, refer to Table 2.2 in Chapter 2.

Algorithm 5.3.3 Verbalization of conjunction ($\sqcap$)

1: A axiom; Variables: a_1, a_2, p_1, p_2; Functions: $getLHS(A)$, $getRHS(A)$, $getPOS(a)$
2: $a_1 \leftarrow getLHS(A)$ ▷ get entity to the left of $\sqcap$
3: $a_2 \leftarrow getRHS(A)$ ▷ get entity to the right of $\sqcap$
4: $p_1 \leftarrow getPOS(a_1)$ ▷ get the part of speech for a_1
5: $p_2 \leftarrow getPOS(a_2)$ ▷ get the part of speech for a_2
6: **if** $p_1 == noun\ and\ p_2 == noun$ **then**
7: **if** $a_2.startsWithVowel()$ **then**
8: Result $\leftarrow$ "a_1 n'a_2" ▷ Verbalize with *n'*
9: **else**
10: Result $\leftarrow$ "a_1 na a_2" ▷ Verbalize with *na*
11: **end if**
12: **else**
13: Result $\leftarrow$ "a_1 kandi a_2" ▷ Verbalize with *kandi*
14: **end if**
15: **return** Result

concept quantified over is required to form the full word for 'one'. The examples below show how $\exists$ can be verbalized (the prefix of the noun and the subject concord with *-mwe* are underlined):

Axiom1: Pill $\sqsubseteq \exists$ reduces.Pain

English: Every pill reduces **some** pain.

Runyankore: *Buri kajuma **hakiri** nikakyeendeza obusaasi.*

Axiom2: Medicine $\sqsubseteq \exists$ takenWith.Milk

English: Every medicine is taken with **some** milk.

Runyankore: *Buri mubazi hakiri neetwaarwa n'amate.*

Axiom3: Pill $\sqsubseteq \exists$ has.ActiveIngredient

English: Every pill has **at least one** active ingredient.

Runyankore: *Buri kajuma **hakiri** kaine ekirungo ky'amaani **kimwe**.*

In axioms 1 and 2, $\exists$ is verbalized as 'some' because the nouns quantified over, 'pain' and 'milk', are an abstract concept and a mass noun, respectively. Similarly, in Runyankore, only *hakiri* is used in the verbalization. Axiom 3, on the other hand, verbalizes $\exists$ as 'at least one', and shows how, for Runyankore, the relative concord *ki* (NC 7) of the concept quantified over is required to form *kimwe* 'one'. Further, in axioms 1 and 3, we show the verbalization of simple roles ('reduces' and 'has'), while axiom 3 is more than a binary relation and contains an indirect object. Finally, the verbalizations all show how the verb conjugation described in Section 4.4 fits into the verbalization process. For axioms 1 and 2, the standard CFG in Section 4.4 is used to conjugate *nikakyeendeza* 'reduces' and *neetwaarwa* 'taken'. The verb conjugation in Axiom 2 is in the passive (*w*) and applies phonological conditioning (*ee*). In Axiom 3, the CFG for the auxiliary in Section 4.4.1 is used to conjugate *kaine*. Algorithm 5.3.4 shows the steps taken to verbalize $\exists$, for both simple and complex roles.

There are three factors that affect the verbalization of $\exists$, the first being the NC of the concept's name, which is required to obtain the relative concord in order to form the full word for 'one' (lines 4 and 5 in Algorithm 5.3.4). The second is the quantifier, $\exists$, which is verbalized as '... *hakiri* ... *-mwe* for countable nouns, but as *hakiri* for mass nouns. Thirdly, the position of the concept

Algorithm 5.3.4 Verbalization of existential quantification ($\exists$)

A axiom; Variables: a, c, r, r', r_p, n; Functions: $getRole(A)$, $getRoleClass(r)$, $getNC(a)$, $getSC(n)$, $split(r)$, $handlePrepositions(r(i))$, $assignAsNoun(r(i))$
$r \leftarrow getRole(A)$ ▷ Get the role in the axiom
$a \leftarrow getRoleClass(r)$ ▷ Get the noun quantified over
$n \leftarrow getNC(a)$ ▷ Get the NC of the noun quantified over
$c \leftarrow getSC(n)$ ▷ Use the NC to obtain the subject concord
if $r.isComplexRole()$ **then**
    $r[] \leftarrow split(r)$ ▷ Split the role into its constituents
    $r' \leftarrow r(0)$ ▷ The first element is the verb
    for $i = 1; i < length(r); i++$ **do**
        if $r(i).isPreposition()$ **then**
            $r_p \leftarrow handlePrepositions(r(i))$ ▷ The handling of prepositions is described in Section 5.4
        else if $r(i).isConcept()$ **then**
            $a \leftarrow assignAsNoun(r(i))$ ▷ This term becomes the concept quantified over
            $n \leftarrow getNC(a)$ ▷ Get the NC of the noun quantified over
            $c \leftarrow getSC(n)$ ▷ Use the NC to obtain the subject concord
        end if
    end for
    if $a.isCountableNoun()$ **then**
        Result $\leftarrow$ "hakiri r' a cmwe" ▷ Verbalize with *hakiri* and *-mwe*
    else
        Result $\leftarrow$ "hakiri r' a" ▷ Verbalize only with *hakiri* for plural a
    end if
else
    if $a.isCountableNoun()$ **then**
        Result $\leftarrow$ "hakiri r a cmwe" ▷ Verbalize with *hakiri* and *-mwe*
    else
        Result $\leftarrow$ "hakiri r a" ▷ Verbalize only with *hakiri* for plural a
    end if
end if
return Result

determines whose NC will be used with *-mwe*. This is evident when a complex role contains a concept in its name. Consider the case where a complex role is composed of a verb, adjective, and noun (for example: $\exists$ hasActiveIngredient.Hydrocodone). It can be seen that the role contains a concept (Ingredient), whose position in the complex role makes it the concept quantified over, rather than 'Hydrocodone'. Therefore, the relative concord used with *-mwe* is that of 'Ingredient' (***eki**rungo* in NC 7), resulting in ***ki**mwe*, instead of ***e**mwe* (from 'Hydrocodone' in NC 9).

5.3.4 Universal Quantification of Roles ($\forall$)

While the verbalization of universal quantification in the context of subsumption is verbalized as 'all/each', the universal quantification of roles is verbalized as 'only'. In Runyankore, 'only' is *-onka* and it requires the possessive concord of the noun quantified over for the full verbalization. This is demonstrated in the examples below, with the possessive concord underlined:

Axiom1: Panado $\sqsubseteq \forall$ reduces.Pain

English: Every Panado **only** reduces pain.

Runyankore: *Buri Panado neekyendeeza obusaasi bw**onka**.*

Axiom2: Fennodil $\sqsubseteq \forall$ takenWith.Milk

English: Every Fennodil is **only** taken with milk.

Runyankore: *Buri Fennodil neetwarwa na amate g**onka**.*

The possessive concords of *obusaasi* 'pain' and *amate* 'milk', *bwa* and *ga* respectively, are required to verbalize $\forall$, resulting in *bwonka* and *gonka* respectively. This verbalization requires phonological conditioning, which is explained further in Section 5.5. In Algorithm 5.3.5, we show the steps required to verbalize $\forall$:

Algorithm 5.3.5 Verbalization of the universal quantification of roles ($\forall$)

1: A axiom; Variables: a_1, r, n_1, g_1; Functions: $getRole(A)$, $getNoun(r)$, $getNC(a_1)$, $getPossessiveConcord(n_1)$
2: $r \leftarrow getRole(A)$ ▷ Get the role in the axiom
3: **if** $r.isComplexRole()$ **then**
4: See lines 6 to 13 in Algorithm 5.3.4
5: **end if**
6: $a_1 \leftarrow getNoun(r)$ ▷ Get the noun quantified over
7: $n_1 \leftarrow getNC(a_1)$ ▷ Get the NC
8: $g_1 \leftarrow getPossessiveConcord(n_1)$ ▷ Use the NC to obtain the possessive concord
9: Result $\leftarrow$ "$r\ a_1\ g_1$onka" ▷ Verbalize with the appropriate possessive concord **return** Result

Similar to existential quantification, the verbalization of $\forall$ is also affected by three factors. The NC of the concept's name is required to obtain the correct possessive concord to use with *-onka* (lines 3 and 4 in Algorithm 5.3.5). The verbalization is also determined by the quantifier, where *-onka* verbalizes $\forall$. Again, similar to the verbalization of $\exists$, the position of the concept in the axiom determines which possessive concord should be used in the case of complex roles. If the role contains a concept in its name, then the concept quantified over becomes the concept within the role, thus affecting which possessive concord *-onka* is completed with.

5.3.5 Negation of Roles ($\neg$)

Negation of roles differs from negation of subsumption presented in Section 5.3.1 because it involves verb conjugation. Therefore, the verbal morphology described in Section 4.2 is referenced, with an emphasis on the pre-initial and formative grammatical slots. Negation of roles is represented by the use of the primary negative, *ti*, which occupies the pre-initial slot. Additionally, negation in this context is done in the participial present continuous tense, which places *riku* in the formative slot. If the role is complex, it is first split, as shown in line 7 of Algorithm 5.3.4, and then the verb is conjugated. In the examples below, we focus on the negation in the conjugated verb, ignoring the quantifiers already described in Sections 5.3.3 and 5.3.4.

Axiom1: Medicine $\sqsubseteq \neg \forall$ reduces.Headache

English: Every medicine **does not** reduce headache only.

Runyankore: *Buri mubazi **ti**gu**riku**kyeendeza obusaasi bw'omutwe bwonka.*

Axiom2: Pill $\sqsubseteq \neg \exists$ heals.Ulcer

English: Every pill **does not** heal some ulcers.

Runyankore: *Buri kajuma **ti**ka**riku**kiza ebironda by'omunda.*

Axiom3: Medication $\sqsubseteq \neg \exists$ hasActiveIngredient.Hydrocodone

English: Every medication **does not** have at least one active ingredient which is Hydrocodone.

Runyankore: *Buri mubazi **ti**gwine ekirungo ky'amaani kimwe, ekiri omubazi gwa Hydrocodone.*

The negation of roles is itself not affected by any of the six factors, as *ti* is always used. However, the verb conjugation is affected by the NC of the concept's name, which is required to obtain the correct subject concord for the initial slot; the position of the concept in the axiom, which affects which concept's NC will be used to obtain the subject concord; and the quantifier used. If the quantifier is $\forall$, then the same verbalization pattern presented in Algorithm 5.3.5 is used, with the only difference being that the verb is negated. If, on the other hand, the quantifier is $\exists$ as shown in axioms 2 and 3, then *hakiri* is dropped from the verbalization, as it introduces a positive sentiment. Algorithm 5.3.6 shows the steps taken during the negation of roles, which includes obtaining the subject concord for verb conjugation:

Algorithm 5.3.6 Verbalization of negation of roles ($\neg$)

A Axiom; Variables: a_1, r_1, c_1, n_c, s_c; Functions: $getRole(A)$, $getConstructor(A)$, $getNoun(A)$, $getNC(a_1)$, $getSubjectConcord(n_c)$
$r_1 \leftarrow getRole(A)$ ▷ Get the role in the axiom
$c_1 \leftarrow getConstructor(A)$ ▷ Get the constructor in the axiom
$a_1 \leftarrow getNoun(A)$ ▷ Get the noun in the axiom
$n_c \leftarrow getNC(a_1)$ ▷ Get the NC of the noun
$s_c \leftarrow getSubjectConcord(n_c)$ ▷ Get the subject concord
if $containsNegation(c_1)$ **then**
    Result $\leftarrow$ "ti$s_c$rikur$_1$" ▷ Conjugate the verb with the subject concord, *-riku-*, as the tense, and pre-initial as the negation *ti-*
else
    Result $\leftarrow$ "ni$s_c r_1$" ▷ Conjugate the verb with the subject concord, no tense morpheme, and pre-initial as the continuous marker *ni-*
end if
return Result

5.3.6 Maximum Cardinality ($\leq$)

Maximum cardinality ($\leq$) can be verbalized in English as either '... a maximum of ...' or '... not more than ...'. We use the latter for the Runyankore verbalization because it is the direct translation of $\leq$, and it is translated as *-tarikurenga*. In order to form the full word for 'not more than', the subject concord of the concept quantified over is required. This is shown in the examples below, where the subject concord is underlined:

Axiom1: Patient $\sqsubseteq \leq 2$ treatedBy.Doctor

English: Every patient is treated by **not more than** 2 doctors.

Runyankore: *Buri murwire naajaanjibwa abashaho <u>ba</u>**tarikurenga** 2.*

Axiom2: Pill $\sqsubseteq \leq 4$ has.SideEffect

English: Every pill has **not more than** 4 side effects.

Runyankore: *Buri kajuma kaine ebikurugamu bitagendereirwe <u>bi</u>**tarikurenga** 4.*

In the above examples, the subject concords of 'doctors' (*abashaho* in NC 2) and 'side effects' (*ebikurugamu bitagendereirwe* in NC 8) *ba* and *bi* respectively, are required to fully verbalize with *-tarikurenga*. Additionally, if the number restriction is greater than one, then the noun quantified over is pluralized. Algorithm 5.3.7 shows the steps taken to generate this pattern:

Algorithm 5.3.7 Verbalization of maximum cardinality ($\leq$)

1: A axiom; Variables: a_1, o, n_1, n_p, nc_p, and c_p; Functions: $getNumber(A)$, $getNext(A)$, $getNoun(o)$, $pluralize(n)$, $getNC(n_p)$, and $getSC(nc_p)$
2: $a_1 \leftarrow getNumber(A)$ ▷ Get the number restriction
3: $o \leftarrow getNext(A)$ ▷ Get the role after the number
4: $n_1 \leftarrow getNoun(A)$ ▷ Get the noun from the axiom
5: **if** $a_1 = 1$ **then**
6: $nc_1 \leftarrow getNC(n_1)$ ▷ Get the NC
7: $c_1 \leftarrow getSC(nc_1)$ ▷ Use the NC to obtain the subject concord
8: Result $\leftarrow$ "$c_1$tarikurenga a_1 n_1" ▷ Verbalize with the appropriate subject concord
9: **else**
10: $n_p \leftarrow pluralize(n_1)$ ▷ Pluralize the noun
11: $nc_p \leftarrow getNC(n_p)$ ▷ Get the plural NC
12: $c_p \leftarrow getSC(nc_p)$ ▷ Get the plural subject concord
13: Result $\leftarrow$ "$c_p$tarikurenga a_1 n_p" ▷ Verbalize with the plural noun and subject concord
14: **end if**
15: **return** Result

The verbalization of $\leq$ is affected by three factors: the NC of the concept's name, the position of the concept in the axiom, and the number restriction. The NC is required to obtain the correct subject concord (lines 7 and 12 in Algorithm 5.3.7). Similar to the verbalizations of $\exists$ and $\forall$, the presence of a concept in the name of a complex role changes the concept whose NC is used to obtain the subject concord. The number in the cardinality constraint determines whether pluralization is required or not (lines 4 and 8 in Algorithm 5.3.7).

5.3.7 Minimum Cardinality ($\geq$)

Minimum cardinality ($\geq$) can be verbalized in English as '... a minimum of ...', '... not less than ...', or '... at least ...'. We used the last ('... at least ...'), which is *hakiri*, in the Runyankore verbalization because this is a more directly translatable version. If the number restriction is '1', then the verbalization of $\geq$ is exactly the same as that of $\exists$ described in Section 5.3.3. Otherwise, the number itself is used in the verbalization, instead of *-mwe*, and the noun is pluralized. We demonstrate these two situations in the examples below:

Axiom1: Medication $\sqsubseteq \geq 3$ takes.Day

English: Every medication takes **at least 3** days.

Runyankore: *Buri mubazi* ***hakiri*** *nigutwara ebiro* ***3***.

Axiom2: Patient $\sqsubseteq \geq 1$ has.Symptom

English: Every patient has **at least one** symptom.

Runyankore: *Buri murweire* ***hakiri*** *aine omucucumo* ***gumwe***.

Axiom2 is verbalized exactly the same way as $\exists$, because the number restriction is '1'. In Axiom1, there is a need to pluralize the noun quantified over ('day') because the number restriction

Algorithm 5.3.8 Verbalization of minimum cardinality ($\geq$)

1: A axiom; Variables: a_1, o, n_1, n_p; Functions: $getNumber(A)$, $getNext(A)$, $getNoun(o)$, $pluralize(n)$
2: $a_1 \leftarrow getNumber(A)$ ▷ Get the number restriction
3: $o \leftarrow getNext(A)$ ▷ Get the role after the number
4: $n_1 \leftarrow getNoun(A)$ ▷ Get the noun from the axiom
5: **if** $a_1 = 1$ **then**
6: call Algorithm 5.3.4 ▷ Verbalize $\geq$ as 'at least one'
7: **else**
8: $n_p \leftarrow pluralize(n_1)$ ▷ Pluralize the noun
9: Result $\leftarrow$ "hakiri o n_p a_1" ▷ Verbalize with *hakiri* and the number
10: **end if**
11: **return** Result

is greater than '1'. Algorithm 5.3.8, only shows the steps taken in the latter situation, as the verbalization when the number restriction is '1' is already presented in Algorithm 5.3.4:

The verbalization of $\geq$ is affected by the NC of the concept's name (only in certain situations) and the number restriction. If the number is '1', then this is the exact same verbalization as $\exists$ and it is therefore handled by Algorithm 5.3.4. If, on the other hand, the number is greater than '1', then the NC is required in order to perform the pluralization (line 8).

5.3.8 Exact Cardinality (=)

The verbalization of exact cardinality (=) in English is '... exactly ...'. However, Runyankore does not have a direct translation for 'exactly[4]', but uses 'only' instead, which is *-onka*. Similar to the verbalization of $\forall$, the possessive concord is required to form the full word for 'only'. The verbalization of =, however, differs from that of $\forall$ in two ways: first, the number is included in the verbalization; and second, there is a need to pluralize the noun if the number is greater than '1'. This is depicted in the examples below:

Axiom1: Patient $\sqsubseteq = 2$ takes.Pill

English: Every patient takes **exactly** 2 pills.

Runyankore: *Buri murweire natwara obujuma 2 bw**onka**.*

Axiom2: Medication $\sqsubseteq = 1$ has.ActiveIngredient

English: Every medication has **exactly** 1 active ingredient.

Runyankore: *Buri mubazi gwine ekirungo ky'amaani 1 ky**onka**.*

The possessive concords, *bwa* for *obujuma* and *kya* for *ekirungo*, are used in the verbalization of =. Due to phonological conditioning (described later in Section 5.5), *bwa**a**onka* and *kyaonka* become *bwonka* and *kyonka* respectively. Axiom1 also shows when pluralization is required. In Algorithm 5.3.9, we algorithmically specify these steps:

The verbalization of = is affected by three factors: the NC of the concept's name, the position of the concept in the axiom, and the number restriction. The NC is required to obtain the correct

[4]It should be noted that a Runyankore dictionary [176] has a translation of 'exactly' as *buzima*. ***bu**zima* is the NC 14 abstract concept for ***ama**zima* 'truth', and can express the level of 'truthfulness'. So, in the following example, 'That is exactly what he said', 'exactly' here can be translated as *buzima*. However, when stating that, 'I can honestly say that he has exactly two children', then the translation is ***Buzima** aine abaana 2 **bonka***, with *buzima* affirming the truthfulness 'I can honestly say', and *bonka* translating 'exactly' in this context.

Algorithm 5.3.9 Verbalization of exact cardinality (=)

1: A axiom; Variables: a_1, o, n_1, n_p, nc_p, and g_p; Functions: $getNumber(A)$, $getNext(A)$, $getNoun(o)$, $pluralize(n)$, $getNC(n_p)$, $getPossessiveConcord(nc_p)$
2: $a_1 \leftarrow getNumber(A)$ ▷ Get the number restriction
3: $o \leftarrow getNext(A)$ ▷ Get the role after the number
4: $n_1 \leftarrow getNoun(A)$ ▷ Get the noun from the axiom
5: **if** $a_1 == 1$ **then**
6: $nc_1 \leftarrow getNC(n_1)$ ▷ Get the NC
7: $g_1 \leftarrow getPossessiveConcord(nc_1)$ ▷ Use the NC to obtain the possessive concord
8: Result $\leftarrow$ "n_1 a_1 $g_1$onka" ▷ Verbalize with the appropriate possessive concord
9: **else**
10: $n_p \leftarrow pluralize(n_1)$ ▷ Pluralize the noun
11: $nc_p \leftarrow getNC(n_p)$ ▷ Get the plural NC
12: $g_p \leftarrow getPossessiveConcord(nc_p)$ ▷ Get the plural possessive concord
13: Result $\leftarrow$ "n_p a_1 $g_p$onka" ▷ Verbalize with the plural noun and possessive concord
14: **end if**
15: **return** Result

possessive concord (lines 7 and 12 in Algorithm 5.3.9). The presence of a concept in the name of a complex role affects which NC will be used in the verbalization. Finally, the number determines whether pluralization is required or not, and if required, the NC is needed to pluralize the noun correctly.

5.4 Handling Prepositions

The presence of prepositions in roles (relations), such as 'taken**With**', and passives, such as 'operated**By**', changes the pattern in which the role is verbalized. Although the developed algorithms cannot handle every preposition, we cover those that appear in the test ontologies. These include: 'with' (*na*), 'in' (*omu*), 'of' (depends on the NC of the noun), and 'by' (*w*). The algorithms for 'of' and 'by' are limited to the situations where the former appears after a noun, and where the verb is in the past tense for the latter. Examples of roles containing these prepositions are:

With: Doctor $\sqsubseteq \exists$ works**With**.Nurse

Runyankore: *Buri mushaho hakiri naakora **na** omujaanjaabi omwe.*

In: Doctor $\sqsubseteq \forall$ found**In**.Hospital

Runyankore: *Buri mushaho naashaang**w**a **omu** eirwariro ryonka.*

Of: Heart $\sqsubseteq \exists$ part**Of**.Body

Runyankore: *Buri mutima hakiri guri ekicweka **ky'**omubiri gumwe.*

By: Patient $\sqsubseteq \forall$ treated**By**.Doctor

Runyankore: *Buri murweire naajaanj**ibw**a omushaho wenka.*

From these examples, we see that 'with' and 'in' are translated as *na* and *omu* respectively. If, however, the verb in the role is in the past tense (such as, 'found', above), the passive is introduced into the verbalization (as is the case with *naashaang**w**a*). We have already explained this use of the passive with the conjugated verb (as is the case for the verbalization of 'by') in Section 4.4, and it is therefore not repeated here.

The verbalization of 'of' is different, as the NC of the noun (NC 7 in the example above) is required to obtain the possessive concord (*ekya*), which then drops its initial vowel to form *kya*. Additionally, if the noun after the possessive concord starts with a vowel, then vowel assimilation occurs. In Algorithm 5.4.1, we show the steps to verbalize prepositions.

Algorithm 5.4.1 Verbalization of the prepositions 'with', 'in', 'by', and 'of'

1: A axiom; Variables: a_1, g_1, g_1', r, n_1, p_1, f_v; Functions: $getNoun(A)$, $getRole(A)$, $getNC(a_1)$, $splitRole(r)$, $getPossessiveConcord(n_1)$, and $dropIV(g_1)$
2: $a_1 \leftarrow getNoun(A)$ ▷ Get the noun in the axiom
3: $r \leftarrow getRole(A)$ ▷ Get the complex role in the axiom
4: $p_1 \leftarrow splitRole(r)$ ▷ Get the preposition by splitting the complex role
5: **if** $p_1.equals(\text{'with'})$ **then**
6: Result $\leftarrow$ "a_1 $r[0]$ na " ▷ Verbalize 'with' as *na*
7: **else if** $p_1.equals(\text{'in'})$ **then**
8: Result $\leftarrow$ "a_1 $r[0]$ omu " ▷ Verbalize 'in' as *omu*
9: **else if** $p_1.equals(\text{'by'})$ **then**
10: Result $\leftarrow$ "a_1 $r[0]$wf_v" ▷ Verbalize 'by' by conjugating the verb with *-w-* before the final vowel f_v
11: **else if** $p_1.equals(\text{'of'})$ **then**
12: $n_1 \leftarrow getNC(a_1)$ ▷ Get the NC
13: $g_1 \leftarrow getPossessiveConcord(n_1)$ ▷ Use the NC to obtain the possessive concord
14: $g_1' \leftarrow dropIV(g_1)$ ▷ Drop the initial vowel of the possessive concord
15: Result $\leftarrow$ "a_1 $r[0]$ g_1'" ▷ Verbalize with the noun and the possessive concord
16: **else**
17: Result $\leftarrow$ "a_1 r " ▷ Verbalize without accounting for prepositions
18: **end if**
19: **return** Result

Because these prepositions exist in complex roles, there is a preprocessing step to their verbalization involving splitting the complex role (see line 7 in Algorithm 5.3.4 and line 4 in Algorithm 5.4.1). Additionally, the verbalization of prepositions (even 'of', which requires the NC; see lines 5 and 6 in Algorithm 5.4.1) is not affected when pluralization is required.

5.5 Phonological Conditioning for the Nasal Compound *nk*

Recall that phonological conditioning is the change of a morpheme because of the phonological features of the morphemes surrounding it [142]. In Sections 3.5 and 4.5, we presented how phonological conditioning was achieved for nouns during pluralization, and for verbs during conjugation. We also explained that, while the phonological conditioning described in Section 4.5 is restricted to verbs, the rules described in Section 3.5 hold throughout Runyankore orthography, except where they contradict those for verbs. Here, we present a single aspect of phonological conditioning, whose need was identified during verbalization, specifically the verbalization of *-onka* 'only'.

Recall that in Sections 5.3.4 and 5.3.8, the possessive concord is required to correctly verbalize *-onka*; for example:

Axiom: Week $\sqsubseteq = 7$ has.Day

English: Every week has only 7 days.

Runyankore: *Buri saande eine ebiro 7* ***byonka***.

Byonka is composed of *ebya* and *-onka*. However, the possessive concord drops its initial vowel and the concatenation becomes *by**ao**nka*. While ***ao*** is usually resolved by the vowel elision of *a* followed by vowel coalescence of *o* (explained in Section 3.5), resulting in a double vowel *byoonka*, this is not so here. Due to the presence of the nasal compound *nk*, the double vowel is not required. Consequently, the phonological conditioning for *-onka* is achieved through the vowel elision of *a* only, resulting in *byonka*.

We added this rule to the phonological conditioning module described in Section 3.5, as it, too, is true throughout Runyankore orthography, whenever a nasal compound is present.

5.6 Evaluation of Verbalizations of $\mathcal{ALC}$ Constructors

When investigating verbalization, we began with most constructors in the DL $\mathcal{ALC}$, as a means of limiting the initial scope. These included: $\sqsubseteq$, $\sqcap$, $\neg$, $\exists$, and $\forall$. Once this was successful, we extended it to the DL $\mathcal{ALCQ}$ by including cardinality constructors. During the verbalization of $\sqsubseteq$, $\neg$, $\exists$, and $\forall$, we had several alternative verbalizations to consider. These were due to: (1) singular and plural alternatives; (2) the placement of the verbalization of the constructor in the sentence; and (3) the appropriate tense in which to conjugate the verb during verbalization. In order to decide which verbalization to implement, we carried out an evaluation similar to that of [69, 99], and asked study participants to identify which verbalization they preferred. The verbalization preferred by most of the study participants is the one whose pattern is in the algorithms in Section 5.3.

5.6.1 Materials and Methods

We used a questionnaire with five questions during this survey. These questions tested verbalizations and alternatives for the following constructors:

(1) Existential Quantification ($\exists$), which had four alternatives, *hakiri* after the noun, *hakiri* after the verb, no *-mwe*, or plural.

A. ***Buri*** *mwegyesa* ***hakiri*** *nayegyesa eishomo* ***rimwe***.
B. ***Buri*** *mwegyesa nayegyesa* ***hakiri*** *eishomo* ***rimwe***.
C. ***Buri*** *mwegyesa* ***hakiri*** *nayegyesa eishomo*.
D. *Abeegyesa boona nibeegyesa* ***hakiri*** *eishomo* ***rimwe***.

(2) Negation of Subsumption ($\sqsubseteq\ \neg$), which had four alternatives, singular, plural, or using either *ti* or *ti -ri*.

A. *Ekikopo* ***ti*** *kintu*
B. *Ekikopo* ***ti kiri*** *kintu*
C. *Ebikopo* ***ti*** *bintu*
D. *Ebikopo* ***ti biri*** *bintu*

(3) Subsumption ($\sqsubseteq$), which had two alternatives, singular or plural.

A. ***Buri*** *njangu n'enyamishwa*
B. *Enjangu* ***zoona*** *n'enyamishwa*

(4) Existential Quantification ($\exists$ with complex roles), which had six alternatives, singular, plural, the placement of *hakiri* before or after the verb, or whether to include *-mwe*.

A. ***Buri*** *musheija* ***hakiri*** *aine omwaana* ***omwe*** *ori omushaho*
B. ***Buri*** *musheija aine* ***hakiri*** *omwaana* ***omwe*** *ori omushaho*
c. ***Buri*** *musheija hakiri aine omwaana ori omushaho*
D. *Abasheija* ***boona hakiri*** *baine omwaana* ***omwe*** *ori omushaho*
E. *Abasheija* ***boona*** *baine* ***hakiri*** *omwaana* ***omwe*** *ori omushaho*

F. *Abasheija* ***boona*** *hakiri baine omwaana ori omushaho*

(5) Negation of Roles ($\neg\exists$), which had six alternatives, singular, plural, or whether to use the present tense, participial present tense, or the participial present continuous tense.

A. ***Buri*** *ntwiga* ***te****rya nyama*
B. ***Buri*** *ntwiga* ***teku****rya nyama*
C. ***Buri*** *ntwiga* ***teriku****rya nyama*
D. *Entwiga* ***zoona tiki****rya nyama*
E. *Entwiga* ***zoona tiziku****rya nyama*
F. *Entwiga* ***zoona tiziriku****rya nyama*

We obtained the study participants from Runyankore speakers in Kampala, Uganda; and used WhatsApp[5] to distribute the questionnaire because it is more familiar and widely used than email or online surveys. We recruited the participants using snowball sampling, where we started with a single family WhatsApp group, who then went on to send the questionnaire to other groups and individuals. We instructed the study participants to answer each of the five questions subjectively by selecting which verbalization they preferred, and encouraged them to explain the reasons for their choices. The answers were delivered to us via WhatsApp, each one consisting of the number of the question and the alternative selected (for example, 1C, 2A, 3A, etc.).

5.6.2 Results

We received 19 responses, but excluded one because it was incomplete. All of the 18 participants were middle-class Banyankore and spoke both English and Runyankore. For this reason, the results from these study participants cannot be generalized to the entire Runyankore speaking population that also includes first language Runyankore speakers. 78.8% were female and their age ranged from 24 to 59. Only three participants explained the reasons for their choices. Table 5.2 shows the number of participants who selected each alternative.

From the results shown in Table 5.2, the singular form was generally preferred by the majority of survey participants. For the first question, on the verbalization of $\exists$, 72.2% preferred the singular form with *hakiri* after the noun. The singular was also preferred in the second question, by 55.6%, for the negation of subsumption using *ti*. Further, for the third question, the singular was preferred for the verbalization of subsumption by 72.2% of survey participants. The plural form was only preferred in the fifth question, by 33.3% of participants, on the negation of roles.

The forth survey question, on the verbalization of complex roles, produced mixed results on individual alternatives: 27.8% of participants preferred the singular with *hakiri* after the noun, the singular with *hakiri* after the verb, and the plural with *hakiri* after the noun. We therefore further analyzed these results by classifying them into three groups: (1) singular versus plural; (2) *hakiri* after the noun versus *hakiri* after the verb; and (3) with *-mwe* versus without *-mwe*. Based on these categories, 72.2% selected the singular for group (1); and in group (2), 44.4% preferred *hakiri* after the noun that included *-mwe* for group (3).

Further, we evaluated for the appropriate tense to use during the negation of roles, to ascertain whether ordinary Runyankore speakers agreed with the linguists who recommended the use of the participial present continuous tense in Section 4.6. We found that both the use of the simple present tense and participial present continuous tense were preferred by 27.8% of participants. Based on the preferences by non-linguists and recommendations by linguists, we maintained the decision to use the participial present continuous tense for the negation of roles.

[5]WhatsApp is a mobile and desktop application that supports the sending and receiving of text, audio, video, documents, and location messages, as well as making voice and video calls. Further details are available at `https://www.whatsapp.com/about/`.

TABLE 5.2: Results from the evaluation on preferences of verbalization alternatives

Question	**Alternatives**	**Number of Preferences**
1 ($\exists$)	A. Singular with *hakiri* after the noun	13
	B. Singular with *hakiri* after the verb	1
	C. Singular with no *-mwe*	2
	D. Plural	2
2 ($\sqsubseteq$ $\neg$)	A. Singular with *ti*	10
	B. Singular with *ti...ri*	3
	C. Plural with *ti*	4
	D. Plural with *ti...ri*	1
3 ($\sqsubseteq$)	A. Singular	13
	B. Plural	5
4 (complex $\exists$)	A. Singular with *hakiri* after noun	5
	B. Singular with *hakiri* after verb	3
	C. Singular with no *-mwe*	3
	D. Plural with *hakiri* after noun	5
	E. Plural with *hakiri* after verb	0
	F. Plural with no *-mwe*	0
5 ($\neg$ of roles)	A. Singular with simple present tense	5
	B. Singular with participial present tense	0
	C. Singular with participial present continuous tense	1
	D. Plural with simple present tense	6
	E. Plural with participial present tense	1
	F. Plural with participial present continuous tense	5

5.7 Discussion

We began the work to verbalize ontologies in Runyankore by identifying which constructors to verbalize. Based on the kind of messages we aimed to generate in the healthcare domain, we selected eight constructors. They are: subsumption and its negation, conjunction, existential and universal quantification, negation of roles, and maximum, minimum, and exact cardinality.

Next, by verbalizing these constructors in Runyankore, and comparing with similar work in isiZulu [98, 99], we were able to identify that six factors affect verbalization. Five of these were identified for the verbalization of $\mathcal{ALC}$ in isiZulu [99], and these are:

(1) The noun class of the concept's name;

(2) The concept's category;

(3) Whether the concept is atomic or an expression;

(4) The quantifier used in the axiom; and

(5) The position of the concept in the axiom.

We found that these also affect verbalization in Runyankore, and additionally, for $\mathcal{ALCQ}$ with cardinality constructors, the number restriction was also a factor. These factors were the basis for the verbalization patterns shown in the algorithms.

Several tools exist for the verbalization of ontologies in English [85, 93], Greek [4], and synthetic languages like Latvian and Lithuanian [68, 69]. However, we cannot use these tools because they are template-based, and templates were found to be inapplicable to Bantu languages [4]. We instead based our work on that by Keet and Khumalo [98, 99], where patterns were used to undertake the verbalization, and accounted for the need for the NC. We were therefore able to bootstrap the verbalizations of five constructors (subsumption and its negation, conjunction, existential and universal quantification, and negation of roles) from the isiZulu patterns in [24]. This enabled us to extend our work to include complex roles and the cardinality constructors.

The evaluation presented in Section 5.6 showed that the singular form was preferred for all constructors except for the negation of roles. This preference for the singular verbalization for most constructors was also seen in the evaluation of controlled Baltic languages [69] and isiZulu [99]. Finally, we cannot conclude that the results from this evaluation will generalize to other dialects of Runyankore because there is disagreement among linguists about whether to regard Runyankore, Rutooro, Runyoro, and Rukiga as dialects of the language called Runyakitara. This issue is quite hotly contested.

5.8 Summary

In this chapter, we answered RQ2, which is about the verbalization patterns, by explaining how we selected the appropriate constructors, identified the factors affecting verbalization, showed the verbalization patterns for the selected constructors, and evaluated alternative verbalizations based on the preferences of study participants. We verbalized eight constructors in the description logic $\mathcal{ALCQ}$, and showed the verbalization patterns for subsumption ($\sqsubseteq$) and its negation, conjunction ($\sqcap$), existential quantification ($\exists$), universal quantification of roles ($\forall$), negation of roles ($\neg$), as well as the cardinality constructors: maximum ($\leq$), minimum ($\geq$), and exact ($=$). In each case, we used examples to depict the axiom input and the corresponding Runyankore output, and we used algorithms to show the steps taken during verbalization. Further, we presented the rules for phonological conditioning in the presence of a nasal compound. Finally, we evaluated the alternative verbalizations for $\sqsubseteq$, $\sqsubseteq\ \neg$, $\exists$ for simple and complex roles, and for the negation of roles. The results of this evaluation showed that: (1) the singular form was preferred for all constructors except the negation of roles; (2) the placement of *hakiri* after the noun was preferred to after the verb; (3) it was preferred for *-mwe* to be present rather than absent; and (4) the participial present continuous tense was preferred for the negation of roles.

Now that we have a pluralizer from Chapter 3, the CFGs from Chapter 4, and these verbalization patterns, the next chapter describes how we combined these three components in order to answer RQ3 from Section 1.2.1 in Chapter 1, about developing the grammar engine.

Chapter 6

Implementation of Runyankore Grammar Engine

6.1 Overview

In the previous chapter, we presented the verbalization algorithms for a subset of the DL $\mathcal{ALCQ}$, the final verbalization component, after noun pluralization presented in Chapter 3 and verb conjugation presented in Chapter 4. In this chapter, we answer RQ3, 'How can the Runyankore verbalization components be combined to form the grammar engine', from Section 1.2.1 in Chapter 1. We had three questions regarding implementation and evaluation:

RQ3 (a). Which annotation model should be used to associate concepts and roles with their corresponding linguistic information?

RQ3 (b). What form should the grammar engine take?

RQ3 (c). What is the appropriate evaluation strategy for the text generated in Runyankore?

In Section 6.2, we explain the decisions made in the design of the grammar engine, including the annotation model and type of application. In Section 6.3, we explain and show the architecture of the grammar engine; while Section 6.4 describes the implementation of the annotator, pluralizer, CFGs, and verbalization algorithms in Java as a Protégé plugin. Section 6.5 details the evaluation of the grammar engine by linguists and non-linguists. There are two main contributions here:

- an architecture for linguistic realization of an agglutinating Bantu language; and
- a Runyankore grammar engine for linguistic realization through ontology verbalization.

In this chapter, we extend the work published at the 10th International Conference on Natural Language Generation (INLG 2017) [26, 27] by showing the architecture of the grammar engine in Section 6.3, presenting a step-by-step example of the text generation process in Section 6.4, and providing details of the evaluation performed with the second linguist in Section 6.5.

6.2 Design Considerations

To recap, we explained in Section 2.4.2 that the verbalization process requires linguistic annotations to associate the ontological elements with linguistic information. We also identified in sections 3.2, 4.4, and 5.3 that linguistic information about each concept's and role's translation, part-of-speech, and NC is required for verbalization. From the various annotation models discussed in Section 2.4.2, we chose the XML-based annotation model by [97] because it enabled the annotation of an ontology with noun class information, and was successfully used in the verbalization of two Bantu languages, isiZulu and ChiShona.

We decided to implement the grammar engine as a Protégé5.X plugin. Protégé is a free, open-source ontology editor and framework for building intelligent systems[1]. It was developed by the Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine[2]. Protégé is widely used for creating, modifying, and reasoning on ontologies, and implementing the grammar engine as a plugin provides a seamless, quick, and easy means to verbalize existing or new ontologies in Runyankore. Since Protégé, the CFG tool [188], and the noun pluralizer are Java implementations, we decided to also implement the verbalization algorithms, phonological conditioning modules, and utility functions as a Java application. We selected a hash map as the data structure to represent the NC information in Table 2.2, with the NC number as the key, through which the corresponding subject concord, adjective concord, and possessive concord can be obtained.

6.3 Architecture of Grammar Engine

Figure 6.1 shows the architecture of the grammar engine, and how the different modules (Protégé's user interface, annotations, pluralizer, CFGs, verbalization algorithms, and phonological conditioning modules) interact to generate text.

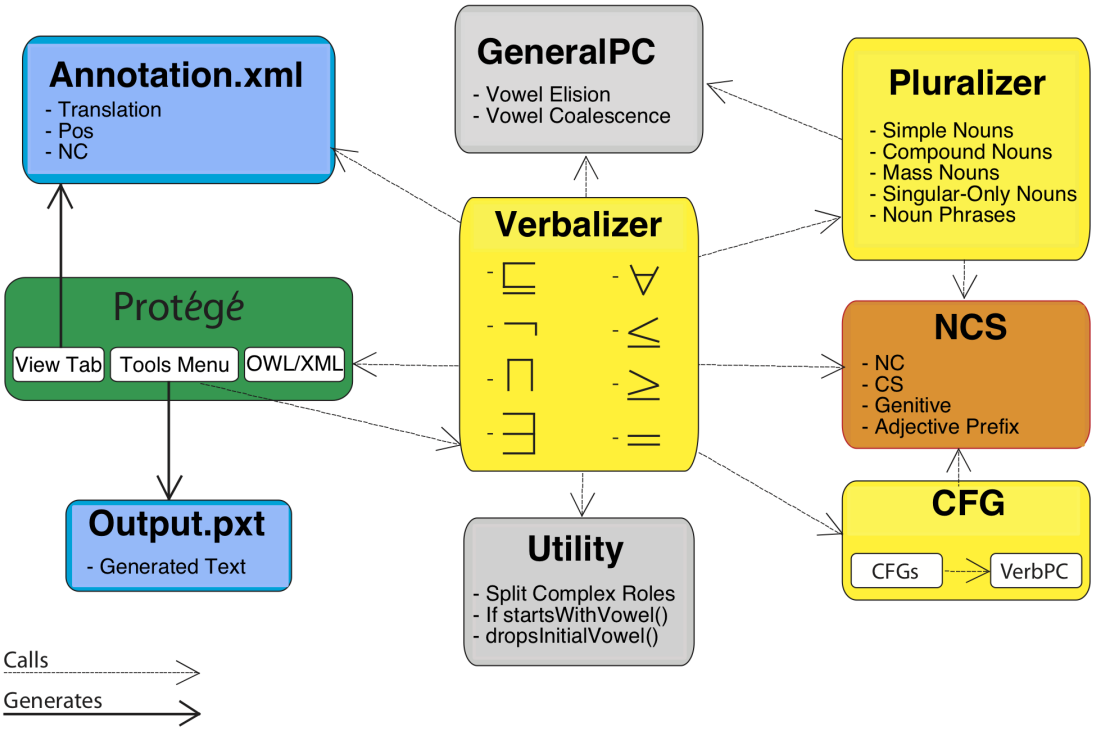

FIGURE 6.1: The architecture of the grammar engine, showing the interactions among Protégé's UI, noun class system (**NCS**), the pluralizer, CFG, verbalizer, phonological conditioning modules (**GeneralPC** and **VerbPC**), and utility functions (**Utility**)

The architecture presented in Figure 6.1 shows the main modules of the grammar engine. Protégé provides the user interface through which: (1) the grammar engine accesses the ontology; (2) the XML annotation file is created after annotating the concepts and roles through the 'View'

[1]For more details on obtaining a license and downloading Protégé, see https://protege.stanford.edu.

[2]For details about the aims and background to the development of Protégé, see https://protege.stanford.edu/about.php.

tab; and (3) the action to verbalize the ontology is performed through the 'Runyankore' submenu under the 'Tools' menu.

The main modules for text generation (verbalizer, pluralizer, and CFG) are shown, with the importance of the noun class system (NCS) highlighted. The **Verbalizer** obtains the NC, part-of-speech, and translation from **Annotations.xml**, and then uses the NC during pluralization and verb conjugation, which all require subject concords, prefixes and possessive concords from the **NCS**.

Utility and **GeneralPC** can be regarded as helper modules, with **Utility** performing preprocessing checks and actions, like checking if the name of a concept starts with a vowel, or splitting complex roles; and **GeneralPC** performing the phonological conditioning described in Sections 3.5 and 5.5 for pluralization and verbalization respectively. The phonological conditioning required for verbs, explained in Section 4.5, is internally handled by **VerbPC** during verb conjugation.

6.4 Implementation of Grammar Engine

As shown in Figure 6.1, Protégé provides the user interface for the plugin. We ensured that when an ontology is loaded, only the axioms that contain the eight constructors (described in Chapter 5) are selected for verbalization. The concepts and roles are then obtained from these axioms, and complex roles split into their constituents, which are then output to the annotate 'View' tab to be annotated with their corresponding translation, part-of-speech, and noun class (NC).

6.4.1 Annotate 'View' Tab

Protégé has a user interface element called a "View' tab', through which one can view and modify concepts, roles, and axioms. We added a 'View' tab with a panel through which information can be input in order to annotate the ontology.

We added several integrity checks during the entry of the annotations to ensure that the part-of-speech and NC information is input in the required format. The input of the part-of-speech is restricted to either 'A' for adjective, 'P' for preposition, 'N' for noun, or 'V' for verb, regardless of case. The NC is restricted to integers from 0 to 18, and 20 and 21, because there are 20 NCs in Runyankore, excluding '19' and '0' is allocated to the non-noun parts-of-speech (verbs, adjectives, and prepositions). Additionally, entries where the NC contains an 'a', 'd', or 'm' to indicate noun phrases with adjectives, definitions, and mass nouns respectively are accepted. Any entries that differ from these for the part-of-speech and NC result in an error dialog informing the user of the expected inputs. This integrity checking was implemented through the use of document filters.

Figure 6.2 shows a screenshot of Protégé's user interface, with the Runyankore annotate View tab selected. In the screenshot, a subset of the concepts and roles in the healthcare ontology SNOMED-CT [83] is shown, with their corresponding fields for translation, part-of-speech, and NC.

After annotating all the concepts and roles, clicking the 'Submit' button saves this information in an XML file. A sample of the structure of the XML annotation model is shown below. Notice the impact of splitting complex roles, which results in entities that do not contain the International-alized Resource Identifier (IRI) of the ontology (see 'Has' and 'active ingredient'), as compared to 'Opiate'.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ontology xmlns="&lt;http://www.ihtsdo.org/snomedct.owl&gt;">
    <annotation entity="&lt;http://www.ihtsdo.org/snomedct.owl#Opiate&gt;">
        <translation>omubazi ogukusinza ogukwejunisibwa kukyendeza obusaasi</translation>
        <partOfSpeech>n</partOfSpeech>
```

FIGURE 6.2: Protégé's UI, showing the 'View' Tab, used to annotate concepts and roles in the ontology

```
        <nounClass>3d</nounClass>
    </annotation>
    <annotation entity="Has">
        <translation>in</translation>
        <partOfSpeech>v</partOfSpeech>
        <nounClass>0</nounClass>
    </annotation>
    <annotation entity="active ingredient">
        <translation>ekirungo eky'amaani</translation>
        <partOfSpeech>n</partOfSpeech>
        <nounClass>7</nounClass>
    </annotation>
</ontology>
```

There is a separate XML annotation file for each ontology to be verbalized. As shown in Figure 6.1, during verbalization the annotation file is queried for information about the translation, part-of-speech, and NC of the concept or role of interest

6.4.2 Implementation of Verbalization Algorithms

We implemented the verbalization algorithms shown in Section 5.3 as methods in one Java class. Before verbalization can be attempted, the presence of the XML annotation file corresponding to the active ontology must be established. If it does not exist, a message dialog is used to alert the user to this. Further, once the verbalization process has started, partial or incomplete annotation of all concepts and roles of interest results in an exception, with the unannotated concept(s) and/or role(s) displayed in the error message. Adding the missing annotations results in the verbalization

process being completed. Here, we explain the step-by-step implementation of Algorithm 5.3.4 for $Hydrocodone \sqsubseteq \exists Has\_active\_ingredient.Hydrocodone$. This axiom has been directly taken from the healthcare ontology SNOMED-CT [83]..

First, the utility class splits the complex role 'Has_active_ingredient' into 'Has' and 'active ingredient'. These two, together with 'Hydrocodone' are annotated through the 'View' tab with their translations, parts-of-speech, and NCs, as shown in Figure 6.2, resulting in the creation of an XML file.

The NC of 'Hydrocodone' is then obtained from the XML file and used to check whether it is singular or plural, and because it is singular (NC 9), *Buri* 'each' is used to verbalize $\sqsubseteq$. Additionally, because 'active ingredient' is neither a mass noun nor belongs to a plural NC, $\exists$ is verbalized as *hakiri ... -mwe* (at least one). The NC of 'active ingredient' (NC 7) is used to obtain the relative concord *ki*, with which to form the full word for 'one', ***ki**mwe*, as shown in lines 4, 5, and 9 in Algorithm 5.3.4. The CFG for the auxiliary shown in Section 4.4 is applied in this case, because the verb is 'has'.

This results in the following output:

Axiom: $Hydrocodone \sqsubseteq \exists Has\_active\_ingredient.Hydrocodone$

English: **each** Hydrocodone has **at least one** active ingredient that is Hydrocodone

Runyankore: ***Buri*** *mubazi gwa Hydrocodone* ***hakiri*** *gwine ekirungo eky'amaani* ***kimwe*** *kiri omubazi gwa Hydrocodone*

omubazi gwa Hydrocodone is the translation of 'Hydrocodone'; *ekirungo eky'amaani* is the translation of 'active ingredient'; *gwine*[3] is the conjugated verb for 'has' with verb-root *in*; and *kiri* results from the conjugation of the copulative (is) with verb-root *ri*.

When the verbalization has been completed, a message dialog is used to alert the user and provide the name of the text file to which the text generated is saved.

6.4.3 Testing

We tested the grammar engine with four ontologies: (1) a subset of SNOMED-CT [83], a very large healthcare ontology; (2) university ontology[4]; (3) people ontology[5]; and (4) family ontology[6]. We used ontologies outside the healthcare domain in order to test the generalizability of the verbalization algorithms[7].

6.5 Evaluation of Generated Text

There are two fundamental methods of evaluating NLG systems: intrinsic and extrinsic [63]. We were interested in the intrinsic methods because they measure the performance of the NLG system independently of its setup, and usually test for correctness and readability of output [63]. There are three main evaluation strategies associated with these methods: (1) subjective human judgements, to assess the correctness of the generated text; (2) a human-based comparison of the generated text to a baseline, such as a state-of-the-art NLG system or human-authored text; and (3) an automatic metric, which assesses the generated text based on a standard metric [63, 159]. The automated metric evaluation requires high-quality reference text [63] which was not available for Runyankore, and it was therefore not used in our evaluation.

[3]It should be noted that *gwine* has undergone phonological conditioning, because the subject concord *gu* is conjugated with the verb-root *in*.

[4]http://www.mindswap.org/ontologies/debugging/university.owl

[5]http://owl.man.ac.uk/2005/07/sssw/people

[6]http://www.mindswap.org/ontologies/family.owl

[7]The implemented grammar engine can be accessed at `https://github.com/runyankorenlg/RunyankoreNLGSystem`.

Instead, we used subjective human judgements (referred from here on as 'the rate test') and the human-based comparison test. The rate test involves presenting the generated text to human subjects (domain experts, linguists, and/or target population) in order to obtain their opinions on the structure and correctness of the generated text [63, 159], as was done by DiMarco et al. [54], de Rosis, Grasso, and Berry [46], and Hussain et al. [81] in their healthcare-based NLG systems. The human-based comparison test involves comparing the generated text to the state-of-the-art text, which is usually human-authored text, based on the opinions of human subjects [63]; de Rosis, Grasso, and Berry [46] and Hussain et al. [81] also used this test.

For the rate test, we asked subjects to read and judge the generated text for grammatical correctness and understandability. For the human-based comparison to a baseline standard, we tested whether the generated text could be distinguished from human-authored text.

6.5.1 Procedure

We applied for and received ethics approval from the Faculty of Science Research Ethics Committee at the University of Cape Town, approval code FSREC 079-2016[8]. We used a questionnaire in this study, which was conducted with non-linguists from Mbarara, a district in Uganda where Runyankore is ethnically and predominantly spoken. We were assisted by a member of the Mirama community to make an announcement at the local Catholic church after the Sunday service. A request was also made to the headmaster of a nearby school, who agreed to let his students and staff take part in our study. All our study participants were at least 18 years old.

The study was undertaken on the school premises with the permission of the headmaster, and the 100 participants gathered in a single classroom. They were briefed about the purpose of the study, the consent form, and the different sections of the questionnaire. Any questions arising after the briefing were answered before the study commenced. After the consent forms were signed and collected, the participants proceeded to complete the Runyankore questionnaire, after which they were each compensated with UGX 5000 (about USD 1.31).

The linguists were contacted via email. We first contacted one from the Department of African Languages in the College of Humanities and Social Sciences at Makerere University in Uganda, and she then recommended two more, who were also requested to take part in the study. Of the three contacted, only two agreed to take part in the study, and they were briefed and given the English questionnaire via email. They in turn used email to return both the signed consent form and completed questionnaire. No compensation was given to the linguists.

6.5.2 Materials and Methods

We used a questionnaire with three main sections: (1) age, highest level of education, occupation, and first language; (2) ten generated sentences for the rate test; and ten sentences, with five generated and five human-authored, for the comparison test. We included sentences from the output of the grammar engine after verbalizing the four ontologies stated above, and we purposively selected sentences that could be used to evaluate important edge cases such as prepositions, complex roles, the passive, and noun phrases in the names of concepts.

For the non-linguists, we purposively chose participants who could read, write, and speak Runyankore. The consent form and questionnaire for non-linguists was translated to Runyankore, and this can be found in Appendix D.

On the other hand, the linguists were given an English version of the questionnaire[9]. We also replaced sentence I in the linguists' questionnaire with **Old_Lady** $\sqsubseteq \exists$ **reads.Publication** $\sqcap \forall$ **reads.Tabloid**, which was verbalized as: *Buri mukaikuru hakiri nashoma ekihandiiko ekishohoziibwe*

[8]The full ethics approval statement is available in Figure C.1 in Appendix C.

[9]The English version of the questionnaire given to linguists is available at `https://github.com/ThesisResources/EnglishQuestionnaire`.

TABLE 6.1: Sentences evaluated for grammatical correctness and understandability.

	DL Axiom	**Constructor Investigated**	**Runyankore Sentence**
A	Chlordiazepoxide_hydro chloride $\sqsubseteq \exists$ Has_active_ ingredient(Chlordiazepoxide)	$\exists$ in medical domain	*Buri mubazi gwa hydrochloride ya Chlordiazepoxide hakiri gwine ekirungo eky'amaani kimwe kiri omubazi gwa Chlordiazepoxide.*
B	Giraffe $\sqsubseteq \forall$ eats.Leaf	$\forall$	*Buri ntwiga nerya amapapa goonka.*
C	Leaf $\sqsubseteq \exists$ part_of.Leaf	$\exists$ with preposition	*Buri eipapa hakiri n'ekicweka kya omuti gumwe.*
D	Newspaper $\sqsubseteq$ Publication	$\sqsubseteq$	*Buri rupapura rw'amakuru n'ekihandiiko ekishohoziibwe.*
E	Person $\sqsubseteq = 2$ has.Parent	$=$	*Buri muntu aine abazaire 2 boonka.*
F	Student $\sqsubseteq \geq 1$ hasDegree	$\geq$	*Buri mwegi hakiri aine diguri emwe.*
G	ScienceProfessor $\sqsubseteq \forall$ adviserOf.ScienceStudent	$\forall$ with preposition	*Buri purofeesa wa sayansi n'omuhabuzi wa boonka abari abeegi ba sayansi.*
H	Man $\sqsubseteq \neg$ Woman	disjointness ($\sqsubseteq \neg$)	*Omukazi ti mushaija.*
I	Old_Lady $\sqsubseteq \exists$ has_pet.Animal $\sqcap \forall$ has_pet.Cat	$\sqcap$	*Buri mukaikuru hakiri aine enyamaishwa erikutungwa abantu kuzaanisa emwe eri enyamishwa, kandi aine enyamaishwa erikutungwa abantu kuzaanisa zoonka eziri enjangu.*
J	LecturerTaking4Courses $\sqsubseteq = 4$ takes.Course	$=$ with noun phrase	*Buri mushomesa orikushomesa amashomo ana natwara amashomo 4 goonka.*

kimwe, kandi naashoma taburoyidi zoonka. We did this to avoid having a definition in the verbalization.

We verbalized a small sample of SNOMED-CT [83] and this required the translation of medical jargon into Runyankore. Based on the translations provided by a linguist, we identified the following:

(1) For terms that are well known and commonly used, like 'Panado', the term is maintained;

(2) Other terms are given context in the translation; for example, 'Hydrocodone' is translated as *omubazi gwa hydrocodone* 'medicine of hydrocodone';

(3) Others still are both translated and contextualized; for example, 'diabetes' is translated as *endwara ya shukari* 'disease of sugar'; and

(4) Where the above three options are not applicable, the term is defined; for example, 'opiate' is translated as *omubazi ogukusinza ogukwejunisibwa kukyendeza obusaasi* 'medicine which intoxicates as treatment to reduce pain'.

Grammatical Correctness and/or understandability

The 10 sentences used for the rate test were required to be graded each according to four criteria: grammatically correct and understandable, incorrect grammar but understandable, grammatically correct but not understandable, and incorrect grammar and not understandable. Table 6.1 shows all the sentences in the questionnaire, as well as the DL axioms they verbalize, and the specific constructor whose verbalization we were testing for.

Sentence G originally had 'ProfessorInHCIorAI' and 'AIStudent' as concepts in the axiom. However, 'AI' and 'HCI' were replaced with 'Science' before the evaluation, because they were unfamiliar to the study participants, and this could have negatively affected how the sentence was graded.

TABLE 6.2: Sentences evaluated as either computer generated (C) or human-authored (H)

Label	DL Axiom	Runyankore Sentence
C1	Dog $\sqsubseteq \exists$ eats.Bone	*Buri mbwa hakiri nerya eigufa rimwe.*
C2	Hydrocodone $\sqsubseteq$ Morphine_derivative	*Buri mubazi gwa Hydrocodone n'omubazi gw'okukyendeeza obusaasi.*
H1	Lecturer $\sqsubseteq \neg$ Professor	*Omushomesa wa yunivasite ti purofeesa.*
C3	Sheep $\sqsubseteq \forall$ eats.Grass	*Buri ntaama nerya ebinyaansi byoonka.*
H2	Student $\sqsubseteq \leq 7$ studies.Course	*Buri mwegi naashoma amashomo gatarikurenga 7.*
C4	TeachingFaculty $\sqsubseteq \leq 3$ takes.Course	*Abashomesa omu kitongore boona nibatwara amashomo gatarikurenga 3.*
H3	Hydrocodone $\sqsubseteq$ Opiate	*Buri mubazi gwa Hydrocodone gurimu ebirungo ebirikukyendeeza obusaasi.*
H4	Parent $\sqsubseteq \exists$ hasChild	*Buri muzaire n'omuntu oine haakiri omwana omwe.*
C5	Cat $\sqsubseteq \neg$ Dog	*Enjangu ti mbwa.*
H5	Van $\sqsubseteq$ Vehicle	*Buri vaani n'emotoka.*

Computer Generated versus Human-Authored

10 sentences, with 5 authored by a Runyankore Linguist (H) and 5 computer generated (C), were presented to study participants. They were then required to grade each sentence either as human-authored or computer generated, based on its construction. The sentences used in this part of the questionnaire are presented in Table **??**, along with the DL axioms verbalized.

All study participants received a questionnaire containing the questions for evaluating grammatical correctness and understandability, and computer generated vs. human-authored.

6.5.3 Results and Analysis

Of the 100 non-linguists that took part in the study, 99% were aged between 18 and 30 years. Regarding their current occupation, 94% were students, 5% were teachers, and 1% were retired nurses. Additionally, 54% were female, 94% had high school as their highest level of education, and they all spoke Runyankore as their first language.

Grammatical Correctness and/or Understandability

For the evaluation on the grammatical correctness and understandability through the rate test, we aimed to obtain a grading of 'grammatically correct and understandable' by over 50% of the study participants. Figure 6.3 summarizes the results, where:

(1) The sentences that verbalized $\exists$ for the medical domain, $\sqsubseteq$, $=$, $\forall$ with preposition, and $\sqsubseteq \neg$–A, D, E, G, and H–were rated as 'grammatically correct and understandable' by 66%, 80%, 86%, 71%, and 92% of study participants, respectively.

(2) The sentences that verbalized $\forall$ and $=$ with noun phrase–B and J–received the next highest scores for 'grammatically correct and understandable' 47% and 38%, respectively.

(3) The sentences that verbalized $\exists$-with preposition, $\geq$, and $\sqcap$–C, F, and I–were rated as 'grammatically correct and understandable' by only 8%, 34%, and 26% of the study participants,, respectively.

Table 6.3 shows the results from calculating the statistical significance of the rating for each sentence using the multinomial exact test.

From Table 6.3, the results are statistically significant for Sentences A, B, D, E, G, H, and J being rated as both grammatically correct and understandable; Sentences C and F rated as having

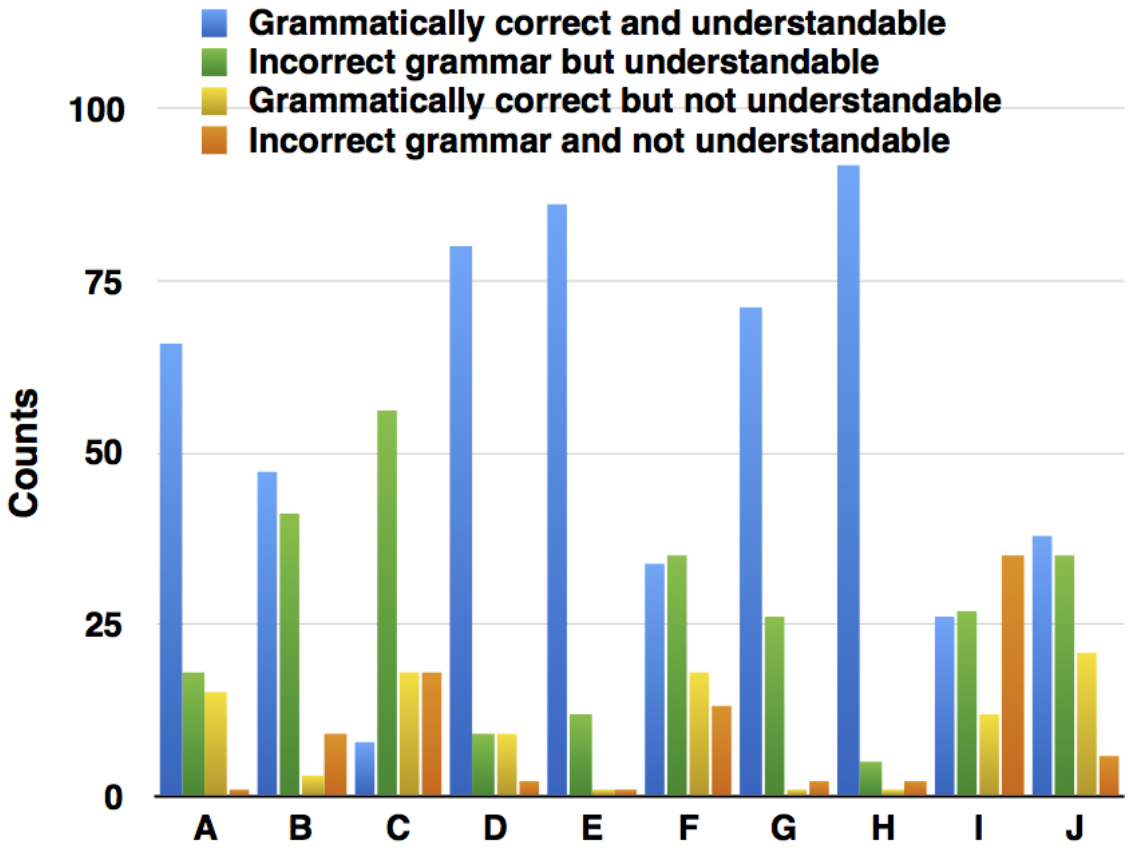

FIGURE 6.3: Evaluation results: A, D, E, G, and H performed very well (> 65%); C, F, and I performed poorly (< 35%); B and J performed marginally

incorrect grammar but understandable; and Sentence I as being both grammatically incorrect and not understandable.

We used the results of this evaluation to update the algorithms in the grammar engine in accordance with the reasons given for not grading a sentence as 'grammatically correct and understandable'. For example, sentence I, *Buri mukaikuru hakiri aine enyamaishwa erikutungwa abantu kuzaanisa emwe eri enyamishwa, kandi aine enyamaishwa erikutungwa abantu kuzaanisa zoonka eziri enjangu.*, had the worst outcome (mostly regarded as incorrect). This was because the noun phrase *enyamaishwa erikutungwa abantu kuzaanisa* (pet) was incorrectly pluralized when verbalizing $\forall$. We resolved this by developing the algorithm for the pluralization of noun phrases with verbs presented in Section 3.4.5, which pluralizes the noun phrase correctly as *enyamaishwa textbfzirikutungwa abantu kuzaanisa zoonka*.

Computer Generated Versus Human-Authored

For the comparison test, our desired outcome was to have all computer generated sentences graded as human-authored by over two-thirds of study participants. Hussain et al. [81] also used a human-based comparison test with 3 professionals, and their best result was that 64% of the overall text was regarded as human-authored. Figure 6.4 summarizes the results, where:

(1) All computer generated sentences (C) were regarded as human-authored.

(2) The generated sentences that verbalized $\forall$ and $\sqsubseteq$ $\neg$–C3 and C5–were regarded as human-authored by 90% and 97% of study participants respectively.

(3) The human-authored sentence that verbalized $\sqsubseteq$ (H5) was regarded as human-authored by only 56% of study participants.

TABLE 6.3: Results on the p-values of different ratings for each test

Sentence	P-value
A	2.38exp −17
B	9.8exp −15
C	1.91exp −07
D	1.79exp −27
E	7.76exp −37
F	0.001427
G	6.37exp −26
H	1.39exp −42
I	0.007307
J	1.09exp −03

Table 6.4 shows the results from using the binomial exact test to calculate the statistical significance of a computer generated sentence being regarded as human-authored.

TABLE 6.4: Results on the statistical significance of regarding a computer generated sentence as human-authored

Sentence	P-value
C1	1.59exp −05
C2	3.22exp −02
C3	$<$ 2.2exp −16
C4	0.006637
C5	$<$ 2.2exp −16

From Table 6.4, the results are statistically significant for Sentences C1, C2, C3, and C5 being regarded as human-authored; but not for C4. Using the binomial exact test on the results obtained for the human-authored sentences showed that the results were statistically significant for Sentences H1, H2, and H3; but not for H3 and H5.

The result that most study participants ($>$ 60%) regarded all generated text as having been written by a human being is a positive outcome, which is further supported by obtaining statistically significant results for four out of the five computer generated sentences.

Results from Linguists

Table 6.5 summarizes the results by linguists, showing how they graded each sentence.

TABLE 6.5: Results from the evaluation by linguists

Test	Grading	L1	L2
Rate Test	Grammatically correct and understandable	B, F, H, J	D, E, F, H
	Grammatically correct but not understandable	I	J
	Incorrect grammar but understandable	A, D, E, G	A, B, G, I
	Incorrect grammar and not understandable	C	C
Comparison Test	Computer generated	H1, C3, H4, C5, H5	C3, C4
	Human-authored	C2, H2, C4, H3	C1, C2, H1, H2, H3, H4, C5, H5

There were similarities and differences between the linguists' evaluations, as shown in Table 6.5. Only L1 provided explanations for the ratings given, and they include:

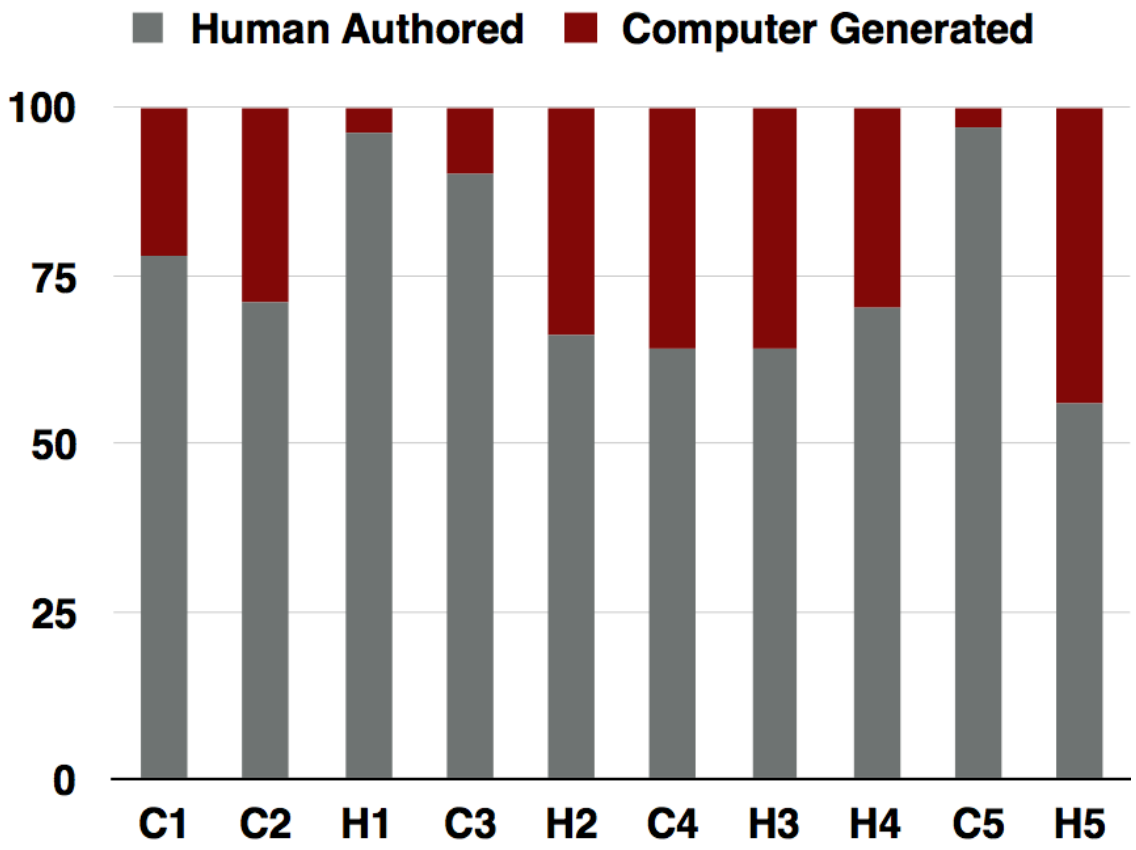

FIGURE 6.4: Computer vs Human: C1, C2, C3, C5 were considered human authored by more than 66%; H2, H3, H5 performed worse than C1, C2, C3, and C5

(1) Problems with the translation of a medical terminology in Sentence A, where the use of the possessive concord *ya* in *mubazi gwa hydrochloride ya Chlordiazepoxide* was discouraged because it 'sounds ungrammatical in foreign words adapted in Runyankore';

(2) The nature of the concepts in Sentence D, which makes it 'sound ambiguous';

(3) The lack of phonological conditioning for the nasal compound *nk* in *-oonka* in Sentences E and J;

(4) The incorrect placement of the prepositional phrase *wa boona abari abeegi ba sayansi* in Sentence G; and

(5) Problem with the conventional way of translating the loan word *taburoyidi* 'tabloid' in Sentence I.

Following this evaluation, we made updates to the algorithms in the grammar engine, for example, developing the algorithm for the phonological conditioning of the nasal compound presented in Section 5.5.

Comparison Between Results by Linguists and Non-linguists

There was some agreement between the evaluations done by linguists and non-linguists. For the rate test, all the sentences, except A and G, graded as 'grammatically correct and understandable' by the majority of non-linguists were graded the same by one of the linguists; while Sentences A and G, with statistically significant results from non-linguists, were graded differently as 'incorrect grammar but understandable' by both linguists. Sentence F, regarded as 'grammatically correct and understandable' by both linguists, was graded as 'incorrect grammar but understandable'

by a statistically significant number of non-linguists, along with sentence C, which was regarded as 'incorrect grammar and not understandable' by both linguists. We find it positive that for all study participants, very few sentences were regarded as having both incorrect grammar and not being understandable.

The human-based comparison test also produced varied results between linguists and non-linguists. Four of the generated sentences (C1, C2, C3, and C5) were regarded as human-authored by a statistically significant number of non-linguists. However, both C3 and C5 were graded as computer generated by L1, and C3 and C4 were regarded as such by L2. The reasons for these differences are unclear, as the participants did not explain their choices. On the whole, however, it is encouraging that most of the computer generated sentences were considered human-authored by linguists.

6.6 Discussion

Our implementation of a grammar engine that verbalizes ontologies in Runyankore is a step both in text generation and in producing computational resources for under-resourced languages. A popular grammar-based surface realizer is SimpleNLG [64], which is also implemented in Java. Though SimpleNLG does not generate text through ontology verbalization, its lexical component performs a similar function to the XML-based annotation model, by defining a lexicon and lexical items that correspond to major grammatical categories (noun, verb, adjective, etc.) [64]. Additionally, SimpleNLG's syntactic component, which handles different phrasal types defined by several grammatical features (such as tense, number, person, and mood) [64] performs similar functions to the verbalization algorithms and Context-Free Grammars in the Runyankore grammar engine.

The architecture presented in Section 6.3 shows that the grammatical complexity unique to agglutinating Bantu languages can be attributed to the noun class system. The architecture in Figure 6.1 shows how the noun class system **NCS** is integrated into all the key components of text generation: pluralization, verb conjugation, and verbalization. Given that other agglutinating Bantu languages are just as grammatically complex as Runyankore, the architecture for the Runyankore grammar engine might not be specific to Runyankore, but might be generalizable to the text generation needs of other agglutinating Bantu languages. If this is the case, then the Runyankore grammar engine can serve as a reference point into similar research for this class of languages.

6.6.1 Grammar Engine Implementation

We answered RQ3 (a), regarding how to associate each concept and role of interest in an ontology with its Runyankore translation, part-of-speech, and NC by using an XML annotation model. This decision, based on the ability of an XML annotation model to account for the NC, also has implications for the generalizability of this approach to other agglutinating Bantu languages, because a separate annotation file can be created and used during text generation for each language. Having the Runyankore translations as part of the annotations, as opposed to directly translating the concepts and roles in the ontology, has two advantages. Firstly, it separates linguistic information from the ontology, which is the recommended approach [36, 76, 112]; and secondly, it leaves room for generalization, as English can then be used as the base source language during translation.

The problem of providing NC-specific information during verbalization was solved using a hash map. While this solution worked perfectly for the Runyankore grammar engine, it limits the generalizability of this approach to other agglutinating Bantu languages, because they have different NC information. A possible solution would be to put the NC information in a text file, one for each language, which would then be loaded into the hash map once the desired language has been selected.

RQ3 (b), about what form the grammar engine should take was answered by implementing it as a Protégé plugin, and this makes it easy to use, as one is only required to create or load their

ontology, annotate the concepts and roles and, through the 'Runyankore' submenu under 'Tools', generate Runyankore text to a text file. However, it should be noted that the role of Protégé in the architecture shown in Figure 6.1 is for the input of the ontology and annotations). These two functions can be performed just as well by a stand-alone user interface or even the command-line. The choice of Protégé however provides a familiar location for verbalization in Runyankore among those who are already dealing with ontologies.

6.6.2 Evaluation of Grammar Engine

When answering RQ3 (c), we obtained encouraging results from the evaluation of the grammar engine by 100 non-linguists and two linguists using the rate test and human-based comparison test. For the rate test, we obtained statistically significant results from non-linguists for the best rating, 'grammatically correct and understandable', for seven (Sentences A, B, D, E, G, H, and J) of the ten sentences. Additionally, of the five computer generated sentences, four of them (Sentences C1, C2, C3, and C4) were regarded as human-authored by a statistically significant number of non-linguists. We used the' feedback from both linguists and non-linguists to improve the algorithms in the grammar engine, specifically regarding phonological conditioning, vowel assimilation, and the pluralization of noun phrases.

An interesting insight obtained during the evaluation is concerned with the four translation strategies for medical jargon explained in Section 6.5.2, which can be seen in the examples presented throughout this thesis. While these strategies make translating some medical jargon possible, there are still several terms for anatomy, diseases, and drugs which are not directly translatable to Runyankore. However, these strategies offer a starting point to handle such cases.

6.7 Summary

This chapter detailed how we answered RQ3 from Section 1.2.1, concerned with how to combine the different verbalization components into a grammar engine. We implemented the pluralizer, Context-Free Grammars (CFGs), and verbalization algorithms presented in Chapters 3, 4, and 5. We undertook the implementation in Java as a Protégé5.X plugin. We used an XML-based annotation model to associate concepts and roles with linguistic information about the translation, part-of-speech, and NC. The architecture of the grammar engine in Figure 6.1 shows how the various components interact to generate text. The generated text is accessible from the text file created after verbalization. Further, we explained and demonstrated the implementation details using an example. We used the rate test and human-based comparison test to evaluate the generated text, and the results showed that most of the generated text was rated as 'grammatically correct and understandable', and all generated text was regarded as human-authored by a majority of non-linguists.

These results answer phase one of our research explained in Section 1.3 in Chapter 1, regarding whether it is possible to verbalize ontologies in Runyankore. The next chapter presents phase two of our research, on the generalizability of the Runyankore approach to other agglutinating Bantu languages.

Chapter 7

Generalizability and Bootstrappability

7.1 Overview

In the previous chapter, we explained how we used a grammar engine as a surface realizer for Runyankore. Similar research on ontology verbalization in isiZulu [29, 96, 98, 99, 100] showed similar success with the same approach. Therefore, we investigated whether the same approach to verbalization is generalizable to other agglutinating Bantu languages.

When initially researching the verbalization patterns for Runyankore, we were able to reduce time and effort by bootstrapping from the existing isiZulu verbalization patterns in [98, 99]. Through the bootstrap approach, we were able to tailor Runyankore patterns for subsumption and its negation, conjunction, existential quantification, and universal quantification [24]; after which we extended the coverage to include complex roles, negation of roles, and number restrictions for Runyankore [26]. We therefore also investigated whether the bootstrap approach could be applied to identify verbalization patterns for other agglutinating Bantu languages.

Despite Guthrie's language classification not always remaining accurate or consistent as more linguistic knowledge is revealed about individual languages, Maho [123] argues that there are fewer problems associated with establishing a referential classification than a genetic one, and has updated the Guthrie zones to reflect new linguistic findings. We therefore found the ability to bootstrap from isiZulu to Runyankore unexpected because they belong to different Guthrie zones (S.40 and J.10 [123] respectively), and further investigated whether bootstrapping across zones is unique to Runyankore and isiZulu, or possible between other languages. We also investigated whether it is more efficient to bootstrap between languages in the same zone than across zones.

Throughout this thesis, we have used the noun classification according to Meinhof (presented in Section 2.3.1). However, for both Runyankore and isiZulu, we identified several limitations in this noun classification when applied to computational tasks, which indicate a possible barrier to generalizability. These include:

(1) The presence of multiple class prefixes in a single NC (for example NC 5 in Runyankore with prefixes *ei* and *eri*), and the same class prefix for different NCs (for example NCs 1 and 3 in Runyankore and isiZulu [29]); this can cause non-determinism in noun pluralization algorithms (as explained in [29] and Chapter 3);

(2) Some of the rules specified by Meinhof's NC tables, especially for lesser known cases, make generalizing computational tasks to multiple languages difficult; for example, the need for phonological conditioning when pluralizing nouns in NC 1 which have a stem commencing with a vowel other than *u* in isiZulu [29] and Runyankore;

(3) In Chapter 3, we defined 'exceptions' as nouns whose pluralization algorithm requires more or different rules than the basic ones in the NC singular/plural pairings, and their rules are specific to a language, thus limiting generalizability; and

(4) The same concepts are classified differently in different languages, and this too limits the ability to develop language-independent computational tools; for example, Runyankore's noun

classes do not have a separate class for kinship like isiZulu's, yet kinship terms are pluralized differently from other 'people' nouns;

When answering RQ4, 'How can the Runyankore and isiZulu approaches to ontology verbalization be generalized to other agglutinating Bantu languages' from Section 1.2.1 in Chapter 1, we had the following questions:

RQ4 (a.1). When applied to computational tasks, are the limitations of Meinhof's noun classification that are observed in Runyankore and isiZulu present in other agglutinating Bantu languages; and if so, how can it be revised systematically to foster generalizability?

RQ4 (a.2). Given that Guthrie's classification, despite its limitations, is regarded as relevant by some linguists, is it more efficient to bootstrap between languages in the same zone than across zones?

RQ4 (a.3). When there are multiple source languages from which to bootstrap, how can the most efficient source–target language pair be identified?

In the context of this research, 'bootstrappability' refers to how efficiently a target language can be bootstrapped from one or more source languages. In Section 7.2, we present the methodology we undertook during this investigation. Section 7.3 explains how the five languages investigated were selected, as well as their grammatical features. The regrouping of the classes of Meinhof's noun classification is presented in Section 7.4. In Section 7.5, we present the analyses undertaken to investigate bootstrappability, and we explain the resulting metric developed to measure bootstrappability in Section 7.6. The major contributions in this chapter are:

- The regrouping of the class prefixes of Meinhof's noun classification in Section 7.4; and
- A metric for measuring the efficiency of bootstrappability in Section 7.5.

In this chapter, we extend the work published at the 27th International Conference on Computational Linguistics (COLING 2018) [28] with results in Section 7.4 to show that the class regrouping can extend to other Bantu languages, beyond the seven used in this investigation. We also present the analyses performed when investigating bootstrappability in Section 7.5 and its efficiency in Section 7.6.

7.2 Methodology

The development of computational resources is an intensive activity for any language [21], and this is even more so for very under-resourced languages like the Bantu languages. The development of language resources in new languages can be fostered by transferring the existing knowledge and resources from a resource-rich language to a language with fewer resources [82]. This application of existing resources to solve a similar problem in another language is referred to as bootstrapping.

The bootstrap approach has successfully been used to develop text generation resources using different methods. Jarrar, Keet, and Dongilli [86] used templates to develop multilingual verbalizations of logical theories. The initial verbalization template file, in Dutch, was tailored to a grammatically related language, in this case, German, by varying the text in the text tags and their position, to reflect the language structure of the target language [86]. The structure of the abstract syntax of the Resource Grammar Library (RGL) used to generate text in the Grammatical Framework (GF), discussed in Chapter 2, was originally designed for European languages, but support for languages in other language families (Thai, Japanese, and Chinese) has been achieved by bootstrapping from the initial abstract syntax [155]. GF currently supports over 30 languages [154, 155]. SimpleNLG, a realization engine initially developed in English [64], has since been ported to French [180], German [18], Italian [1], Mandarin [34], Brazilian Portuguese [42], and Spanish [152].

[1]The Italian implementation of SimpleNLG is available from https://github.com/alexmazzei/SimpleLEX-IT

This was achieved by using language-specific components like the lexicon, while tailoring the syntactic component to each language's grammar [18, 34, 42, 152, 180]. JSrealB, an English and French surface realizer, bootstraps from the French version of SimpleNLG [180] and JSreal (a web surface realizer written in JavaScript) [39] to produce a bilingual text realizer for French and English [137]. JSrealB uses the lexicon structure and rules from JSreal, while SimpleNLG provides the syntactic hierarchical tree representation to generate the text [137].

For Bantu languages, Bosch, Pretorius, and Fleisch [21] applied an experimental bootstrapping approach to develop morphological analyzers for isiXhosa, seSwati, and isiNdebele based on the existing one for isiZulu. Starting with the isiZulu morphological analyzer, they made the following language-specific modifications for each language: word roots lexicon, grammatical morpheme lexicon, as well as the language appropriate morphophonological rules [21]. The bootstrap approach was also used to obtain Runyankore verbalization patterns, based on the isiZulu ones [24]. The same factors affecting verbalization of five constructors in isiZulu were found to also hold for Runyankore (see Section 5.7), and the Runyankore verbalization patterns were then tailored from the isiZulu ones [24].

The benefits of using the bootstrap approach to develop language resources include: (1) reduction in development time and effort, for example, in [6], English initially required four days, while the others each required a matter of hours; and (2) maintaining a high level of accuracy, for example, the increase from an average of 71.3% to 95.6% in [21].

When investigating the use of the bootstrap approach for other agglutinating Bantu languages, we used the following questions:

(1) Do the same factors that affect verbalization in Runyankore and isiZulu also hold for other agglutinating Bantu languages?

(2) Given multiple source and target languages, how can the most efficient source–language bootstrap pair be identified?

Here, we investigate the verbalization of most constructors in the description logic $\mathcal{ALC}$, on which bootstrapping from isiZulu to Runyankore [24] was performed. We differentiate between bootstrapping among languages in the same zone (intra-zone bootstrappability) and bootstrapping among languages across different zones (inter-zone bootstrappability). For the former, we selected two languages, Luganda, from the same zone as Runyankore (J.10), and isiXhosa, from the same zone as isiZulu (S.40). For inter-zone bootstrappability, we only considered agglutinating Bantu languages, with the same morphology as Runyankore and isiZulu. We selected three languages from three different Guthrie zones: Kinyarwanda (J.60), Kikuyu (E.50), and chiShona (S.10). Figure 7.1 shows the location of the selected languages across Africa.

We selected these five languages because they are actively used in their countries of origin–isiXhosa in South Africa, Luganda in Uganda, Kinyarwanda in Rwanda, Kikuyu in Kenya, and chiShona in Zimbabwe. Additionally, their linguistics are taught at university level, which provides access to current documentation.

In Sections 2.3.1, 2.3.2, and 2.3.3, we explained that the grammatical complexity of Bantu languages is due to noun classification, agglutination, and their verbal morphology. We thus researched the grammatical structures of the selected five languages.

7.3 Grammatical Features of Selected Languages

There are several aspects of the grammars of chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda which we were interested in: (1) the Guthrie zones [123] to which they belonged, in order to ensure that we could investigate both intra-zone and inter-zone bootstrapping; (2) their noun class (NC) systems, which, as shown in Figure 6.1 in Chapter 6, are crucial to the text generation process; (3) their verbal morphologies, which are generally complex for Bantu languages as

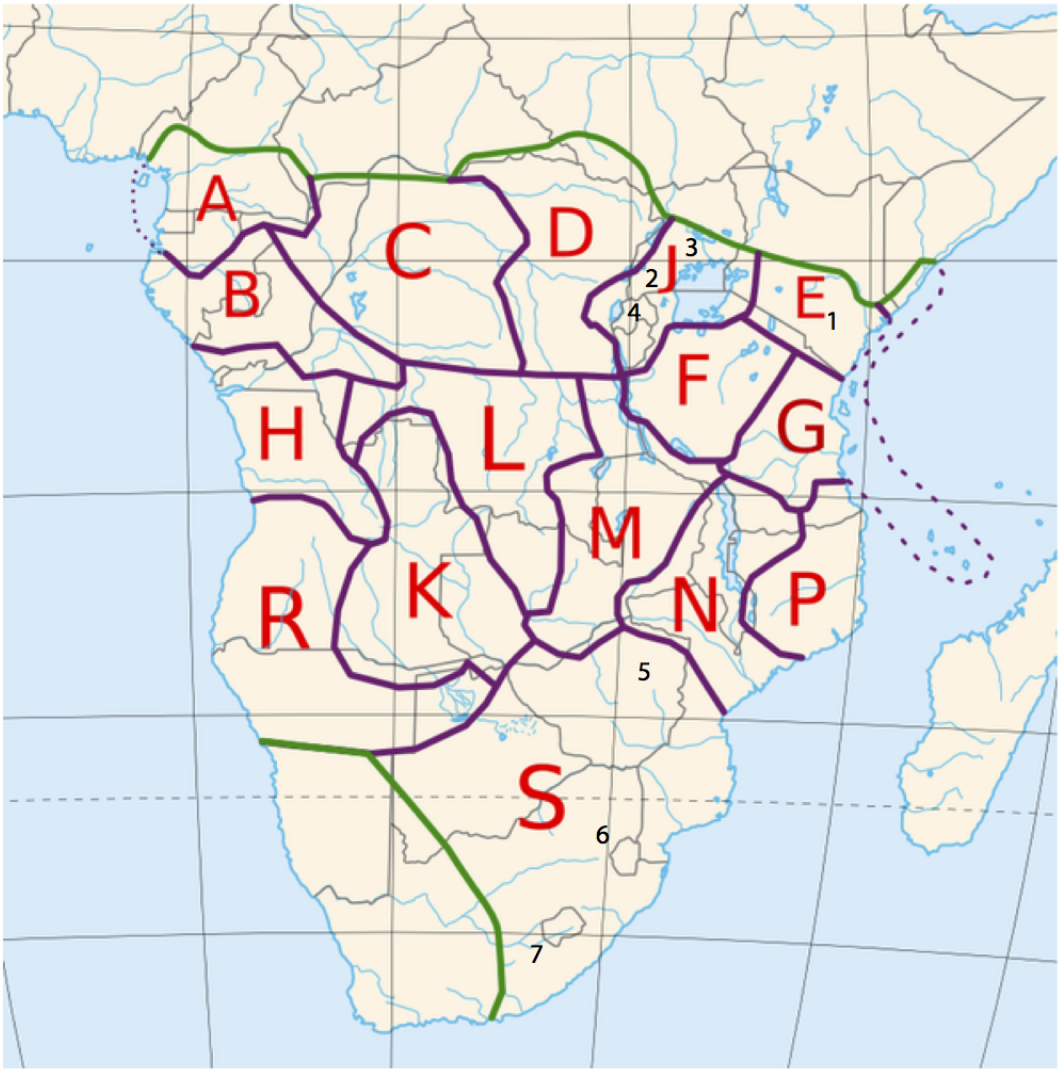

FIGURE 7.1: The Guthrie zones of the selected languages: (1) Kikuyu (E.50), (2) Runyankore and (3) Luganda (J.10), (4) Kinyarwanda (J.60), (5) chiShona (S.10), (6) isiZulu and (7) isiXhosa (S.40)

explained in Sections 2.3.3 and 2.3.3; and (4) the need for phonological conditioning, which we have shown to be necessary during noun pluralization, verb conjugation, and verbalization. The summary of this information is presented in Table 7.1 [11, 47, 60, 88, 89, 101, 128, 140, 174, 190].

Similar to Runyankore and isiZulu, these five languages also possess noun classes with the same class prefix but belonging to different NCs [11, 47, 88, 101, 128, 174, 190], which are highlighted in Table 7.2.

Katamba [94] explained that some NCs have gone into disuse in some Bantu languages, which can be seen in the number of NCs in Table 7.1 and the marking with 'N/A' in Table 7.2. Further, as explained in Section 2.3 and in Chapters 3, 4, and 5, the NC determines the agreement markers on the associated lexical categories, which is depicted with the NC-dependent subject concord ('SC') in Table 7.1[2].

The details of the grammatical features of these five languages revealed the following features common among them and Runyankore and isiZulu:

[2]The details on other lexical categories determined by the NC for chiShona, isiXhosa, isiZulu, Kikuyu, Kinyarwanda, and Luganda can be seen in tables F.1, F.2, F.3, F.4, and F.5 in Appendix F.

TABLE 7.1: Relevant features possessed by the selected languages ('FV', Final Vowel; 'Neg', Negation; 'NegSC', Negative Subject Concord; 'SC', Subject Concord; 'TnA', Tense and Aspect; 'VC', Vowel Coalescence; 'VE', Vowel Elision; 'VH', Vowel Harmony; and 'VR', Verb-Root)

Language	**Guthrie Zone**	**Number of NCs**	**Verbal Morphology Simple Present | Negation**	**Phonological Conditioning**
chiShona	S.40	20	SC-TnA-VR-FV | Neg-SC-VR-FV	Yes, with VC, VE, and VH
isiXhosa	S.40	15	SC-VR-FV | Neg-SC/NegSC-VR-FV	Yes, with VC and VE
Kikuyu	E.50	17	SC-VR-TnA-FV | Neg-SC-VR-TnA-FV or SC-Neg-VR-TnA-FV	Yes
Kinyarwanda	J.60	16	SC-VR-FV | Neg-SC-VR-FV	Yes, with VC and VE
Luganda	J.10	21	SC-TnA-VR-FV | Neg-SC-TnA-VR-FV	Yes, with VC and VE

(1) They require phonological conditioning [47, 60, 88, 101, 128].

(2) Their phrasal structure is 'subject, verb, object', and the adjective is always placed after the noun [47, 60, 88, 101, 128].

(3) As shown in Table 2.1 in Section 2.3.1, those languages with NCs 12 and 13 (chiShona, Kikuyu, Kinyarwanda, Luganda, and Runyankore) all place diminutive objects in these classes [11, 88, 101, 190].

(4) Also shown in Table 2.1 and found here is that NC 15 contains infinitives, abstract nouns are found in NC 14, and NCs 16, 17, and 18 are locative classes [11, 47, 88, 101].

On the other hand, tables 7.1 and 7.2 show several differences among these languages, which prevent the direct reuse of the Runyankore and isiZulu algorithms:

(1) Each language has its own lexicon; and they have varying numbers of NCs, sometimes with different NC prefixes and associated affixes.

(2) The simple verbal morphology presented in Table 7.1 shows that the placement of morphemes for the simple present tense and negation varies for each language; and this is made even more apparent when the full range of tenses, aspects, and extensions presented in Section 2.3.3 is considered for each language [60, 88, 101, 128, 140].

(3) Finally, while some rules for handling phonological conditioning, such as **u + vowel = w + vowel**, are found in some languages like isiZulu, Kinyarwanda, Luganda, and Runyankore [47, 60, 88, 101, 140], it is not the case for other languages like Kikuyu and chiShona. Additionally, there are language-specific rules, for example; Runyankore uses vowel coalescence and elision next to a nasal compound, while Luganda only uses vowel elision.

The grammatical similarities among these languages offer areas around which to generalize algorithms. However, the differences between them indicate a need for language-specific algorithms. One of the main limitations to generalizability is the complexity of computationally applying Meinhof's noun classification system across multiple languages.

7.3.1 Computational Limitations to Generalizing Meinhof's Noun Classification

For noun pluralization algorithms, the structure of Meinhof's noun classification does not provide a one-to-one singular/plural mapping. Maho [122] admits that only considering the class prefixes to obtain the correct plural can lead to mistakes owing to the presence of classes with the

TABLE 7.2: Standard NC classification by singular/plural pair (the first line is singular, and its plural is the second or third line), highlighting the same prefixes across more than one NC for a language. The dashes between the letters in the prefix represent separation between the initial vowel (augment) and prefix; '-': empty prefix; 'N/A': the NC is not present in that language (none use NC19 or NC22).

NC	chiShona	isiXhosa	isiZulu	Kikuyu	Kinyarwanda	Luganda	Runyankore
1	**mu-**	**u-m-**	**u-m-/u-mu-**	**mu-**	**u-mu-**	**o-mu-**	**o-mu-**
2	va-	a-ba-	a-ba-/a-b-	a-	a-ba-	a-ba-	a-ba-
1a	**-**	**u-**	**u-**	**-**	N/A	N/A	N/A
2a	vana-	oo-	o-	**-**	N/A	N/A	N/A
2b	a-	N/A	N/A	N/A	N/A	N/A	N/A
3a	N/A	N/A	**u-**	N/A	N/A	N/A	N/A
2a	N/A	N/A	o-	N/A	N/A	N/A	N/A
3	**mu-**	**u-m-**	**u-m-/u-mu-**	**mu-**	**u-mu-**	**o-mu-**	**o-mu-**
4	mi-	i-mi-	i-mi-	mi-	**i-mi-**	**e-mi-**	**e-mi-**
5	**-**	**i-**/i-li-	**i-**/i-li-	ri-	**i-**/i-ri-	**e-**/e-li-	e-i-/e-ri-
6	ma-	a-ma-	a-ma-	ma-	a-ma-	a-ma-	a-ma-
7	chi-	i-si-	i-si-	ki-/gi-	i-ki-/i-cy-/i-gi-	e-ki-	e-ki-
8	zvi-	i-zi-	i-zi-	ci-/i-	i-bi-	e-bi-	e-bi-
9	**n-**	**i-**/i-n-	**i-**/i-n-	**n-**	**i-/i-n-/i-nz-**	**e-n-/e-m-**	**e-n-/e-m-**
10	**n-**/dzi-	ii-/i-zin-	i-zi-/i-zin-	**n-**	**i-/i-n-/i-nz-**	**e-n-/e-m-**	**e-n-/e-m-**
9a	N/A	N/A	**i-**	**-**	N/A	N/A	N/A
10a	N/A	N/A	N/A	**-**	N/A	N/A	N/A
6	N/A	N/A	a-ma-	N/A	N/A	N/A	N/A
11	ru-	**u-**/u-lu-	**u-**/u-lu-	ru-	u-ru-	o-lu-	o-ru-
10	**n-**/dzi-	ii-/i-zin-	i-zi-/i-zin-	**n-**	**i-/i-n-/i-nz-**	**e-n-/e-m-**	**e-n-/e-m-**
12	ka-	N/A	N/A	ka-/ga-	a-ka-/a-ga-	a-ka-	a-ka-
13	tu-	N/A	N/A	tu-	u-tu-	o-tu-	o-tu-
14	u-	u-bu-	u-bu-	**-**	u-bu-	o-bu-	o-bu-
6	ma-	N/A	N/A	N/A	N/A	N/A	N/A
15	**ku-**	u-ku-	u-ku-	**ku-/gu-**	u-ku-/u-du-	**o-ku-**	**o-ku-**
6	N/A	N/A	N/A	ma-	a-ma-	a-ma-	a-ma-
16	ha-	N/A	N/A	ha-	a-ha-	a-ha-	a-ha-
17	**ku-**	N/A	ku-	**ku-/gu-**	N/A	**o-ku-**	**o-ku-**
18	**mu-**	N/A	N/A	N/A	N/A	**o-mu-**	**o-mu-**
20	N/A	N/A	N/A	N/A	N/A	o-gu-	o-gu-
21	N/A	N/A	N/A	N/A	N/A	a-ga-	a-ga-
21	zi-	N/A	N/A	N/A	N/A	N/A	N/A
6	ma-	N/A	N/A	N/A	N/A	N/A	N/A
23	N/A	N/A	N/A	N/A	N/A	**e-**	N/A

same prefix, such as NCs 1 and 3 in Table 7.2. Another source of non-determinism for pluralization algorithms is the presence of multiple class prefixes for a single class (for example, NC 5 in Table 7.2 for all languages except chiShona). This was handled in Algorithm 3.3.1 in Chapter 3 using if-statements with language-specific class prefixes, which impedes generalizability to other languages.

Further, the different numbers of noun classes in each language (see Table 7.1) with different class prefixes (see Table 7.2) also make the direct reuse of algorithms such as Algorithm 3.3.1 in Chapter 3 by another language impossible. This is because the noun class (NC) number and class prefix are included in the algorithm. This limitation further affects the pluralization of compound nouns and noun phrases which require concords obtained through the NC.

Finally, Maho [122] makes a distinction between additive and substitutive pluralization. Additive pluralization occurs when the plural prefix attaches to the entire noun without dropping the singular prefix, while substitutive pluralization occurs when the plural prefix replaces the singular prefix [122]. Computationally specifying the rules for individual cases for each language also renders the reuse of the Runyankore and isiZulu algorithms impractical, especially for unseen cases.

These limitations to the reuse of the existing Runyankore and isiZulu algorithms led us to investigate whether Meinhof's NC system can be regrouped with the aim of implementing generic algorithms.

7.4 Regrouping of Meinhof's Noun Class System

There are several ways in which the regrouping of Meinhof's NC system can be done: (1) morphologically (and syntactically for compound nouns), where each different class prefix is regrouped as a subdivision of the general class, with the subdivision indicated using roman numerals, such as, NC5, NC5i, etc.; (2) semantically, based on the meaning of a noun and thus be able to account

for special categories of nouns such as mass nouns, singular-only nouns, and prefix exceptions; (3) revising the entire NC classification to account for all fine-grained details regarding class prefixes, concords, and agreement; and (4) a semantic non-morphological reclassification.

Option (2) requires extensive linguistic analysis of each language, such as further classifying nouns into two categories based on grammatical number: individuated (for singular count nouns), and non-individuated (plural count nouns and mass nouns), which would also apply to singular-only nouns that would no longer be regarded as singular, but rather placed in the plural NC and marked as non-individuated [122]. Option (3) also requires extensive linguistic analysis, such as that presented by Taraldsen [174] for Nguni languages, where it is suggested that the singular/plural pairings manifest a partial flexibility that leaves the concords as the only way of determining a noun's class membership. However, such analyses are beyond the scope of our work and better suited to the domain of linguists. Option (4) has been found to be inconsistent because, as in all languages, all semantic analyses of Bantu NC systems end up with miscellaneous or inconsistent categories [122].

For our immediate computational needs, we applied an approach of morphology and syntax to the regrouping which ensures that there is always a one-to-one singular-plural mapping, but a many to one plural mapping. Table 7.3 shows the regrouped NC systems; the new classes added are those with one or more 'i's after the number.

TABLE 7.3: Regrouped noun classes (NCs). The first line in each pairing is the singular and the other line(s) its plural class (if more than one line is paired, the one with the prefix is applicable, or it is N/A); '-': empty prefix; 'N/A': NC is not present in that language.

NC	chiShona	isiXhosa	isiZulu	Kikuyu	Kinyarwanda	Luganda	Runyankore
1	mu-	u-m-	u-mu	mu-	u-mu-	o-mu-	o-mu-
2	va-	a-ba-	a-ba-	a-	a-ba-	a-ba-	a-ba-
1a	-	u-	u-	-	N/A	N/A	N/A
2a	vana-	oo-	o-	-	N/A	N/A	N/A
2b	a-	N/A	N/A	N/A	N/A	N/A	N/A
1i	mw-	N/A	u-m-	mw-	u-mw-	o-mw-	o-mw-
2i	v-	N/A	N/A	N/A	a-b-	N/A	N/A
2	N/A	N/A	a-ba-	N/A	N/A	a-ba-	a-ba-
2a	N/A	N/A	N/A	-	N/A	N/A	N/A
1ii	N/A	N/A	u-m-	m-	-	-	-
2i	N/A	N/A	a-b-	N/A	N/A	N/A	N/A
2ii	N/A	N/A	N/A	N/A	ba-	ba-	ba-
2	N/A	N/A	N/A	a-	N/A	N/A	N/A
3a	N/A	N/A	u-	N/A	N/A	N/A	N/A
2a	N/A	N/A	o-	N/A	N/A	N/A	N/A
3	mu-	u-m-	u-mu-	mu-	u-mu-	o-mu-	o-mu-
4	mi-	i-mi-	i-mi-	mi-	i-mi-	e-mi-	e-mi-
3i	mw-	N/A	u-m-	m-	u-mw-	o-mw-	o-mw-
4	mi-	N/A	i-mi-	mi-	N/A	N/A	N/A
4i	N/A	N/A	N/A	N/A	i-my-	e-my-	e-my-
5	-	i-	i-	ri-	i-	e-	e-i-
6	ma-	a-ma-	a-ma-	ma-	a-ma-	a-ma-	a-ma-
5i	N/A	i-li-	i-li-	i-	i-ri-	e-li-	e-ri-
6	N/A	a-ma-	a-ma-	ma-	a-ma-	a-ma-	a-ma-
5ii	N/A	N/A	N/A	-	N/A	N/A	N/A
6	N/A	N/A	N/A	ma-	N/A	N/A	N/A
7	chi-	i-si-	i-si-	ki-	i-ki-	e-ki-	e-ki-
8	zvi-	i-zi-	i-zi-	i-	i-bi-	e-bi-	e-bi-
7i	N/A	i-s-	i-s-	gi-	i-cy-	e-ky-	e-ky-
8i	N/A	i-z-	i-z-	N/A	N/A	e-by-	e-by-
8	N/A	N/A	N/A	i-	i-bi-	N/A	N/A
7ii	N/A	N/A	N/A	N/A	i-gi-	N/A	N/A
8	N/A	N/A	N/A	N/A	i-bi-	N/A	N/A
9a	N/A	N/A	i-	-	N/A	N/A	N/A
10a	N/A	N/A	N/A	-	N/A	N/A	N/A
6	N/A	N/A	a-ma-	N/A	N/A	N/A	N/A
9	n-	i-	i-	n-	i-	e-n-	e-n-
10	dzi-	ii-	i-zi-	n-	i-	e-n-	e-n-
9i	N/A	i-n-	i-n-	N/A	i-n-	e-m-	e-m-
10i	N/A	i-zin-	i-zin-	N/A	i-n-	e-m-	e-m-
9ii	-	N/A	N/A	N/A	-	-	-
10ii	-	N/A	N/A	N/A	-	-	-
9iii	N/A	N/A	N/A	N/A	i-zn-	N/A	N/A
10iii	N/A	N/A	N/A	N/A	i-zn-	N/A	N/A
11	ru-	u-	u-	ru-	u-ru-	o-lu-	o-ru-
10	N/A	ii-	i-zi-	n-	N/A	e-n-	e-n-
10i	N/A	N/A	N/A	N/A	i-n-	N/A	N/A
6	ma-	N/A	N/A	N/A	N/A	N/A	N/A
11i	N/A	u-lu-	u-lu-	N/A	u-rw-	o-lw-	o-rw-
10	N/A	ii-	i-zi-	N/A	i-	N/A	N/A
12	N/A	N/A	N/A	N/A	N/A	a-ka-	a-ka-
12	ka-	N/A	N/A	ka-	a-ka-	a-ka-	a-ka-
13	tu-	N/A	N/A	tu-	u-tu-	N/A	N/A
14	N/A	N/A	N/A	N/A	N/A	o-bu-	o-bu-
12i	N/A	N/A	N/A	ga-	a-ga-	a-k-	a-k-
13	N/A	N/A	N/A	tu-	N/A	N/A	N/A
13ii	N/A	N/A	N/A	N/A	u-du-	N/A	N/A
14i	N/A	N/A	N/A	N/A	N/A	o-bw-	o-bw-
13	N/A	N/A	N/A	N/A	N/A	o-tu-	o-tu-
13i	N/A	N/A	N/A	N/A	u-tw-	o-tw-	o-tw-
14	u-	u-bu-	u-bu-	-	u-bu-	o-bu-	o-bu-
6	ma-	N/A	N/A	ma-	N/A	N/A	N/A
14i	N/A	N/A	N/A	N/A	u-bw-	o-bw-	o-bw-

We used the concords and similar prefix features among languages to guide our regrouping. The concords, as explained by Katamba [94] and Maho [122], are used to distinguish between classes. This was very helpful when deciding how to regroup nouns with no class prefixes. Our regrouped noun classes are thus based on the following:

(1) For NCs 1, 3, 11, 13, 14, 17, and 20, instead of using if-statements to handle the differences between vowel-commencing stems (prefix ends in *w*) and consonant-commencing stems (prefix ends in *u*), we regrouped the vowel-commencing stems into a new noun class, NCs 1i, 3i, 11i, 13i, 14i, 17i, and 20i;

(2) For NCs 7 and 8, instead of using if-statements to handle the differences between vowel-commencing stems (prefix ends in *y*) and consonant-commencing stems (prefix ends in *i*), we regrouped the vowel-commencing stems into a new noun class, NCs 7i and 8i;

(3) Nouns without prefixes in the singular and/or plural are placed in a new NC with *ii* such as NCs 1ii, 5ii, 9ii, and 10ii; NC 2ii is the pairing for NC 1ii; and

(4) Instead of using if-statements to handle NCs with multiple class prefixes in a single class (not resulting from conditioning for vowel-commencing stems) such as NC 5, we regrouped the alternative prefix to the new class with an 'i'.

The NC regrouping shown in Table 7.3 addresses the limitations presented in Section 7.3.1 regarding generalizing computational resources to multiple Bantu languages. The problem of how to generalize resources when similar concepts are represented differently in different languages is also addressed by this regrouping. Consider kinship terms that are placed in a separate NC in chiShona, isiXhosa, isiZulu, and Kikuyu, but not in Runyankore and Luganda. These are now regrouped as NC 1ii in Runyankore and Luganda because they do not have a class prefix (for example, *maama* 'mother' and *taata* 'father'). These can be paired as having NC 2ii as their plural (resulting in ***ba****maama* and ***ba****taata*).

The regrouped NC system in Table 7.3 is based on the NC systems of seven languages across five Guthrie zones. However, given that there are over 300 Bantu languages spread over 80 Guthrie zones, we further analyzed the stability of our regrouping criteria to evaluate whether it is consistent with the NCs in other agglutinating Bantu languages. We refer to the research done by Maho [122], which presents a comparative analysis of the Bantu noun class system. We selected this for our evaluation for three main reasons: (1) his work is completely focused on the Bantu noun class system; (2) his analysis is based on data from 333 languages; and (3) all Guthrie zones are represented in the analysis by at least three languages each.

Maho [122] stated that information about noun class membership can be deduced from the concords, because the series of concords are distinct for each noun class. This supports the rationale by which we regrouped the noun classes, which focused on upholding the concordial agreement system.

In our regrouped NC system, we can develop algorithms to handle additive and substitutive pluralization. Examples of additive pluralization in other Bantu languages include: Thimbukushu (***gu****thu* 'camel thorn tree' in NC 14 is pluralized as ***ma****ghuthu*); Chinyanja (***li****lime* 'toungue' in NC 5 is pluralized as ***ma****lilime* in NC 6); and Northern Sotho (***n****ta* 'louse' in NC 9 is pluralized as ***di****nta* in NC 10) [122]. We regroup nouns that undergo additive pluralization as 'ii' (for example, NCs 1ii, 5ii, and 9ii), while nouns that undergo substitutive pluralization are grouped in the original noun class. Since both additive and substitutive pluralization are found to be operational in any given language [122], algorithms based on this regrouping can be generalized to other languages without changing the core rules.

Finally, several exceptional NCs have been observed in specific languages, and their occurrence has been mostly attributed to class mergers and near mergers [122]. However, where this has been evident, it seems to result in a subset of consistent rules for pluralization [122]. Using our regrouped noun classes, where the standard singular prefix as the main NC, and the merged

or near-merged prefix as the regrouped NC with an 'i', it is possible to develop algorithms that result in a deterministic output during noun pluralization.

So, based on our regrouped noun classes, we can develop language-independent algorithms that can handle multiple class prefixes in a single NC, as wel as both additive and substitutive pluralization. Having regrouped the noun classes in a manner that fosters generalizability, we next investigated whether the bootstrap approach could be used to obtain the verbalization patterns in other agglutinating Bantu languages.

7.5 Bootstrapping Language Resources for Agglutinating Bantu Languages

We first obtained natural language descriptions for subsumption and its negation, conjunction, negation of roles, existential quantification, and universal quantification for each of the five languages. We next analyzed the verbalizations to identify the factors affecting verbalization for each language.

7.5.1 Factors Affecting Verbalization

The verbalization of subsumption and its negation, conjunction, negation of roles, existential quantification, and universal quantification in Runyankore and isiZulu [99] is affected by five factors:

(1) The noun class of a concept's name,

(2) The concept name's category,

(3) Whether the concept is atomic or an expression,

(4) The quantifier, and

(5) The position of the concept in the axiom.

From the natural language descriptions of the five constructors in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda, we found that the same factors affect verbalization in these languages. We then investigated intra-zone bootstrappability, (where we use Runyankore and isiZulu as the source languages, and Luganda and isiXhosa as the target languages.

7.5.2 Intra-zone Bootstrappability

In the context of this research, the term 'bootstrappability' refers to how efficiently a target language can be bootstrapped from one or more source languages. Table 7.4 summarizes the similarities and differences in how verbalization in isiXhosa, isiZulu, Luganda, and Runyankore is achieved; such as where the NC is required, or where the category of a concept applies.

In the following sections, we use examples to explain intra-zone bootstrappability.

Subsumption

The verbalization of subsumption ($\sqsubseteq$) as 'is a' in isiXhosa and Luganda depends on several factors as shown in Table 7.4. In isiXhosa, the copulative depends on the NC of C1, the name of the concept to the left of $\sqsubseteq$, and the name of the concept to the right, C2, drops its initial vowel, and the two are combined to form a single word. Luganda, on the other hand, has no copulative in the translation, but C2 drops its initial vowel if it has one. The examples below show this:

Axiom: Panado $\sqsubseteq$ Medicine

TABLE 7.4: Verbalization of the five constructors in isiXhosa, isiZulu, Luganda, and Runyankore (N_c, NC; C_1 and C_2, the concepts to the LHS and RHS of the constructor respectively; R, the roles in the axiom; C_r, the concept quantified over; 'IV', the initial vowel; S_c, R_c, Q_c, P_c, and Neg_c, the subject, relative, quantitative, possessive, and negative concords respectively; E_p, the enumerative prefix; P_a, the prepositional agreement; and P_n, the pronominal)

DL	**English**	**isiZulu**	**isiXhosa**	**Runyankore**	**Luganda**
1. $\sqsubseteq$	... is a... All Each	Category of C_2 R_c1 + *onke* R_c1 + *onke*	N_c1 + category of C_2 R_c1 + *onke nganye*, placed after C_1	Category of C_2 P_c1 + *ona buri*, placed before C_1	Category of C_2 P_c1 + *onna buli*, placed before C_1
$\sqsubseteq \neg$	... is not ...	Neg_c1 + P_n	*asi* + P_a1	Category of C_2	Category of C_2
2. $\sqcap$	And (enumeration) And (connecting clauses)	*na*, *ne*, or *no* *kanye* or *futhi*	*na*, *ne*, or *no* *kweye*	*na* or *n'* *kandi*	*na* or *n'* *Ate*
3. $\exists$	... at least one ... Some	R_c2 + Q_c2 + *dwa* *noma* E_p + *phi*	*nokuba* R_cr + *nye* R_cr + *nye*	*hakiri* ... R_Cr + *mwe hakiri*	*kyeenkana* ... R_cr + *mu ko ku*
4. $\forall$	Only	R_cr + *odwa*	R_cr + *odwa*	P_cr + *onka*	P_cr + *okka*
5. $\neg$ ... R.C	Not	Neg_c, verb conjugation	Neg_c, verb conjugation	*ti*, verb conjugation	*te* or *si*, verb conjugation

English: Panado **is a** medicine

isiXhosa: *Ipanado* ***li****yeza*

Luganda: *Panado mubazi*

In Luganda, *omubazi* 'medicine' drops its initial vowel and becomes *mubazi*. For isiXhosa, the copulative depends on the NC of the name of the noun [89], and if C2 (the name of the concept to the right of $\sqsubseteq$) belongs to NCs 1, 1a, 2, 2a, 3, or 6, the copulative *ng-a/o/u* is placed after the copulative of C1, and is then followed by C2. In the example below, the second copulative is underlined:

English: Panados **are** medicines

isiXhosa: *iipanado* ***zi****ngamayeza*

In the above example, *zi* is the copulative for *iipanado* in NC 10, and *nga* is the copulative for *mayeza* 'medicines' in NC 6.

When voicing the universal quantification in the subsumption, there are alternatives between singular ('each') and plural ('all'). For the singular, both isiXhosa and Luganda have a direct translation, *nganye* and *buli* respectively. However, isiXhosa places *nganye* after C1, while Luganda places *buli* before C1, which drops its initial vowel if it has one. The examples below show this:

Axiom: Panado $\sqsubseteq$ Medicine

English: **Each** Panado is a medicine

isiXhosa: *Ipanado* ***nganye*** *liyeza*

Luganda: ***Buli*** *Panado mubazi*

In the plural, the translation of 'all' for both isiXhosa and Luganda relies on the NC of C1 to obtain the relative concord to form the full word. isiXhosa uses *-onke*, which is placed before C1; while Luganda uses *-onna*, placed after C1. Consider the examples below (the relative concord is underlined):

Axiom: Panado $\sqsubseteq$ Medicine

English: **All** Panado are medicines

isiXhosa: ***Zonke*** *iipanado zingamayeza*

Luganda: *Panado* ***zonna*** *mibazi*

In both isiXhosa and Luganda, the relative concords, *z-* and *za-* respectively, are obtained through the NC. Additionally, this verbalization requires pluralization of both concepts.

Analysis The copulative in isiZulu is *ng* for nouns starting with 'a', 'o', or 'u'; but *y* everywhere else [98]. In Runyankore, the copulative is *ni* if C2 starts with a consonant, and they are written as separate words; but *n'* for nouns starting with a vowel, which are written as a single word [24]. This reliance on a syntactic approach that depends on the first letter of C2 was the basis for bootstrapping from isiZulu to Runyankore.

The Luganda verbalization has some similarities with that of both Runyankore and isiZulu because it also depends on whether C2 starts with a vowel. The two differences however, are: (1) Luganda has no word for the copulative; and (2) the initial vowel is dropped if it exists. The isiXhosa verbalization, on the other hand, is unique, as it requires the NC for the copulative, which the others do not.

When voicing the universal quantification, we see that Runyankore, Luganda, and isiXhosa all have a direct translation for 'each', while isiZulu requires the NC for the relative concord. Additionally, Runyankore and Luganda place the verbalization before C1, which drops its initial vowel. However, isiXhosa places *nganye* after C1.

In the plural, all four languages require the NC for the relative concord. isiZulu and isiXhosa use identical verbalizations for 'all', both using *-onke* placed before C1. Runyankore and Luganda have different translations for 'all', (*-ona* and *-onna* respectively), but both place it after C1 and require phonological conditioning (for example, in Luganda, from *zaonna* to *zonna*).

Negation of Subsumption

The negation of subsumption in isiXhosa requires the NC of C1, the name of the concept to the left of $\sqsubseteq$, to form the negation of the copulative, which comprises the negative prefix *a*, negative affix *si*, and the NC-dependent prepositional agreement); the copulative is omitted [89]. In Luganda, the translation for the negation of the copulative is always *si*, but C2, the name of the concept to the right of $\sqsubseteq\ \neg$, drops its initial vowel if it has one. Below are examples to show this:

Axiom: Cup $\sqsubseteq\ \neg$ Glass

English: A cup **is not a** glass

isiXhosa: *Ikapu* ***asiyo*** *igilasi*

Luganda: *Ekikopo* ***si*** *gilasi*

Analysis In isiZulu, the negation of subsumption omits the copulative, and combines the negative subject concord (which is obtained through the NC) with the pronomial to form the negation [98]. This pattern is similar to that of isiXhosa, but requires the morphemes to be tailored to each language.

Runyankore and Luganda follow the same pattern; both use a direct translation for the negation of the copulative (*ti* and *si* respectively), and C2 drops its initial vowel.

Conjunction

In both isiXhosa and Luganda, the verbalization of $\sqcap$ depends on whether $\sqcap$ is used for enumeration or to join clauses. Luganda makes a further distinction between the items in a list: whether they are only nouns, or whether they contain adjectives. Table 7.5 shows the verbalization of $\sqcap$ in these different situations for each language.

TABLE 7.5: Verbalization of $\sqcap$ in different conditions in isiXhosa and Luganda ('C2', the name of the concept after $\sqcap$ in the axiom; 'IV', initial vowel)

Condition	**isiXhosa**	**Luganda**
(1) Enumeration of nouns	*na*, *ne*, or *no*, depending on IV of C2, and C2 drops IV	*na* or *n'*, depending on IV of C2
(2) Enumeration of adjectives	*na*, *ne*, or *no*, depending on IV of C2, and C2 drops IV	*ate*
(3) Joining clauses	*kweye*	*ate*

Analysis In isiXhosa, isiZulu, Luganda, and Runyankore, the verbalization of $\sqcap$ depends on whether the concept is atomic (enumeration) or an expression (join clauses). In the former, in all these languages, $\sqcap$ is verbalized using *na*, which is altered depending on the first letter of C2. In both isiZulu and isiXhosa, if C2 starts with *i* or *u*, then *na* becomes *ne* or *no* respectively; this is then combined with C2, which drops its initial vowel. Both Luganda and Runyankore use *na* if C2 starts with a consonant, and the two are written as separate words; but *n'* if C2 starts with a vowel, and the two are combined into one word.

Runyankore and Luganda make a further distinction when $\sqcap$ is used for enumeration, conditional on whether the items in the list are only nouns, or contain adjectives. In the former case, both use *na* as explained above, while in the latter case, both use a different verbalization (*kandi* for Runyankore and *ate* for Luganda).

When joining clauses, which are represented as expressions in the DL, the pattern is the same for all four languages, but requires their respective translations for 'and': *kanye* or *futhi* for isiZulu [98], *kweye* for isiXhosa, *kandi* for Runyankore, and *ate* for Luganda.

Existential Quantification

The verbalization of $\exists$ is either '... at least one ...' or 'some'. Both isiXhosa and Luganda have a direct translation for 'at least', but the NC of Cr, the name of the concept quantified over, is required to form the full word for 'one', which is *-nye* for isiXhosa and *-mu* for Luganda. Consider the examples below, where the relative concord is underlined:

Axiom: Teacher $\sqsubseteq$ $\exists$ teaches.Subject

English: Every teacher teaches **at least one** subject

isiXhosa: *Utishala nganye ufundisa isifundo* ***nokuba sinye***

Luganda: *Buli musomesa asomesa* ***kyeenkana*** *esomo* ***limu***

In the above example, 'at least' is translated as *nokuba* in isiXhosa and *kyeenkana* in Luganda. In isiXhosa, 'one' is translated as *sinye*, where *si-* is the relative concord of *isifundo* 'subject' from NC 7. In Luganda, *limu* for 'one' contains *li-*, the relative concord of *esomo* 'subject' from NC 5.

The verbalization of $\exists$ as 'some' is a little different. isiXhosa still verbalizes this using *-nye* prefixed with the relative concord obtained through the NC of Cr, and *nokuba* is dropped from the verbalization. Luganda uses a totally different pattern, conjugating the morpheme *ko* at the end of the verb, thus introducing a notion of 'a portion of' into the semantics; and *ku* is placed before Cr, which drops its initial vowel if it has one. The examples below show this:

Axiom: Panado $\sqsubseteq$ $\exists$ reduces.Pain

English: Every Panado reduces **some** pain

isiXhosa: *Ipanado nganye lihlisa* ***ezinye*** *iintlungu*

Luganda: *Buli Panado ekendeeza****ko ku*** *busaasi*

In the above example, the relative concord *ezi-* for *iintlungu* 'pain' in NC 10 is attached to *-nye* to form the word for 'some' in isiXhosa. Luganda uses *ko* conjugated with the verb, and *ku* in its verbalization, and *obusaasi* 'pain' drops its initial vowel to become *busaasi*.

Analysis Runyankore also has a direct translation for 'at least', *hakiri*, and requires the relative concord of Cr to form the full word for 'one' using *-mwe*. isiZulu on the other hand verbalizes '...at least one...' using the quantitative suffix *-dwa*, which is prefixed by the relative and quantitative concords (both are determined by the NC of Cr) [98].

The pattern for the isiXhosa and Luganda verbalizations is similar to that of Runyankore (except for the positions and translations of the terms). They all have separate translations for 'at least' and 'one'.

The verbalization of $\exists$ as 'some' is also different in Runyankore and isiZulu. Runyankore drops the *-mwe* and only uses *hakiri*, while isiZulu uses *noma*, the enumerative prefix, and *phi* [99]. These two differ from isiXhosa, which requires the NC for the relative concord, and Luganda, which uses *ko* conjugated with the verb.

Universal Quantification

In both isiXhosa and Luganda, the verbalization of $\forall$ depends on the NC of Cr (the name of the concept quantified over) to obtain the relative and possessive concords, respectively. isiXhosa verbalizes $\forall$ as *-odwa*, while Luganda uses *-okka*. This is shown in the examples below (the concords are underlined):

Axiom: Sheep $\sqsubseteq$ $\forall$ eats.Grass

English: All sheep eat **only** grass

isiXhosa: *Zonke iigusha zitya ingca* ***yodwa***

Luganda: *endiga zonna ziry'ebisubi* ***byokka***

For isiXhosa, *y*, the relative concord of NC 9[3] is used with *-odwa*, because Cr is *ingca* 'grass', which is in NC 9. Similarly, in Luganda, the NC 8[4] possessive concord *bya* is used with *-okka*, because Cr, *ebisubi* 'grass', is in NC 8.

[3] See Table F.2 in Appendix F for some concords in isiXhosa.
[4] See Table F.5 in Appendix F for some concords in Luganda.

Analysis Both Runyankore and isiZulu require the NC to obtain the concords with which to verbalize ∀ using *-onka* and *-odwa* respectively. The isiZulu and isiXhosa verbalizations are exactly the same; both require the relative concords. The Runyankore and Luganda verbalizations only differ in the translations; otherwise, the pattern is the same, and they both require the possessive concord. They both also require phonological conditioning (for example, from *byaokka* to *byokka* in Luganda).

Negation of Roles

As shown in Table 7.4, negation of roles in isiXhosa and Luganda is undertaken during verb conjugation [60, 89], and Table 7.1 shows the arrangement of the morphemes in the conjugated verb. Luganda uses two different morphemes during negation, *te* or *si*, the latter only used if the subject is the first person singular [60]. isiXhosa on the other hand depends on the NC to obtain the negative subject concord [175].

Analysis In both Runyankore and isiZulu, the negation of roles is undertaken during verb conjugation [25, 100]. However, whereas Runyankore has a single negation morpheme, *ti* [177], isiZulu requires the noun class to obtain the correct negative subject concord [100]. When considering how the negation morpheme is obtained, isiZulu and isiXhosa have the same pattern that requires the NC, while Runyankore and Luganda have a similar pattern with direct translations.

Discussion on Intra-zone Bootstrappability

Our analysis of the verbalizations of isiXhosa and Luganda throughout Section 7.5.2 is aimed at identifying whether it is indeed more efficient to bootstrap between languages in the same Guthrie zone. We have categorized bootstrappability into three broad categories: (1) same pattern and same lexicon; (2) same pattern but different lexicon; and (3) different pattern and different lexicon.

Category (1) is evident between isiZulu and isiXhosa in the verbalization of subsumption in the plural, conjunction, and universal quantification; as well as between Runyankore and Luganda when ⊓ is used to enumerate a list of nouns.

Category (2) can be seen in the verbalization of subsumption in the singular and plural, negation of subsumption, conjunction when the concept is an expression or adjective, and for universal quantification, when bootstrapping from Runyankore to Luganda. This is the same case for isiZulu and isiXhosa when verbalizing conjunction when the name of the concept is an expression.

Another similarity in this category between Runyankore and Luganda is the need for phonological conditioning when verbalizing subsumption in the plural and universal quantification. In Sections 3.5 and 5.5, we explained that Runyankore uses both vowel coalescence and elision for *-ona*, while only vowel elision is used in the presence of a nasal compound (as is the case with *-onka*). However, Luganda uses vowel elision for both *-onna* and *-okka*.

Categories (1) and (2) offer some evidence in support of intra-zone bootstrappability being the better option, though they do not quantify by how much. Category (3) on the other hand provides mixed results: for example, the patterns for the verbalizations of the copulative in isiXhosa and Luganda are different from those of Runyankore and isiZulu. However, the Luganda pattern shows more similarity to the Runyankore and isiZulu patterns, which are all based on the first letter of C2. On the other hand, isiXhosa's is NC-dependent, unlike the rest.

Additionally, the verbalization of existential quantification as '... at least one...' in Luganda and isiXhosa is more similar to the Runyankore pattern than the isiZulu one. This is because the three have separate translations for 'at least' and 'one', while isiZulu has a single verbalization. In each case, however, we cannot yet quantify by how much bootstrapping from one source language instead of another improves the efficiency of verbalization.

Similar patterns between languages in different zones have been observed, and this is inter-zone bootstrappability. In the following section, we investigate further across five Guthrie zones[123].

7.5.3 Inter-zone Bootstrappability

For this investigation, we selected three languages from three different Guthrie zones: chiShona (zone S.10), Kikuyu (zone E.50), and Kinyarwanda (zone J.60) (see Figure 7.1 in Section 7.2). Table 7.6 summarizes the factors that affect the verbalization of the five selected constructors.

TABLE 7.6: Verbalization of the five constructors in chiShona, Kikuyu, and Kinyarwanda (C_1 and C_2, the concepts to the LHS and RHS of the constructor respectively; R, the roles in the axiom; C_r, the concept quantified over; 'IV', the initial vowel; R_c and P_c, the relative and possessive concords respectively)

DL	English	chiShona	Kikuyu	Kinyarwanda
1. $\sqsubseteq$	...is a...	-	*ni*	*ni*, Depends on IV of C_2
	All	R_c1 + *ose*	P_c1 + *othe*	P_c1 + *ose*
	Each	R_c1 + *ese*	*oo*	*buri*
$\sqsubseteq$ $\neg$	...is not...	*ha* + R_c1 + *si*	*ti*	*ntabwo ari*, IV of C_2
2. $\sqcap$	And, Enumeration	*na* or *ne*	*na*	*na* or *n'*, IV of C_2
	Joining clauses	*uye*	*na*	*kandi*
3. $\exists$	...at least one...	R_cr + *mwe chete zvichienda mberi*	R_cr + *mwe kana-makeria*	*nibura* R_Cr + *mwe*
	Some	R_cr + *mwe*	R_cr + *mwe*	R_cr + *ke*
4. $\forall$	Only	*chete*	*tu*	*gusa*
5. $\neg$ ... R.C	Not	*ha*, verb conjugation	*nd* or *ti*, verb conjugation	*nti*, verb conjugation

We present the details in the following sections, where Runyankore and isiZulu are considered as source languages.

Subsumption

Subsumption ($\sqsubseteq$), verbalized as 'is a' in English, is *ni* in both Kikuyu and Kinyarwanda. Also, in Kinyarwanda, *ni* becomes *n'* if C2, the name of the concept to the right of $\sqsubseteq$, starts with a vowel. chiShona, on the other hand, has no translation for the copulative. This is shown in the examples below:

Axiom: Panado $\sqsubseteq$ Medicine

English: Panado **is a** medicine

chiShona: *Panado mushonga*

Kikuyu: *Panado* ***ni*** *dawa*

Kinyarwanda: *Panado* ***n'****umuti*

When the universal quantification in the subsumption is voiced, there are two possible verbalizations: singular, 'each', or plural, 'all'. chiShona, Kikuyu, and Kinyarwanda all have separate translations for both cases. 'Each' is *-ese* in chiShona, and requires the NC of C1, the name of the concept to the left of $\sqsubseteq$, to form the full word. Kikuyu and Kinyarwanda both have direct translations for 'each': *oo* and *buri*, respectively. The examples below show this (the relative concord is underlined):

Axiom: Panado $\sqsubseteq$ Medicine

English: **Each** Panado is a medicine

chiShona: *Panado* ***yese*** *mushonga*

Kikuyu: ***Oo*** *Panado ni dawa*

Kinyarwanda: ***Buri*** *Panado n'umuti*

For chiShona, *y*, the relative concord of NC 9 to which *panado* belongs, is required for the verbalization. In addition to having a direct translation for 'each', both Kikuyu and Kinyarwanda place it before C1, while chiShona places it after C1.

The plural 'all' require the NC of C1 to obtain the appropriate concords in the three languages. Both chiShona and Kinyarwanda verbalize 'all' as *-ose*, while Kikuyu uses *-othe*. Consider the examples below (the concords are underlined):

Axiom: Panado $\sqsubseteq$ Medicine

English: **All** Panado are medicines

chiShona: *Panado* ***dzose*** *mushonga*

Kikuyu: *Panado* ***ciothe*** *ni dawa*

Kinyarwanda: *Panado* ***zose*** *n'imiti*

The verbalization is placed after C1 in all these languages. chiShona's requires the relative concord *dz*, while Kikuyu and Kinyarwanda verbalizations require the possessive concord, *cia* and *za*, respectively. These two also require phonological conditioning through vowel elision.

Analysis We have seen in Section 7.5.2 that the verbalization of the copulative in Runyankore and isiZulu is determined by the first letter of C2. Kinyarwanda has the exact same verbalization as Runyankore, while Kikuyu, like Runyankore, uses *ni*, though it is not affected by C2. chiShona's verbalization, which has no copulative, is different from both Runyankore and isiZulu.

When voicing the universal quantification in the singular, both Runyankore and Kinyarwanda are exactly the same, as they both use *buri*, placed before C1, which drops its initial vowel. Kikuyu is similar, in that it also has a direct translation (*oo*) and places it before C1. On the other hand, the patterns for isiZulu and chiShona are the same; both require the NC for the full translation and place the verbalization after C1. There is, however, a need to change *-onke* in isiZulu to *-ese* in chiShona.

For the plural form, all five languages require the NC during verbalization. Runyankore (with *-ona*), Kikuyu (*-othe*), and Kinyarwanda (*-ose*) use the possessive concord, and require phonological conditioning. isiZulu (*-onke*) and chiShona (*-ose*) require the relative concord. Unlike isiZulu, however, the rest place the verbalization after C1.

Negation of Subsumption

Kikuyu and Kinyarwanda both have direct translations for the negation of subsumption: *ti* and *ntabwo ari*, respectively. The chiShona verbalization requires the NC to obtain the relative concord of C1, the name of the concept to the left of $\sqsubseteq \neg$. Consider the examples below (the relative concord is underlined):

Axiom: Cup $\sqsubseteq \neg$ Glass

English: A cup **is not a** glass

chiShona: *Mukombe* ***hausi*** *girazi*

Kikuyu: *Gikobe* ***ti*** *girathi*

Kinyarwanda: *Igikombe* ***ntabwo ari*** *ikirahure*

In chiShona, the relative concord *u* from NC 3 (the NC of *mukombe* 'cup') is used with the negation morpheme *ha* and the affix *si*, to negate the copulative.

Analysis Similar to Kikuyu, Runyankore negates the copulative with *ti*, but additionally requires C2 to drop its initial vowel. Kinyarwanda also has a direct translation, but C2 does not drop its initial vowel. Both isiZulu and chiShona's verbalizations require concords that are obtained through the NC.

Conjunction

Both chiShona and Kinyarwanda verbalize conjunction differently when used to enumerate list items or join clauses, while Kikuyu uses the same verbalization in both cases. Table 7.7 shows the verbalization of ⊓ in these different situations for each language.

TABLE 7.7: Verbalization of ⊓ in different conditions in chiShona, Kikuyu, and Kinyarwanda ('C2', the name of the concept after ⊓ in the axiom; 'IV', initial vowel)

Condition	chiShona	Kikuyu	Kinyarwanda
(1) Enumeration of nouns	*na*	*na*	*na* or *n'*, depending on IV of C2
(2) Enumeration of adjectives	*na*	*na*	*kandi*
(3) Joining clauses	*uye*	*na*	*kandi*

Analysis Runyankore, isiZulu, chiShona, and Kinyarwanda verbalize ⊓ differently depending on whether the concept is atomic or an expression. In the former case, all five languages use *na* for the verbalization. Additionally, both isiZulu and chiShona require phonological conditioning here. Further, the Kinyarwanda verbalization is exactly the same as that of Runyankore in terms of the lexicon and conditions that apply.

Existential Quantification

Existential quantification can either be verbalized as '...at least one...' or 'some'. When verbalized as '...at least one...', all three languages have a direct translation for 'at least', but the NC of Cr, the name of the concept quantified over, is required to obtain the relative concord for the full verbalization of 'one'. This is shown in the examples below (the relative concord is underlined):

Axiom: Teacher ⊑ ∃ teaches.Subject

English: Every teacher teaches **at least one** subject

chiShona: *Mudzidzisi wese anodzidzisa chidzidzo* ***chimwe chete zvichienda mberi***

Kikuyu: *Oo mwarimu athomithagia ithomo* ***rimwe kanamakeria***

Kinyarwanda: *Buri Mwalimu yigisha* ***nibura*** *isomo* ***rimwe***

The direct translation for 'at least' is *chete zvichienda mberi* in chiShona, *kanamakeria* in Kikuyu, and *nibura* in Kinyarwanda. All three languages translate 'one' as *-mwe* and the relative concord of Cr ('Subject', in the above example) is required for the full translation: *chi-* from NC 7 in chiShona, and *ri* from NC 5 in Kikuyu and Kinyarwanda.

The verbalization of ∃ as 'some' is the same in chiShona and Kikuyu, which both drop 'at least' but retain 'one' (*-mwe*). Kinyarwanda translates 'some' as *-ke*, which requires the NC of Cr for the full verbalization. Consider the examples below (the relative concord is underlined):

Axiom: Panado ⊑ ∃ reduces.Pain

English: Every Panado reduces **some** pain

chiShona: *Panado yese inoderedza* ***ma<u>mwe</u>*** *marwadzo*

Kikuyu: *Oo panado enyihagia ruo* ***<u>ru</u>mwe***

Kinyarwanda: *Buri panado ugabanya uburibwe* ***<u>bu</u>ke***

Analysis The verbalization of ∃ as '...at least one...' in chiShona, Kikuyu, and Kinyarwanda is similar to that of Runyankore, which also has separate translations for 'at least' and 'one'. Additionally, these four languages translate 'one' as *-mwe* and it requires the NC to obtain the relative concord. The differences among their verbalizations lie in the lexicon used to translate 'at least' and in the placement of terms in the verbalization: before the verb, after the verb, before Cr, or after Cr.

When verbalizing ∃ as 'some', isiZulu, chiShona, Kikuyu, and Kinyarwanda require the NC of Cr. The difference lies in the lexicon of each language, and the concords required; the enumerative prefix in isiZulu, and relative concord for chiShona, Kikuyu, and Kinyarwanda.

Universal Quantification

Universal quantification (∀), verbalized as 'only' in English, has direct translations in chiShona, Kikuyu, and Kinyarwanda: *chete*, *tu*, and *gusa*, respectively. This can be seen in the examples below:

Axiom: Sheep ⊑ ∀ eats.Grass

English: All sheep eat **only** grass

chiShona: *Makwai ose anodya uswa* ***chete***

Kikuyu: *Gondu ciothe iriaga nyeki* ***tu***

Kinyarwanda: *Intama zose zirya* ***gusa*** *ibyatsi*

Analysis Both Runyankore and isiZulu require the NC to obtain the appropriate concord with which to verbalize ∀. The patterns for verbalization in chiShona, Kikuyu, and Kinyarwanda are different from those of Runyankore and isiZulu.

Negation of Roles

As shown in Table 7.6, negation of roles in chiShona, Kikuyu, and Kinyarwanda happens during verb conjugation [88, 128, 140], and Table 7.1 shows the arrangement of the morphemes in the conjugated verb. chiShona and Kinyarwanda each have a single negation morpheme: *ha* and *nti*, respectively [128, 140]. Kikuyu has two morphemes for negation, and they depend on the nature of the subject concord: if the subject concord is a vowel only, then *nd* is used and it is placed before the subject concord; but in all other cases, *ti* is used and it is placed after the subject concord [88].

Analysis Runyankore, like chiShona and Kinyarwanda, has a single negation morpheme, *ti*. Negation in Kikuyu bares some similarity to Runyankore, because it has *ti* as one of its translations. And although further processing is required to check the nature of the subject concord, Kikuyu's verbalization is still different to that of isiZulu, which requires the NC to obtain the correct negative concord..

We make no further comparisons because the verbal morphologies of these languages differ in the number, placement, and lexicon of the morphemes used.

Discussion on Inter-zone Bootstrappability

The analysis presented in Section 7.5.3 offers more evidence for inter-zone bootstrappability. The three broad categories presented in Section 7.5.2 also apply here: (1) same pattern and same lexicon; (2) same pattern but different lexicon; and (3) different pattern and different lexicon. We also identified a fourth category in inter-zone bootstrapping: different pattern but same lexicon.

Category (1) is evident only between Runyankore and Kinyarwanda for the verbalization of the copulative, subsumption in the singular, and conjunction.

Category (2) can be seen between isiZulu and chiShona for the verbalization of negation of subsumption and conjunction. This category is also observed between Runyankore and chiShona, Kikuyu, and Kinyarwanda for the verbalization of subsumption in the plural, and for existential quantification as '...at least one...'.

Category (3) is observed between Runyankore and Kinyarwanda for the verbalization of the negation of subsumption; as well as between both source languages and all target languages for the verbalization of existential quantification as 'some', universal quantification, and negation of roles.

Category (4), same lexicon but different pattern, is evident between Runyankore and Kikuyu for the verbalization of the copulative and for the negation of subsumption. This is due to Kikuyu not having the augment, which Runyankore drops under some conditions.

Though these categories enable us to group different bootstrapping possibilities, they do not reveal the degree to which a target language is bootstrappable from each source language.

We therefore used the analyses carried out for both intra-zone and inter-zone bootstrappability to investigate how to quantify the bootstrap process, in an attempt to measure bootstrappability.

7.6 Measuring Bootstrappability

Bootstrappability refers to how efficiently a target language can be bootstrapped from one or more source languages. In our investigation into bootstrappability, we have used two source languages, and this raises the question of whether, for a given target language, it is more efficient to bootstrap from Runyankore or isiZulu. We therefore need to identify the most efficient source–target language pair. To do this, we investigated whether the bootstrap process can be measured, and discovered that by considering the nature of actions required to convert the source language pattern to the target language pattern, we can categorize bootstrappability.

Based on the broad categories into which bootstrappability was placed in Sections 7.5.2 and 7.5.3, we developed an ordinal scale that identifies five levels of bootstrappability, which represent the extremes, from Level1, 100% the same pattern, to Level5, 100% not bootstrappable. Further, based on the findings from Sections 7.5.2 and 7.5.3, we assigned events to these five levels as below:

Level1: 100% bootstrappable; same pattern and same lexicon;

Level2: Same pattern, but different lexicon;

Level3: Different pattern, but same lexicon;

Level4: Different pattern and different lexicon;

Level5: Not bootstrappable

This categorization is intended to convey that the difficulty of bootstrapping increases as the levels increase from 1 to 5. It can be interpreted as, at Level1 bootstrappability, no action is taken to convert from source to target pair; essentially, the source pattern is directly reused to generate text in the target language. On the other extreme, Level5 can be interpreted as requiring more effort to bootstrap than would be required to create the target language pattern from scratch. In between these two extremes, we have levels 2, 3, and 4 representing increasing effort from end to end.

However, though this categorization helps to frame the problem of bootstrapping according to effort, it does not quantify it in a manner that would enable us to precisely identify the more efficient bootstrap pair. We therefore investigated whether, by considering the actions performed when converting a source language pattern to that of the target language, we can assign weights to these actions, thus quantifying bootstrappability. We relied on two key concepts here: minimum edit distance [91] and linear gap penalty [66].

7.6.1 Minimum Edit Distance

Our task is to quantify the number of edit operations required to convert a source language verbalization pattern to a target language pattern. This is very similar to 'edit distance', which measures the operations required to transform one string into another [91]. We thus applied the concept of minimum edit distance, but in the context of transforming one pattern into another, and adopted the use of the three operations (insertion, deletion, and substitution [91]) to our context, as below:

(1) Substitution is used to tailor the lexicon from that of the source language to that of the target language;

(2) Insertion is used to insert a new step into the pattern–a step that the source pattern lacks but is required in the target pattern; and

(3) Deletion is used to remove a step in the source pattern that does not apply in the target pattern.

When we apply these operations to the events associated with each level of bootstrappability, we get the following:

Level1: no operations;

Level2: Same pattern, but different lexicon (substitution only);

Level3: Different pattern, but same lexicon (deletion and/or insertion only);

Level4: Different pattern and different lexicon (substitution and deletion, and/or insertion); and

Level5: Not bootstrappable

When measuring minimum edit distance, weights are usually assigned to each operation. The simplest weighting factor, the Levenshtein distance, assigns a weight of one to each operation [91, 115]. However, we consider tailoring the lexicon in the pattern (substitution) to require less effort than deletion or insertion, because they alter the structure of the pattern. Therefore, in order to quantify the weights for creating a gap (deletion) or extending a gap (insertion) in the pattern, we used Linear Gap Penalty [66].

7.6.2 Linear Gap Penalty

We applied Gotoh [66]'s definition of the Linear Gap Penalty used to compare protein sequences by calculating the distance of converting one sequence to another. We find this metric adequate for our task because: (1) it focuses on quantifying deletions and insertions (which alter a pattern) in the sequence, which we are interested in; and (2) Gotoh [66]'s metric is a more efficient update to previous metrics in this area. Additionally, we found that verbalization patterns can be regarded to have some similarity to the biological sequences that this metric is intended for, because in both the sequence of steps is important.

Gotoh [66]'s definition of the linear gap penalty is: $W = G + XL$, where G is the gap opening penalty, X is the gap extension penalty, and L is the length of the gap.

We associated the linear gap penalty with any operation that creates or extends a gap in the pattern, that is, deletion and insertion, where the former creates a gap in the pattern, and the latter both creates and extends a gap in the pattern. Therefore, the weight of deletion will only be the gap opening penalty G, while that of insertion will be W, the linear gap penalty that accounts for both opening and extending a gap.

7.6.3 Bootstrappability Metric

Having found a way to quantify the operations required to transform a source pattern to a target pattern, we assigned the following weights to each operation:

- Substitution $S = n$, where n is a real number;
- Deletion $D = G$, the gap opening penalty; and
- Insertion $I = W = (G + XL)$, the linear gap penalty.

Where: $n < G$, $n < R$, and $G > R$.

The measure of bootstrappability, B, is therefore the minimum number of operations required to transform a source pattern to a target pattern, and can be obtained by summing the number of operations used for bootstrapping as below: $B = \Sigma S + \Sigma D + \Sigma I$. The higher the bootstrappability value, the more the effort required for bootstrapping. Therefore, Level1, which represents 100% bootstrappable, is always 0, as it requires no operations. This measure of bootstrappability continues to increase until Level5. Additionally, when there are two or more source patterns from which to bootstrap, this metric can be used to determine the most efficient source–target bootstrap pair.

7.6.4 Application of Bootstrappability Metric

We applied the above measure of bootstrappability to the analysis carried out for intra-zone bootstrappability in Section 7.5.2 and inter-zone bootstrappability in Section 7.5.3. Following the weighting criteria explained in Section 7.6.3, we assigned arbitrary initial weights to the variables in the metric: $n = 0.5$, $G = 1$, and $X = 0.75$[5]; and obtained the following results.

Intra-zone Bootstrappability

From the analyses in Section 7.5.2, we represented the verbalization pattern of each constructor for each source language as a sequence of steps, and then showed how the three operations (substitution, deletion, and insertion) can be used to transform the source pattern to the target pattern. We then calculated B, the measure of bootstrappability, for each source–target language pair. Table 7.8 shows the results obtained.

[5]Different values for the weights were tested, resulting in consistent conclusions. The resources of this further analysis as well as more detailed analyses, including the steps for each language and other values considered for n, G, and W, are available at `https://github.com/ThesisResources/BootstrappabilityResults`.

TABLE 7.8: Measures of intra-zone bootstrappability for each source–target language bootstrap pair (n, the weight for substitution; G, the gap penalty; W, the linear gap penalty; and L, the length of the extension)

DL	English	Runyankore		isiZulu	
		isiXhosa	**Luganda**	**isiXhosa**	**Luganda**
$\sqsubseteq$	is a	Level5	1G + 1W (L = 3) + 1n = 4.75	Level5	1G + 1W (L = 3) + 2n = 5.25
	each	1G + 2n = 2	1n = 0.5	2G + 2n = 3	2G + 1n + 1W (L = 3) = 4.75
	all	2n = 1	1n = 0.5	0	2n = 1
$\sqcap$	and (enumeration)	2G + 1W (L = 3) + 2n = 6.25	1n = 0.5	0	3G + 2W (L = 4) + 2n = 10
	and (joining clauses)	1n = 0.5	1n = 0.5	1n = 0.5	1n = 0.5
$\exists$	at least one	3n = 1.5	3n = 1.5	1G + 1W (L = 3) + 1n = 3.75	1G + 1W (L = 3) + 1n = 3.75
	some	2W (L = 3) + 1n = 5	1W (L = 1) + 2n = 2.75	1G + 3n = 2.5	3G + 1W (L = 1) + 2n = 5.75
$\forall$	only	1n = 0.5	1n = 0.5	0	1n = 0.5

From the results in Table 7.8, we conclude the following for intra-zone bootstrappability:

(1) Intra-zone bootstrappability is more efficient than inter-zone bootstrappability between Runyankore and Luganda for the verbalization of $\sqsubseteq$ in both singular and plural, conjunction, and $\exists$ as '...at least one...' and 'some'; as well as between isiZulu and isiXhosa for plural $\sqsubseteq$, conjunction, $\exists$ as 'some', and $\forall$.

(2) Inter-zone bootstrappability is more efficient than intra-zone bootstrappability when Runyankore, instead of isiZulu, is used to bootstrap the verbalizations of $\sqsubseteq$ as 'each' and $\exists$ as '...at least one...' in isiXhosa.

(3) Sometimes intra-zone and inter-zone bootstrappability are equally efficient, such as in the verbalization of $\forall$ for Luganda and $\sqcap$ for joining clauses for both source and target languages.

(4) The verbalization of the copulative in isiXhosa is marked as Level5 because bootstrapping from either Runyankore or isiZulu requires deleting all the steps in the source pattern, followed by inserting the isiXhosa steps to form the new pattern. It would therefore be more efficient to create the new pattern (associated only with the weights of inserting new steps) than to bootstrap (associated with both the weights of deleting all the source steps plus inserting new steps). For this reason, we regard it as not bootstrappable from either source language.

Inter-zone Bootstrappability

From the analyses in Section 7.5.3, we represented the verbalization pattern of each constructor for each source language as a sequence of steps, and then showed how the three operations (substitution, deletion, and insertion) can be used to transform the source pattern to the target pattern. We then calculated B, the measure of bootstrappability, for each source–target language pair. Table 7.9 shows the results obtained (more detailed analysis, including the steps for each language and other values considered for n, G, and X, are available at `https://github.com/ThesisResources/BootstrappabilityResults`).

From the results shown in Table 7.9, we have concluded the following on inter-zone bootstrappability:

TABLE 7.9: Measures of inter-zone bootstrappability for each source–target language bootstrap pair (n, the weight for substitution; G, the gap penalty; W, the linear gap penalty; and L, the length of the extension).

DL	English	Runyankore			isiZulu		
		chiShona	**Kikuyu**	**Kinyarwanda**	**chiShona**	**Kikuyu**	**Kinyarwanda**
$\sqsubseteq$	is a	2G + 1n = 2.5	2G = 2	0	2G + 1n = 2.5	2G + 1n = 2.5	3n = 1.5
	each	2G + 2W (L = 3) = 8.5	1G + 1n = 1.5	0	2n = 1	2G + 1n = 2.5	2G + 1n + 1W (L = 3) = 4.75
	all	1n = 0.5	1n = 0.5	1n = 0.5	2n = 1	2n = 1	2n = 1
$\sqcap$	and (enumeration)	4G + 1n = 4.5	4G = 4	0	5G = 5	5G = 5	3G + 2W (L = 4) + 2n = 10
	and (joining clauses)	1n = 0.5	1n = 0.5	0	1n = 0.5	1n = 0.5	1n = 0.5
$\exists$	at least one	2n = 1	2n = 1	2n = 1	1G + 1W (L = 3) + 1n = 3.75	1G + 1W (L = 3) + 1n = 3.75	1G + 1W (L = 3) + 1n = 3.75
	some	2W (L = 3) + 1n = 5	2W (L = 3) + 1n = 5	2W (L = 3) + 1n = 5	1G + 3n = 2.5	1G + 3n = 2.5	1G + 3n = 2.5
$\forall$	only	2G + 1n = 2.5	2G + 1n = 2.5	2G + 2n = 3	2G + 1n = 2.5	2G + 1n = 2.5	2G + 2n = 3

(1) Bootstrappability is more efficient with Runyankore as the source language for the verbalizations of: the copulative and singular $\sqsubseteq$ in Kikuyu and Kinyarwanda; $\sqcap$ for enumeration, plural $\sqsubseteq$, and $\exists$ as '...at least one...' in all three target languages; as well as $\sqcap$ for joining clauses in Kinyarwanda.

(2) isiZulu is a more efficient source language for bootstrapping the verbalizations of: singular $\sqsubseteq$ in chiShona; and $\exists$ as 'some' in all three target languages.

(3) Both Runyankore and isiZulu are equally efficient as source languages when bootstrapping the verbalizations of: the copulative in chiShona; $\sqcap$ for joining clauses in chiShona and Kikuyu; and $\forall$ in all three target languages.

7.7 General Discussion

We carried out this investigation in order to answer RQ4 from Section 1.2.1 in Chapter 1 by finding out whether the Runyankore and isiZulu approaches to ontology verbalization can be generalized to other agglutinating Bantu languages.

7.7.1 Noun Class Reclassification

When answering RQ4 (a.1) about the limitations of applying Meinhof's NC system to computational tasks, we found the same limitations present in other Bantu languages. We therefore used a morphological and syntactic approach to regroup the noun classes from 29 classes (Table 7.2) to 54 classes (Table 7.3). The regrouped noun classes ensure a deterministic outcome during noun pluralization, and can handle additive and substitutive pluralization in multiple languages using language-independent rules. Our NC regrouping also respects the details on concordial agreement in Meinhof's noun classification, which are important for verbalization and verb conjugation.

7.7.2 Bootstrappability

When answering RQ4 (a.2) about bootstrappability, we divided our investigation into two parts: intra-zone bootstrappability and inter-zone bootstrappability. Under the former, we selected Luganda and isiXhosa, in the same zones as Runyankore and isiZulu respectively, to investigate whether it is more efficient to bootstrap between languages in the same zone. For inter-zone bootstrappability, we selected chiShona, Kikuyu, and Kinyarwanda to investigate whether it is always possible to bootstrap among languages from different zones.

We investigated whether the same five factors that affect the verbalization of $\mathcal{ALC}$ in Runyankore and isiZulu also apply to these five languages, and found that they do indeed. The factors are:

(1) The noun class of a concept's name affects the verbalization of subsumption in the plural and existential quantification for all five languages; negation of subsumption in chiShona and isiXhosa; universal quantification in isiXhosa and Luganda; singular subsumption in chiShona; as well as the copulative and negation of roles in isiXhosa.

(2) The concept name's category affects the verbalization of the copulative and singular subsumption in Kinyarwanda and Luganda; and $\sqcap$ for enumeration in all languages but Kikuyu.

(3) Whether the concept is atomic or an expression affects the verbalization of $\sqcap$ in all languages but Kikuyu.

(4) The quantifier affects the verbalization of $\exists$ and $\forall$ in all languages.

(5) The position of the concept in the axiom affects the verbalization of $\exists$ in all languages; and $\forall$ in isiXhosa and Luganda.

Intra-zone Bootstrappability

The results of the analysis presented in Section 7.5.2 as well as the results in Table 7.8 show that, for most cases, the Guthrie zone does affect bootstrappability, as we found intra-zone bootstrappability to be more efficient than inter-zone bootstrappability. The metric developed in Section 7.6 answers RQ4 (a.3) concerned with measuring bootstrappability. For example, in the verbalization of the copulative in Luganda, inter-zone bootstrappability is 8.5 times more costly than intra-zone bootstrappability.

Further, quantifying bootstrappability enables us to measure the cumulative weight of an entire axiom; for example, $\mathsf{Panado} \sqsubseteq \mathsf{Medicine}$, has a cumulative weight of 5.75 in the singular and plural for Luganda with Runyankore as the source language; but 9.5 in the singular and 6.25 in the plural with isiZulu as the source language. What is important about these results is that the least 'costly' source language can be identified for an entire axiom. It is also possible to set a value beyond which a source language can be regarded as not bootstrappable. For the case of the copulative in isiXhosa, we regard it as Level5 because bootstrapping essentially incurs the cost of throwing out the source pattern plus inserting the target pattern.

We can extend this to measure even more complex axioms, such as: $\mathsf{Old\_Lady} \sqsubseteq \exists\, \mathsf{reads.Publication} \sqcap \forall\, \mathsf{reads.Tabloid}$, which has a cumulative weight of 3 in the singular and plural for Luganda with Runyankore as the source language; but 9 in the singular and 5.75 in the plural with isiZulu as the source language. The calculation involves the cumulative weight of verbalizing 'each/all', '...at least one...', joining clauses, and 'only'.

Inter-zone Bootstrappability

The results of the analysis presented in Section 7.5.3 as well as the results in Table 7.9 show that inter-zone bootstrappability is possible for our selected source and target languages. Additionally,

the metric developed in Section 7.6 enables us to select the most efficient source–target language pair, that is, the one with the least weight. For example, in the verbalization of $\exists$ as 'some' in chiShona, Kikuyu, and Kinyarwanda, using Runyankore as the source language is twice as costly as using isiZulu.

Further, quantifying bootstrappability enables us to measure the cumulative weight of an entire axiom. For example, $\mathsf{Panado} \sqsubseteq \mathsf{Medicine}$, has a cumulative weight of 3.5 in the singular and plural for chiShona with isiZulu as the source language; but 11 in the singular and 3 in the plural with Runyankore as the source language.

We can extend this to measure even more complex axioms, such as: $\mathsf{Old\_Lady} \sqsubseteq \exists\ \mathsf{reads.Publication} \sqcap \forall\ \mathsf{reads.Tabloid}$, which has a cumulative weight of 7.75 in the singular and plural for chiShona with isiZulu as the source language; but 12.5 in the singular and 4.5 in the plural with Runyankore as the source language.

Our measure of bootstrappability makes it possible to identify the more efficient source–target language pair for an individual constructor (singular or plural) or an entire axiom.

7.7.3 Other Considerations

Throughout this chapter, we have used Runyankore and isiZulu as our source languages because they were the only available ones at the time. However, having identified the verbalization patterns for chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda, they can now also be regarded as source languages. For some verbalizations, this can result in Level1 bootstrappability; for example, between chiShona and Luganda that both have no word for the copulative; between chiShona and Kinyarwanda for 'all', as they both use *-ose*; and between Kikuyu and Luganda during negation of roles, as they both have two negation morphemes that depend on the structure of the subject concord.

7.8 Summary

In this chapter, we investigated the generalizability and bootstrappability of the Runyankore and isiZulu verbalization approaches, to other agglutinating Bantu Languages, in order to answer the first part of RQ4, "How can the Runyankore and isiZulu approaches to ontology verbalization be generalized to other agglutinating Bantu languages" from Section 1.2.1 in Chapter 1. We selected five languages: chiShona (zone S.10), isiXhosa (zone S.40), Kikuyu (zone E.50), Kinyarwanda (zone J.60), and Luganda (zone J.10). We used Luganda and isiXhosa (in the same zones as Runyankore and isiZulu, respectively) to investigate whether it is more efficient to bootstrap between languages in the same zone (intra-zone bootstrappability). We used chiShona, Kikuyu, and Kinyarwanda in the investigation into bootstrapping among languages in different zones (inter-zone bootstrappability).

We analyzed the noun class systems of these five languages, and found that, like in Runyankore and isiZulu, Meinhof's NC system has several limitations when applied to computational tasks. We thus revised Meinhof's NC system by adding 25 extra classes to account for multiple class prefixes in a single NC, and additive and substitutive pluralization. We also showed that our revised classification can support the NC systems of other Bantu languages.

When investigating bootstrappability, we verbalized five constructors in the DL $\mathcal{ALC}$ in five languages and analyzed their verbalizations. From these results, we were able to categorize bootstrappability into five levels, and develop a metric to quantify bootstrappability, and then show that, for most constructors, intra-zone bootstrappability is more efficient than inter-zone bootstrappability. We also used the metric to identify the more efficient source–target language pair when there are multiple source languages.

In the next chapter, we further investigate generalizability to find out whether the Runyankore and isiZulu approaches to noun pluralization and verb conjugation are applicable to other agglutinating Bantu languages.

Chapter 8

Architecture for API for Agglutinating Bantu Languages

8.1 Overview

In Chapter 7, we showed how the bootstrap process can be used to reduce the time and effort required to obtain verbalization patterns for new languages. We also regrouped Meinhof's noun class (NC) system in order to foster generalizability. In this chapter, we continue to answer RQ4, 'How can the Runyankore and isiZulu approaches to ontology verbalization be generalized to other agglutinating Bantu languages' from Section 1.2.1 in Chapter 1. Here, we investigate the generalizability of the approaches taken for noun pluralization and verb conjugation (having obtained the verbalization patterns in Chapter 7). For this investigation, we also use chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda to ultimately investigate the feasibility of having an API to verbalize ontologies in agglutinating Bantu languages. We had the following questions:

RQ4 (b.1). Is the combined morphology with syntax and semantics approach used for Runyankore and isiZulu also needed to pluralize nouns in other agglutinating Bantu languages?

RQ4 (b.2). Are the same classes of pluralization exceptions, found in Runyankore and isiZulu, also present in other agglutinating Bantu languages, and if so, are the Runyankore and isiZulu solutions applicable to corresponding exceptions in other languages?

RQ4 (b.3). In attempting to develop a generic pluralizer, how can the language-specific pluralizers be generalized whilst maintaining roughly the same accuracy?

RQ4 (b.4) Are Context-Free Grammars (CFGs), used for verb conjugation in Runyankore and isiZulu, also sufficient for conjugating verbs in other agglutinating Bantu languages, and if so, is the use of a generic CFG possible?

RQ4 (b.5). Given the processes and resources that are language-specific, can enough areas for generalizability be identified to support the development of an API to generate text in agglutinating Bantu languages?

In Section 8.2, we present the results from the investigation into the development of a generic pluralizer, including details on handling exceptions. Section 8.3 presents the analyses done on the verbal morphologies of chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda in order to identify whether their verbs can be conjugated using CFGs. Finally, the architecture that describes how an API can be developed is presented in Section 8.4. The main contributions in this chapter are:

- A generic pluralizer for agglutinating Bantu languages;
- More evidence to support the use of CFGs to conjugate verbs in agglutinating Bantu languages; and

- An architecture to show how an API can be built to verbalize ontologies in agglutinating Bantu languages.

Some of the work on the generic noun pluralizer in this chapter was published at the 27th International Conference on Computational Linguistics (COLING 2018) [28]. We however provide more analyses in Section 8.5.1 on how our work generalizes to other agglutinating Bantu languages not used in this investigation.

8.2 Generic Pluralizer

In Section 3.2 in Chapter 3, we explained the limitations of the existing noun pluralization approaches, which resulted in the use of an approach that combines morphology with syntax and semantics to pluralize nouns in Runyankore and for isiZulu [29]. We used the same approach to develop the generic pluralizer because: (1) it achieved high accuracy with two Bantu languages, Runyankore and isiZulu [29]; and (2) a deterministic outcome is now possible due to the noun class regrouping in Table 7.3 in Section 7.4 in Chapter 7. Having a deterministic outcome is important for generalizability because it ensures a one-to-one singular/plural mapping.

8.2.1 Design Decisions

We first investigated how the language-specific rules in the algorithms based on the standard singular/plural pairings can be generalized. We later extended this to handle exceptions.

Standard Noun Class Pluralization

We have seen that the NC determines pluralization, and that the singular/plural pairings are language-specific. However, having ensured a one-to-one mapping in the regrouped noun classes in Table 7.3 in Section 7.4 in Chapter 7, we found that, if the NC is stored as an external resource, it can be loaded into the pluralizer as variables, and a generic pluralization algorithm can then be applied. This keeps the NC details language-specific but makes the rules for pluralization generic. Algorithm 8.2.1 shows the generic pluralization rules:

Algorithm 8.2.1 Generic pluralization rules based on variables

1: Variables: N, noun to be pluralized; C_n, the NC of the noun; C, the NC in the NC system; S_p, the singular prefix; P_p, the plural prefix; and L_s, the length of the singular prefix
2: **if** $C_n == C$ **then**
3: $L_s \leftarrow getLength(S_p)$ ▷ Get the number of characters in the singular prefix
4: Result $\leftarrow$ "$P_p + N.substring(L_s)$" ▷ Pluralize by replacing the singular prefix with the plural prefix
5: **end if**
6: **return** Result

The rules in Algorithm 8.2.1 are written once, and the language specific noun class pairings are loaded as variables for each language during pluralization. This can, for example, pluralize nouns like ***ei**shomero* and ***eri**isho* in Runyankore, which are in NC 5 and NC 5i respectively, according to Table 7.3. However, before the noun class regrouping (both belonged to NC 5) pluralization required the hard-coding of the rules from line 6 in Algorithm 3.3.1, which cannot be generalized.

Handling Exceptions

Recall that in Chapter 3, we defined 'exceptions' as those nouns whose pluralization requires different rules or additional rules to the basic ones in the singular/plural noun class (NC) pairings.

We have seen in Section 3.4 in Chapter 3 that there are several types of exceptions: (1) mass nouns (whose noun class is marked with an '**m**' for identification); (2) prefix exceptions (whose correct plural is made directly available to the pluralizer); (3) singular-only nouns (placed in a separate look-up file for identification); and (4) compound nouns and noun phrases (require further processing with concords).

We found that the same exceptions occur during pluralization in other agglutinating Bantu languages, and that the same solutions used for Runyankore and isiZulu are also applicable to other agglutinating Bantu languages. However, instead of hard-coding the rules for prefix exceptions as we did for Runyankore, we placed them in a look-up file (this was done for isiZulu in [29]). Algorithm 8.2.2 shows the generic rules to handle exceptions.

Algorithm 8.2.2 Pluralizing mass nouns, singular-only nouns, compound nouns, and prefix exceptions in the generic pluralizer

1: Variables: N, noun to be pluralized; C_n, the NC of the noun; N', the prefix exception plural; N_1, the pluralized noun; C_p, the plural NC; P_c, the possessive concord
2: **if** $SingularOnlyFile.contains(N)$ **then**
3: Result $\leftarrow$ "N" ▷ Return noun unchanged if it is found in the 'singular-only' file
4: **else if** $ExceptionsFile.contains(N)$ **then**
5: $N' \leftarrow getExceptionPlural(N)$ ▷ Get the prefix exception plural from the 'exceptions' file
6: Result $\leftarrow$ "N'" ▷ Pluralize with the prefix exception
7: **else if** $C_n.contains(\text{'m'})$ **then**
8: Result $\leftarrow$ "N" ▷ Return original noun if NC is marked with an '**m**'
9: **else if** $N.isCompoundNoun()$ **then**
10: $a[] \leftarrow split(N)$ ▷ Split the compound noun into: main noun, possessive concord, and modifier noun
11: $N_1 \leftarrow pluralize(A[0], C_n)$ ▷ Pluralize main noun according to Algorithm 8.2.1
12: $P_c \leftarrow getPossessiveConcord(C_n)$ ▷ Get the possessive concord of the pluralized noun
13: Result $\leftarrow$ "N_1 P_c $A[2]$" ▷ Pluralize with the plural main noun, plural possessive concord, and original modifier noun
14: **else**
15: $N_1 \leftarrow pluralize(A[0], C_n)$ ▷ Pluralize main noun according to Algorithm 8.2.1
16: **end if**
17: **return** Result

The handling of exceptions in Algorithm 8.2.2 is generic and is made language-specific by loading the resources containing the noun class system, singular-only nouns, and prefix exceptions for each language, and using them as variables as shown in lines 4, 5, and 6 in Algorithm 8.2.2. However, our algorithm is limited to compound nouns that require the possessive concord for pluralization, and no phonological conditioning is performed here.

Architecture of Generic Pluralizer

The design of the generic pluralizer needs to achieve the following:

(1) Provide a singular noun and its NC, so as to obtain its plural prefix;

(2) Ensure that mass nouns and singular-only nouns are not pluralized;

(3) Obtain the plural of the main noun and its plural possessive concord during the pluralization of compound nouns; and

(4) Correctly pluralize nouns whose pluralization deviates from the NC pairings.

The language-specific NC details are made available to the pluralizer through text files, one for each language, which are read into the program, and made available to the pluralizer as variables, as shown in the architecture in Figure 8.1.

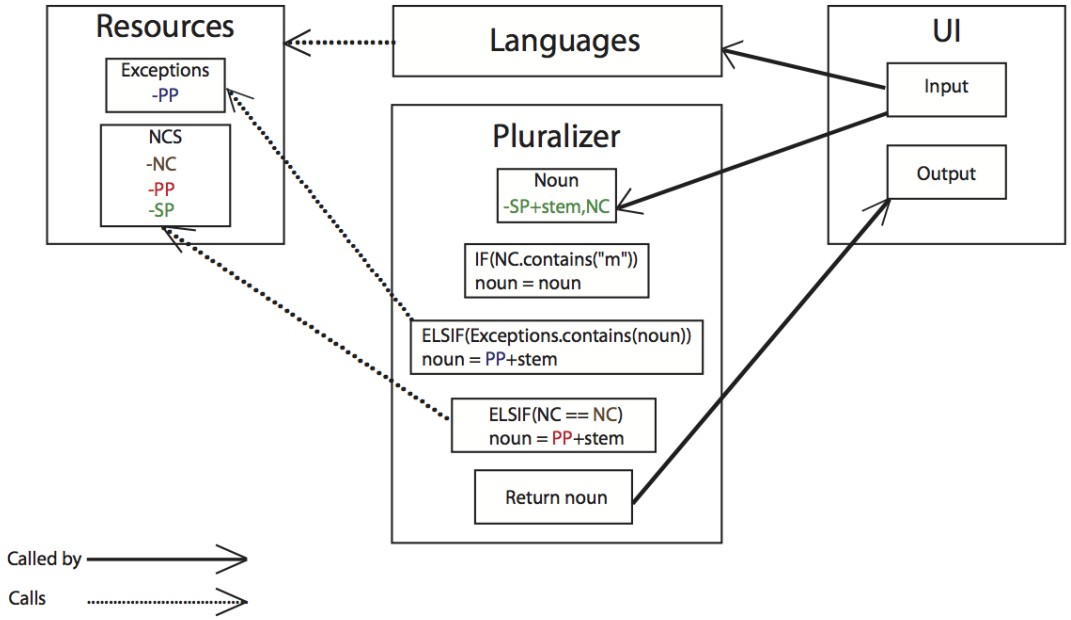

FIGURE 8.1: The architecture of the generic pluralizer: 'PP', plural prefix; 'SP', singular prefix; 'NC', noun class

Figure 8.1 shows how the selected language and noun(s) to be pluralized are entered through the **UI**. Through the **Languages** resources, the text files that contain the regrouped noun classes, the singular/plural pairings, as well as prefix exceptions are loaded into the **Pluralizer** as variables. The following then takes place (as shown in Algorithm 8.2.2):

(1) If the noun is found in the **Exceptions** look-up file, it is pluralized according to the exceptions prefix instead of the standard one;

(2) If the noun is a mass noun (with an '**m**' on the NC), then it is returned without pluralization; and

(3) If the noun's NC is the same as that in **NCS** (the noun class system), then it is pluralized accordingly.

We implemented the algorithms for the generic pluralizer as a Java application, and created a 'Resources' folder in which language-specific resources (text files for the noun class system, singular-only nouns, and exceptions) are contained[1].

8.2.2 Evaluation of Generic Pluralizer

We evaluated the generic pluralizer using seven agglutinating Bantu languages: chiShona, isiXhosa, isiZulu, Kikuyu, Kinyarwanda, Luganda, and Runyankore; and carried out two types of evaluation: verification and validation. Verification was used to ensure that all NCs were accounted for (based on what the literature states should be in each NC) and were correctly pluralized. Validation was aimed at comparing the accuracy of the generic pluralizer to that of the language-specific Runyankore and isiZulu pluralizers in [29].

[1]The generic pluralizer can be accessed from `https://github.com/runyankorenlg/Generic-Pluralizer`.

Materials and Methods

We used two test sets, one for verification (internal) testing, **SetI**, and another for validation (coverage) testing, **SetC**. Both test sets were compiled from English sources and then translated into each of the seven languages. We did this because most of the test languages are computationally under-resourced, and thus lack resources such as dictionaries from which singular nouns could otherwise have been extracted randomly.

SetI is a wordlist comprising 81 nouns that was compiled to reflect the contents of each NC according to the semantic generalizations presented in Table 2.1 in Section 2.3.1. Mass nouns, abstract concepts, and compound nouns were purposefully added to this test set. **SetC** is the same wordlist used as **Set1** in Section 3.6 to evaluate the Runyankore and isiZulu pluralizers [29]. It consists of 101 words informed by multiple ontologies [2].

These two test sets were translated from English to chiShona, isiXhosa, isiZulu, Kikuyu, Luganda, and Runyankore by native speakers. The noun class (NC) (according to the regrouping in Table 7.3) was then associated with each noun in each test set. For Runyankore and isiZulu, we updated the NCs in **SectC** according to the regrouping in Table 7.3. While not all nouns in the English wordlist were translatable into all languages, all NCs were accounted for in the resulting wordlists.

We used accuracy as the metric for evaluation, which is determined as the percentage of correct plurals over the test set. Additionally, for compound nouns, the evaluation tested for the ability to correctly pluralize the main noun and then use it to obtain the correct plural possessive concord.

We have stated that phonological conditioning cannot be performed by the generic pluralizer, due to the rules largely being language-specific. We thus evaluated the correctness of compound nouns based on: correctly pluralizing the main noun, and obtaining the correct plural possessive concord.

Results and Analysis

Table 8.1 shows the accuracy of the generic pluralizer for each language and for both test sets, except for **SetC** for Kinyarwanda where we were unable to get the translations for this dataset. The accuracy is based on the total number of translatable nouns.

TABLE 8.1: Accuracy of the generic pluralizer.

	SetI		SetC	
Language	**Translatable Nouns**	**Accuracy**	**Translatable Nouns**	**Accuracy**
chiShona	68	94.12%	74	94.59%
isiXhosa	74	97.30%	83	100%
isiZulu	71	100%	101	97.03%
Kikuyu	77	96.10%	76	94.74%
Kinyarwanda	70	97.14%	-	-
Luganda	75	93.33%	78	97.44%
Runyankore	81	93.83%	88	96.59%

From Table 8.1, it can be seen that the lowest accuracy is 93%, and up to 100% is achieved for isiZulu for **SetI**. With the exception of **SetC** for isiZulu, all the incorrect plurals for the other languages and both test sets are due to the lack of phonological conditioning. In chiShona, for example, *gumbo* 'leg', is pluralized as *makumbo*, instead of the generated *magumbo*.

The lower accuracy in **SetC** for isiZulu is due to the generic pluralizer only handling compound nouns with the possessive concord. The isiZulu pluralizer in [29] handles compound nouns with adjectives using the adjective concord[3].

[2] TheEnglish 101 wordlist can be found in Appendix G.

[3] The detailed accuracy statistics for all languages can be accessed at https://github.com/runyankorenlg/Generic-Pluralizer.

Prefix Exceptions We identified nouns in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda which deviate from the singular/plural rules. We placed these exceptions in a separate look-up file with their corresponding plurals, because there are no rules among these true exceptions. The file was added to the resources of each language (as shown in Figure 8.1). They are then pluralized according to lines 4, 5, and 6 in Algorithm 8.2.2. Table 8.2 shows an example of the prefix exceptions identified for each language.

TABLE 8.2: Examples of prefix exceptions identified for each language

Language	Singular	NC-based Plural	Correct Plural
chiShona	***i**mbwa* (dog)	***dzi**mbwa*	***i**mbwa*
isiXhosa	***u**makhulu* (grandmother)	***oo**makhulu*	*imithi*
Kikuyu	*dagetari* (doctor)	*dagetari*	***ma**dagetari*
Kinyarwanda	***in**zu* (house)	***in**zu*	***ama**zu*
Luganda	***aka**kelenda* (pill)	***otu**kelenda*	***ama**kelenda*

We also observed that the Prefix exceptions appear to be rare in each language.

Singular-only Nouns We found that singular-only nouns in the five additional languages are also nouns for abstract concepts (such as 'greed' and 'thirst') or infinitive nouns (in NCs 15 and 15i). We also placed these nouns in a separate look-up file, and lines 2 and 3 in Algorithm 8.2.2 ensure that they are not pluralized. Table 8.3 shows some of the singular-only nouns identified in the test sets.

TABLE 8.3: Examples of singular-only nouns identified in each language

Language	Singular-only Nouns Identified
chiShona	*moto* (fire); *zuva* (sun); *kunaka* (beauty); *hushamwari* (friendship)
isiXhosa	*ubuhle* (beauty); *ubuhlobo* (friendship); *ukulala* (sleep); *ukunxanwa* (thirst); *indlala* (hunger)
Kikuyu	*toro* (sleep); *ukoroku* (greed); *wedo* (love); *nyota* (thirst); *ngaragu* (hunger)
Kinyarwanda	*umuriro* (fire); *izuba* (sun); *umucyo* (daylight); *urukundo* (love); *kwangana* (hate)
Luganda	*omululu* (greed); *enyoonta* (thirst); *enjala* (hunger); *okukyawa* (hate); *omukwano* (friendship)

Abstract nouns (such as 'beauty') can be found in either singular or plural NCs. All nouns in plural NCs are returned without modification.

Mass Nouns We applied the same identification strategy used for Runyankore and isiZulu [29] to flag mass nouns in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda. This involves marking the NC with an '**m**' so that it is not pluralized (lines 7 and 8 in Algorithm 8.2.2). Examples of some singular mass nouns identified include: *iria* 'milk' in NC 5 in Kikuyu; and 'wine', which is *doro* in NC5 in chiShona and *umdiliya-omfaxangiweyo* in NC 3 in isiXhosa.

Mass nouns that belong to plural NCs, such as *mae* (NC6, Kikuyu), *mvura* (NC10ii, chiShona), and *amanzi* (NC6, isiXhosa), the translations for 'water', are left unmodified.

Compound Nouns Compound nouns in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda are also pluralized by first pluralizing the main noun, obtaining the plural possessive concord, and then associating it with the modifier noun (lines 10, 11, 12, and 13 in Algorithm 8.2.2).

We also found that, with the exception of Kikuyu, phonological conditioning is required for the pluralization of compound nouns for all these languages. However, as explained in Section 7.3, the rules for phonological conditioning are largely language-specific. We instead reverse the phonological conditioning in the singular inputs to the generic pluralizer, and assessed correctness based on whether the generic pluralizer is able to pluralize the main noun and obtain the correct plural possessive concord. Table 8.4 shows the results of pluralizing 'schoolteacher' in

each language, as well as the phonologically-reversed inputs and generated output. We include their phonologically conditioned versions for clarity.

TABLE 8.4: Pluralizing compound nouns in the five languages; example: 'schoolteacher' ('PC', Phonologically Conditioned)

Language	PC Singular	Non-PC Singular	PC Plural	Generated Plural
chiShona	*Mudzidzisi **we**chikoro*	*Mudzidzisi **wa** chikoro*	*vadzidzisi **ve**chikoro*	*vadzidzisi **va** chikoro*
isiXhosa	*utishala **we**sikolo*	*utishala **wa** isikolo*	*ootishala **be**sikolo*	*ootishala **ba** isikolo*
Kikuyu	-	*mwarimu **wa** cukuru*	-	*arimu **a** cukuru*
Kinyarwanda	*umurezi **w**ishuri*	*umurezi **wa** ishuri*	*abarizi **b**ishuri*	*abarizi **ba** ishuri*
Luganda	*omusomesa **w'**esomelo*	*omusomesa **wa** esomelo*	*abasomesa **b'**esomelo*	*abasomesa **ba** esomelo*

This evaluation completes our investigation into whether it is possible to develop a generic pluralizer. Next, we investigate whether Context-Free Grammars (CFGs) are adequate for verb conjugation in other agglutinating Bantu languages.

8.3 Verb Conjugation with Context-Free Grammars

In Runyankore [25] and isiZulu [100], verb conjugation was achieved through the use of Context-Free Grammars (CFGs). When investigating whether CFGs could be similarly applied to other agglutinating Bantu languages, we started by researching their verbal morphologies. We used chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda in this investigation. We have seen (in Section 2.3.3) that the verbal morphology of Bantu languages is very complex. We therefore limited the scope of this analysis to the simple present tense and negation, which offer a starting point from which further analyses can be extended.

We studied the literature on the verbal morphologies of these languages, and found the following [60, 88, 89, 101, 121, 128, 140]:

(1) The subject concord, verb-root, and final are required verbal slots in all languages;

(2) Other verbal slots (such as negation, tense and aspect, object concords, and extensions) are subject to the context;

(3) Each verbal slot holds one or more morphemes; and

(4) As shown in Table 7.1 in Section 7.3 in Chapter 7, the arrangement of the verbal slots varies from language to language.

Runyankore and isiZulu, whose verbs can be conjugated using CFGs, also share the verbal structure documented about the five additional languages. We therefore hypothesized that CFGs can also be applied to conjugate verbs in these languages, with the verbal slots making up the non-terminals in the CFGs. In the next sections, we provide details of the verbal morphologies of each of the five languages, and the analyses performed to determine that CFGs can be used to conjugate verbs in the simple present tense and negation. Though we use verbs from the healthcare domain as examples, the conjugation process is the same for verbs in other domains.

8.3.1 chiShona

chiShona has fourteen verbal slots, though only the five shown in Table 8.5 are needed for the simple present tense and negation.

The negation verbal slot is placed before the subject concord, and does not coexist with the aspect marker [128]. Additionally, the final vowel *a* is indicative and used when the simple present tense is applied, while *e* is used during negation [128]. We found that this verbal structure can be

TABLE 8.5: The verbal slots and morphemes for the simple present tense and negation in chiShona

Verbal Slot	**Morpheme**
Negation	*ha*
Subject	See subject concords shown in Table F.1 in Appendix F
Continuous Aspect Marker	*no*
Verb-Root	*VR*
Final Vowel	*a* | *e*

represented with a CFG, with the non-terminals NG, SC, AS, VR, FV used to represent the verbal slots for negation, subject concords, aspect, verb-root, and final vowel respectively. The CFG is shown below (VR, the verb-roots 'eat', 'cure', and 'reduce' as examples).

$S \rightarrow NG\ SC\ AS\ VS\ FV$
$NG \rightarrow \emptyset$ | ha
$SC \rightarrow$ a | va | u | i | ri | a | chi | zvi | dzi | ru | ka | tu | hu | ku
$AS \rightarrow \emptyset$ | no
$VR \rightarrow$ dy | rap | deredz
$FV \rightarrow$ a | e

The examples below show the derivations for the simple present tense and negation:

Simple Present Tense: Every medicine reduces only pain.

chiShona: *Mushonga wese* ***unoderedza*** *marwadzo chete*
Derivation: $S \rightarrow NG\ SC\ AS\ VR\ FV \rightarrow SC\ AS\ VR\ FV \rightarrow u\ AS\ VR\ FV \rightarrow uno\ VR\ FV \rightarrow unoderedz\ FV \rightarrow unoderedza$

Negation: All medicine does not only cure fever.

chiShona: *Mishonga yese* ***hairape*** *gwirikwiti chete*.
Derivation: $S \rightarrow NG\ SC\ AS\ VR\ FV \rightarrow ha\ SC\ AS\ VR\ FV \rightarrow hai\ AS\ VR\ FV \rightarrow hai\ VR\ FV \rightarrow hairap\ FV \rightarrow hairape$

8.3.2 isiXhosa

isiXhosa has three verbal slots for conjugating verbs in the indicative simple present tense, but four with negation [89]. These are shown in Table 8.6 along with their associated morphemes.

TABLE 8.6: The verbal slots and morphemes for the simple present tense and negation in isiXhosa

Verbal Slot	**Morpheme**
Negative Affix	*a*
Subject	See the subject concords in Table F.2 in Appendix F
Negation	Negative subject concord obtained through the NC
Verb-Root	*VR*
Final Vowel	*a* | *i*

The negation and subject verbal slots do not cooccur: during negation, only the negative subject concord is used, while the subject concord applies outside negation [89]. The NC-based negative subject concords are shown in Table 8.7.

In the indicative present tense, the negative affix *a* is used with the final vowel *i* [89]. However, the final vowel *i* is omitted if the verb does not end with *a* [89].

TABLE 8.7: The negative subject concords for each NC in isiXhosa ('1PS', first person singular; '1PP',, first person plural; '2PS', second person singular; '2PP', second person plural)

NC	Negative Subject Concord
1PS	*-ndi-*
1PP	*-si-*
2PS	*-wu-/-ku-*
2PP	*-m-*
1	*-ka-*
2	*-ba-*
3	*-wu-*
4	*-yl-*
5	*-Ii-*
6	*-wa-*
7	*-Si-*
8	*-Zi-*
9	*-yi-*
10	*-zi-*
11	*-Iu-*
14	*-bu-*
15	*-ku-*

This structure can be represented using a CFG with the non-terminals NA, IN, SC, NG, VR, FV for the negative affix, initial, subject concords, negative concords, verb-root, and final vowel respectively. The 'initial' verbal slot contains the production rule that results in either the subjunctive concords or the negative concords, as they cannot cooccur. The CFG is shown below (VR, the verb-roots 'reduce', 'read', and 'cure' as examples).

$S \rightarrow NA\ IN\ VR\ FV$
$NA \rightarrow \emptyset \mid$ a
$IN \rightarrow SC \mid NG$
$SC \rightarrow$ u | ba | i | li | a | si | zi | lu | bu | ku
$NG \rightarrow$ ndi | si | wu | ku | m | ka | ba | wu | yl | Ii | wa |
Si | Zi | yi | zi | Iu | bu | ku
$VR \rightarrow$ nciphis | dl | philis
$FV \rightarrow$ a | i

The derivations from this CFG are shown in the examples below:

Simple Present Tense: Every medication reduces only pain.

isiXhosa: *Iyeza nganye **lihlisa** intlungu zodwa.*
Derivation: $S \rightarrow NA\ IN\ VR\ FV \rightarrow IN\ VR\ FV \rightarrow SC\ VR\ FV \rightarrow li\ VR\ FV \rightarrow lihlis\ FV \rightarrow lihlisa$

Negation: All medicine does not only cure fever.

isiXhosa: *Onke amayenza **ayiphilisi** umkhuhlane wodwa.*
Derivation: $S \rightarrow NA\ IN\ VR\ FV \rightarrow a\ IN\ VR\ FV \rightarrow a\ NG\ VR\ FV \rightarrow ayi\ VR\ FV \rightarrow ayiphilis\ FV \rightarrow ayiphilisi$

8.3.3 Kikuyu

In Kikuyu, the simple present tense is usually presented with the habitual aspect, whose verbal slot appears between the verb-root and final vowel [88]. The other verbal slots for the simple present tense and negation are shown in Table 8.8, along with their respective morphemes.

TABLE 8.8: The verbal slots and morphemes for the simple present tense and negation in Kikuyu

Verbal Slot	**Morpheme**
Negation	*nd* | *ti*
Subject	See the subject concords in Table F.3 in Appendix F
Verb-Root	*VR*
Habitual Aspect Marker	*ag*
Final Vowel	*a*

The morpheme and its placement during negation depend on the nature of the subject concord: if the subject concord is a vowel only, then *nd* is used and it is placed before the subject concord; but in all other cases, *ti* is used and it is placed after the subject concord [88]. This verbal structure can also be represented with a CFG. However, in order to account for the special conditions around negation, we used two non-terminals to represent the different negation morphemes, especially because their placement in the verb changes. The CFG thus comprises the non-terminals VN, SC, CN, VR, AS, FV used to represent the verbal slots for negation with a vowel subject concord, subject concords, negation with a subject concord starting with a consonant, verb-root, aspect, and final vowel respectively. The CFG is shown below (verb-roots 'reduce', 'eat', and 'cure' as examples):

$S \rightarrow VN\ SC\ CN\ VS\ AS\ FV$
$VN \rightarrow \emptyset \mid \text{nd}$
$SC \rightarrow \text{a} \mid \text{ma} \mid \text{u} \mid \text{i} \mid \text{ri} \mid \text{ki} \mid \text{ci} \mid \text{ru} \mid \text{ka} \mid \text{tu} \mid \text{ku}$
$CN \rightarrow \emptyset \mid \text{ti}$
$VR \rightarrow \text{nyih} \mid \text{re} \mid \text{hon}$
$AS \rightarrow \emptyset \mid \text{ag}$
$FV \rightarrow \text{a}$

We show the derivations from the above CFG in the examples below:

Simple Present Tense: Every medication reduces only pain.

Kikuyu: *Oo dawa* ***enyihaga*** *ruo tu*.
Derivation: $S \rightarrow VN\ SC\ CN\ VR\ AS\ FV \rightarrow SC\ CN\ VR\ AS\ FV \rightarrow e\ CN\ VR\ AS\ FV \rightarrow e\ VR\ AS\ FV \rightarrow enyih\ AS\ FV \rightarrow enyihag\ FV \rightarrow enyihaga$

Negation (Vowel SC): Every medication does not only cure fever.

Kikuyu: *Oo dawa* ***ndehonaga*** *homa tu*.
Derivation: $S \rightarrow VN\ SC\ CN\ VR\ AS\ FV \rightarrow nd\ SC\ CN\ VR\ AS\ FV \rightarrow nde\ CN\ VR\ AS\ FV \rightarrow nde\ VR\ AS\ FV \rightarrow ndehon\ AS\ FV \rightarrow ndehonag\ FV \rightarrow ndehonaga$

Negation, Other: All medications do not only reduce pain.

Kikuyu: *Dawa ciothe* ***citinyihaga*** *ruo tu*.
Derivation: $S \rightarrow VN\ SC\ CN\ VR\ AS\ FV \rightarrow SC\ CN\ VR\ AS\ FV \rightarrow ci\ CN\ VR\ AS\ FV \rightarrow citi\ VR\ AS\ FV \rightarrow citinyih\ AS\ FV \rightarrow citinyihag\ FV \rightarrow citinyihaga$

8.3.4 Kinyarwanda

Kinyarwanda uses four verbal slots for the simple present tense and negation [101]. These are shown in Table 8.9, together with their respective morphemes.

In the simple present tense, the final vowel is also considered to be the aspect marker [101]. The negation morpheme is found in the pre-prefix slot (placed before the subject), which is either the negation morpheme *nti* or the temporal morpheme *ni* [101]. This verbal structure is derivable using a CFG, with the non-terminals PP, SC, VR, FV used to represent the grammatical slots for

TABLE 8.9: The verbal slots and morphemes for the simple present tense and negation in Kinyarwanda

Verbal Slot	**Morpheme**
Pre-prefix	*nti* | *ni*
Subject	See the subject concords in Table F.4 in Appendix F
Verb-Root	*VR*
Final Vowel	*a* | *e*

the pre-prefix, subject concords, verb-root, and final vowel respectively. The CFG is shown below (verb-roots 'reduce', 'eat', and 'cure' as examples).

$S \rightarrow PP\ SC\ VS\ FV$
$PP \rightarrow \emptyset$ | nti | ni
$SC \rightarrow$ a | o | n | ba | u | i | ri | a | ki | bi | i |
zi | ru | ka | tu | bu | ku | ha
$VR \rightarrow$ gabany | ry | kijij
$FV \rightarrow$ a | e | ire

The derivations from the above CFG are shown in the examples below:

Simple Present Tense: Every medication reduces only pain.

Kinyarwanda: *Buri muti* ***ugabanya*** *gusa uburibwe.*
Derivation: $S \rightarrow PP\ SC\ VR\ FV \rightarrow\ SC\ VR\ FV \rightarrow u\ VR\ FV \rightarrow ugabany\ FV \rightarrow ugabanya$

Negation: All medicine does not only cure fever.

Kinyarwanda: *Imiti yose* ***ntikijija*** *gusa ubuganga.*
Derivation: $S \rightarrow PP\ SC\ VR\ FV \rightarrow nti\ SC\ VR\ FV \rightarrow nti\ VR\ FV \rightarrow ntikijij\ FV \rightarrow ntikijija$

8.3.5 Luganda

Luganda also requires four verbal slots to conjugate verbs in the simple present tense and negation. These, with their associated morphemes, are shown in Table 8.10.

TABLE 8.10: The verbal slots and morphemes for the simple present tense and negation in Luganda

Verbal Slot	**Morpheme**
Negation	*te* | *si*
Subject	See the subject concords in Table F.5 in Appendix F
Verb-Root	*VR*
Final Vowel	*a* | *e*

The morpheme *te* is always used for negation, except in the first person singular, when *si* is used instead [60]. The final vowel is also sometimes referred to as the aspect marker, and it can be either the indicative *a* or the subjunctive *e* [60]. A CFG can also be used to conjugate this structure, with the non-terminals NG, SC, VR, FV used to represent the verbal slots for the negation, subject concords, verb-root, and final vowel respectively. The CFG is shown below (verb-roots 'reduce', 'eat',, and 'cure').

$S \rightarrow NG\ SC\ VS\ FV$
$NG \rightarrow \emptyset$ | te | si
$SC \rightarrow$ a | o | n | tu | mu | ba | gu | gi | li | ga | ki |
bi | e | zi | lu | tu | ka | bu | ku | gu | ga
$VR \rightarrow$ kendez | ly | janjab
$FV \rightarrow$ a | e | ire

The derivations from this CFG are shown in the examples below:

Simple Present Tense: Every medicine reduces only pain.

Luganda: *Buli mubazi* ***gukendeza*** *obusassi bwokka.*
Derivation: $S \rightarrow NG\ SC\ VR\ FV \rightarrow SC\ VR\ FV \rightarrow gu\ VR\ FV \rightarrow gukendez\ FV \rightarrow gukendeza$

Negation: All medication does not only cure fever.

Luganda: *Emibazi yonna* ***tegijajamba*** *omusujja gwokka.*
Derivation: $S \rightarrow NG\ SC\ VR\ FV \rightarrow te\ SC\ VR\ FV \rightarrow tegi\ VR\ FV \rightarrow tegijanjab\ FV \rightarrow tegijanjaba$

Though the above results show that CFGs can also be used to conjugate verbs in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda in the simple present tense and negation, it can also be seen that separate CFGs are required for each language, because of their different verbal morphologies, and the use of a generic CFG for verb conjugation is not possible.

We next investigate whether the generic pluralizer in Section 8.2, CFGs in Section 8.3, and verbalization patterns in Section 7.5 can be integrated into a single API to verbalize ontologies in agglutinating Bantu languages.

8.4 Architecture of API

When investigating how the bootstrapped verbalizations, generic pluralizer, and CFGs can be combined into an API to generate text in agglutinating Bantu languages, we identified the following requirements:

(1) An annotation model is necessary to associate a concept with its translation, part of speech, and NC;

(2) There are language-specific processes, like verb conjugation and phonological conditioning, that are required for text generation, but are language-specific;

(3) There are language-specific resources, such as the noun class (NC) system, which are required across several modules; for example, pluralization, verb conjugation, and verbalization all require the NC;

(4) There are algorithms, such as the generic pluralizer, that can be generalized; and

(5) There are features whose role is only to provide access, but are not involved in the verbalization process, but are still nonetheless necessary.

8.4.1 Key Processes

Similar to Runyankore and isiZulu, there are three key processes required to generate text in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda: noun pluralization, verb conjugation, and verbalization.

We developed a generic pluralizer in Section 8.2 that can handle the same exceptions as the Runyankore and isiZulu pluralizers, and it achieved high levels of accuracy. This pluralizer can be integrated into an API in the same manner as that for Runyankore in Figure 6.1, however, there is a need to make available the language-specific resources shown in Figure 8.1.

We have shown that verbs in these languages can be conjugated using CFGs. However, due to their different verbal morphologies, a generic CFG cannot be applied to verbs in agglutinating Bantu languages. This is true regardless of whether the languages being compared are in the same zone or different zones (shown in Section 8.3; also see Table 7.1). Considering that our

analysis in Section 8.3 is limited to the simple present tense and negation, the verbal morphology becomes more complex as more tenses, aspects, moods, and extensions are considered. In an API, therefore, verb conjugation has to be handled at the language-specific scope. Additionally, a means of accessing the noun class is necessary for verb conjugation.

Verbalization is also largely language-specific, as shown in Section 7.5. With the exception of conjunction, all other constructors considered (in at least two cases for each language) require the NC during verbalization. This also places the verbalization patterns at the language-specific scope, requiring access to the NC.

Phonological conditioning is another key process. It is necessary to correct the errors that arise from agglutination. Runyankore requires two separate modules: one for nouns and verbalizations, and another for verbs (see Figure 6.1). The need for phonological conditioning in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda during noun pluralization (see Section 8.2) and verbalization (see Section 7.5) has been explained. During verb conjugation, for example negation in Kinyarwanda and Luganda, phonological conditioning is also required. Similar to Runyankore, the negation morphemes of Kinyarwanda (*nti*) and Luganda (*te*) undergo vowel elision if the subject concord is a vowel only. In isiXhosa, phonological conditioning was found to affect at least 45% of conjugated verbs; it is as high as 67% in isiZulu [121]. Additionally, rules for phonological conditioning are largely language-specific, and an API would therefore include a phonological conditioning module for each language.

8.4.2 Complementary Process

We consider bootstrapping verbalizations as a complementary process to reduce the time and effort required to add a new language to an API. A bootstrap module, which makes use of the bootstrappability metric developed in Section 7.6, would allow the selecting of one or more source language patterns, tailoring them to the target language, and saving the result from the most efficient bootstrap pair. We consider this module as a generic process, which selects from the available source languages.

8.4.3 Resources

The most important resource that the API must possess is the NC system for each language. As we have explained, the NC is required for noun pluralization, verb conjugation, and verbalization.

Annotations are another important resource. We explained in Section 2.4.2 that when verbalizing ontologies, it is necessary to associate linguistic information with ontology elements [23, 35]. Based on this, an API needs to also provide a means through which the concepts and roles can be annotated with linguistic information such as the translation, part of speech, and NC. We used an XML-based annotation model for this purpose for Runyankore in Section 6.2 in Chapter 6. While the same annotation model can be used for different languages, the resulting annotations will be language-specific.

Other resources required for pluralization are a list of singular-only nouns and prefix exceptions for each language (see Figure 8.1). An API also needs to include them.

8.4.4 Architecture

Based on the above requirements, and keeping in mind the need to foster generalizability whilst allowing for language specificity, we developed the architecture for an API, shown in Figure 8.2.

RESOURCES contains the rules and lexica that are language-specific. The former include the Context-Free Grammars for verb conjugation (**CFG**), the verbalization patterns (**Patterns**), and the phonological conditioning module (**PC**). The lexicon includes the noun class system (**NCS**), singular-only nouns (**SON**), prefix exceptions (**EXP**), and the annotations (**Annotations**). The **CFG** and **Patterns** both access **NCS** to obtain the appropriate concords.

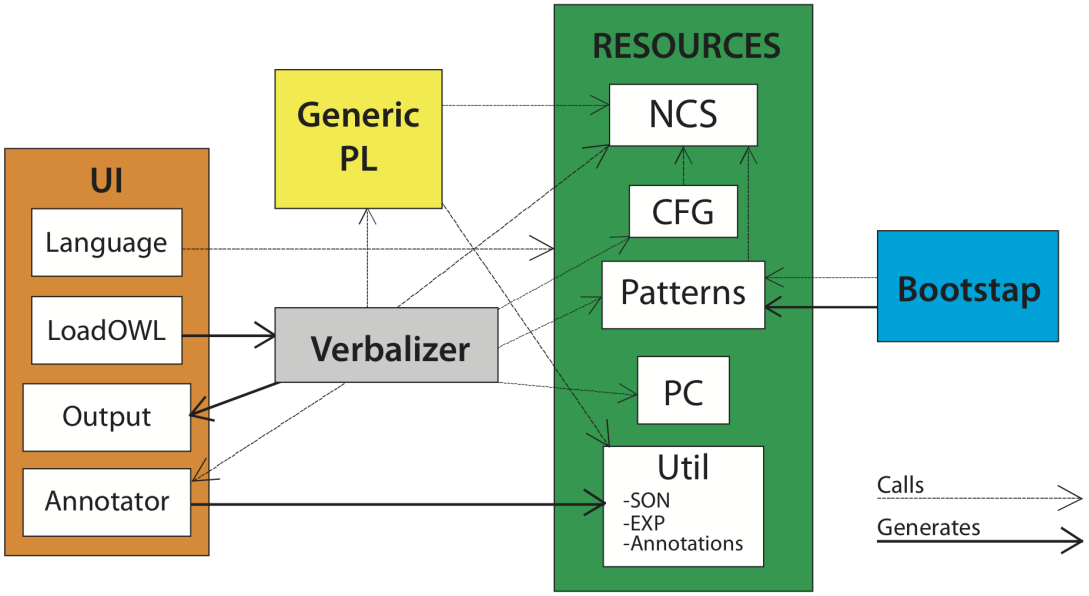

FIGURE 8.2: The proposed architecture of an API, showing the interaction among **RESOURCES**, **UI**, **GenericPL**, **Bootstrap**, and **Verbalizer**

The generic pluralizer (**GenericPL**) is not language-specific, but it accesses the language-specific **NCS** and **Util** in order to function. We showed the details of this in Figure 8.1.

The bootstrap module (**Bootstrap**) is also generic, but it uses existing language-specific patterns, generates the target-language pattern, and then saves it to the resources of the target language.

Verbalization is performed through **Verbalizer**, and this accesses **GenericPL** as well as the language-specific noun class system (**NCS**), verbalization patterns (**Patterns**), Context-Free Grammar (**CFG**), phonological conditioning module (**PC**), and the annotations with the translation, part of speech, and NC (**Annotations**).

The user interface (**UI**) is used to select a particular language (**Language**), load the OWL file to be verbalized (**LoadOWL**), annotate the concepts and roles in the OWL file (**Annotator**), and receive the text generated after verbalization (**Output**).

Once a particular language has been selected, its **RESOURCES** are loaded. The axioms in the loaded OWL file are passed to **Verbalizer** for verbalization, and the annotations are saved in **Util** as a resource. The generated text (**Output**) is passed back to **UI**.

The architecture in Figure 8.2 thus addresses all the requirements identified; that is, it is able to separate generic and language-specific components, thereby fostering generalizability while respecting language-specific features.

8.5 Discussion

In this investigation into developing an API to generate text in agglutinating Bantu languages, we identified the generic and language-specific components, and then used the architecture in Figure 8.2 to show how they can interact to generate text. The results obtained in sections 7.5, 8.2, and 8.3 show that using the bootstrap approach does reduce development time, as the solutions to verbalization, noun pluralization, and verb conjugation for five more languages were identified faster than they were for Runyankore and isiZulu. This was mainly because the requirements were already known from the research into Runyankore and isiZulu, and these were then investigated

for the other languages. The results on the evaluation of the generic pluralizer presented in Section 8.2.2 provide more proof that using the bootstrap approach does result in high levels of accuracy.

8.5.1 Generic Pluralization

When investigating noun pluralization for agglutinating Bantu languages, we had three questions based on the work in Runyankore and isiZulu: RQ4 (b.1), whether the same approach can be applied; RQ4 (b.2), whether the same exceptions are present; and RQ4 (b.3), whether a similar level of accuracy can be achieved.

We found that, due to the presence of the same class prefixes across different NCs, a combined morphology with syntax and semantics approach was also necessary to pluralize nouns in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda. Additionally, by purposively including the kinds of exceptions identified for Runyankore and isiZulu in the test sets, we showed that the same exceptions are present in these five languages. Further, through the regrouping of the classes in Table 7.3, we were able to develop generic rules for pluralization (as shown in algorithms 8.2.1 and 8.2.2). These algorithms make use of language-specific resources (as shown in Figure 8.1). Finally, the evaluation of the generic pluralizer achieved accuracy levels comparable to the language-specific pluralizers in [29].

Noun pluralization is still a recognized problem area in Bantu language computational linguistics, and generalizing existing approaches to new languages is further limited by the observed inconsistencies with determining the grammatical number of a noun based on the class prefixes [122]. The exceptions to pluralization identified for Runyankore, isiZulu, and the five additional languages, have also been identified in other Bantu languages. Mass nouns, for example, are not always treated as such in all languages, and can be found in singular NCs; for example, 'milk' in Bobe, Ifumu, Tshivenda, Kamba, and Emakhuwa; 'oil' in Bushong and Thimbukushu; and 'water' in Kikongo [122]. Maho [122] presents a case of Lingala, where an analysis of nouns in NCs 5 and 6[4] led to the proposal of differentiating between individuated (singular count nouns) and non-individuated (plural count nouns and mass nouns), because non-individuated nouns are unmarked in relation to individuated nouns. We used a 'marking system' for mass nouns with an '**m**' that serves to differentiate between the two.

Further, singular-only nouns are another exception found across Bantu languages and, as shown in the results of the generic pluralizer, they traditionally lack plural forms [122]. These arise from abstract nouns that are found in singular NCs; for example, ***u**nene* 'bigness' in NC 3 in Umbundu, ***li**bunga* 'forgetfulness' and ***li**memia* 'respect' in NC 5 in Lingala, and 'truth' in NC 9 in Duala, Tunen, Ifumu, Kinyamwezi, Kiswahili, and Tshivenda [122].

Further still, prefix exceptions are not limited to the seven languages we have presented. In Lingala, as in many Bantu languages, there are singular count nouns in NC 14 (typically a class of singular nouns with no plurals, such as abstract nouns); for example, ***bw**oto* 'canoe' is pluralized as ***m**ato* in NC 6 [122]. Maho [122] admits that such nouns may be regarded as exceptional, which is how we handled any nouns that deviated from the standard NC pairings.

We identified the need to handle phonological conditioning in the generic pluralizer from the incorrect plurals it generated. This need for phonological conditioning during noun pluralization has also been found in other Bantu languages, sometimes for NCs 3 and 4, but commonly for NC 9, due to the presence of a homorganic nasal, that is, a nasal that is phonologically adapted to an immediately following consonant [122]. However, as explained in sections 7.3 and 8.2.2, phonological conditioning is largely language-specific.

In Runyankore, we found that prefix exceptions are rare (see Section 3.4.2). The same is true for isiZulu [29] and the five additional languages, and is reported to be so in other Bantu languages

[4]For the semantic categorization of nouns in NC 5 and NC 6, see Table 2.1 in Section 2.3.1 in Chapter 2.

[122]. However, we found a larger number of prefix exceptions in Kikuyu than in the other languages, as shown in Table 8.11 (for Runyankore and isiZulu, no exceptions were found in these test sets; however, six were identified for Runyankore and two for isiZulu in [29]).

TABLE 8.11: Number of prefix exceptions found in both test sets for each language

Language	**Number of Prefix Exceptions**
chiShona	9
isiXhosa	2
isiZulu	0
Kikuyu	14
Kinyarwanda	5
Luganda	2
Runyankore	0

The higher number of exceptions in Kikuyu is due to it having several nouns whose pluralization deviates from the standard rules. Jeon et al. [88] state that the singular/plural pairings of 9/6, 11/6, 12/6 are rare, while 7/6, 1/8, 14/10 are extremely rare. Consequently, the number of observed prefix exceptions is larger than that of the other languages.

Another oddity first observed with the Runyankore pluralizer was that, instead of the pairing 12/13 as stated in the standard NC pairings, the correct pluralization is in fact 12/14. We did not only verify this with a Runyankore speaker, but also with a Runyankore dictionary [176], where no NC 13 nouns are found as plurals to NC 12. We found the same result in Luganda, but not in the others. According to the analysis by Maho [122], pairing 12/14 does occur in some languages, though the most common NC 12 plural is NC 13.

Finally, the processing of the resources required to add a new language to the generic pluralizer is quite considerable, with text files required for the noun class system (NC number, class prefixes, and possessive and adjective concords), noun class pairings, and for singular-only nouns and prefix exceptions, if known. This amount of effort is however still considerably less than that required by other multilingual noun pluralization implementations, notably, GF [155] and SimpleNLG [64]. The GF approach uses distinct functions for singulars and plurals, and linearization rules to handle the language-specific inflectional forms of nouns during pluralization [154, 155], and this requires the definition of a large resource grammar for each language [154]. For SimpleNLG, where pluralization is achieved through the language-specific lexicon saved as an XML file [64], both the singular and plural forms should be stated for each word in the lexicon [64, 135].

8.5.2 Verb Conjugation with CFGs

When answering RQ4 (b.4), the results from our research on literature on the verbal morphologies of chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda showed that Context-Free Grammars (CFGs) can also be used to conjugate verbs in these languages. However, even with the limited scope of the simple present tense and negation, we found differences in their verbal morphologies that make the use of a generic CFG for agglutinating Bantu languages not possible.

The limitations of using a generic CFG are still true for languages in the same zone. For example, negation in Luganda (see Table 8.10 in Section 8.3.5) uses two morphemes dependent on the nature of the subject concord, while Runyankore uses a single morpheme. Additionally, investigation by [121] into the grammatical similarity between isiXhosa and isiZulu revealed that, for the past, present, and future tenses and for the inductive, subjunctive, and participial moods, their verbs are at most 59.5% similar. Though the differences between them were found to be minor, this study concluded that the use of a singular merged set of grammar rules to generate verbs in both languages would require significant maintenance, due to dependencies that exist in one language but not in the other [121].

On the other hand, the general verbal morphology of Bantu languages, which is typically arranged with morphemes preceding and proceeding the verb-root, should also be derivable from a CFG. The morphemes that precede the verb-root typically represent the person or noun class (NC) of the subject, and inflectional categories such as time, negation, or aspect [165]. The morphemes proceeding the verb-root are typically the extensions (causative, applicative, stative, reciprocal, reversive, and passive) and the final (mood or final vowel) [145].

In Section 4.5, we found that phonological conditioning is required during verb conjugation. For isiZulu and isiXhosa, the need for phonological conditioning was identified in 67% and 45%, respectively, of verbs generated in the past, present, and future tenses and for the inductive, subjunctive, and participial moods [121]. We also found phonological conditioning necessary for Kinyarwanda and Luganda verbs, and further research is required to fully assess its rules in other agglutinating Bantu languages. Nonetheless, we can still conclude that the use of a generic phonological conditioning module for verb conjugation in all these languages is not possible.

8.5.3 Architecture for API

The key processes necessary for ontology verbalization were already known for Runyankore and isiZulu; and these are: noun pluralization, verb conjugation, and verbalization. Having bootstrapped the verbalizations, developed a generic pluralizer, and showed that CFGs can also be used for verb conjugation in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda, we proceeded to propose an architecture for an API to generate text in agglutinating Bantu languages in order to answer RQ4 (b.5).

The architecture in Figure 8.2 shows both the generic and language-specific processes and resources. The pluralizer, verbalizer, and bootstrap modules are generic; while the CFGs, verbalization patterns, and phonological conditioning modules are language-specific. There are also language-specific resources: the NC system, singular-only nouns, prefix exceptions, and annotations for each language.

While the architecture shown in Figure 8.2 shows that the interactions between the different processes and resources are quite complex, it nonetheless provides a means by which an API can be developed to verbalize ontologies in agglutinating Bantu languages. Further, the work presented in sections 7.5, 8.2, and 8.3 provides solutions to the generalizability and/or bootstrappability of these key processes, thus leaving their integration into an API as the only remaining issue. Figure 8.2 shows how the integration can be done.

8.6 Summary

In this chapter, we answered RQ4, 'How can the Runyankore and isiZulu approaches to ontology verbalization be generalized to other agglutinating Bantu languages' from Section 1.2.1 in Chapter 1. We investigated whether it is possible to develop an API for this purpose using five languages: chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda. We had five questions: RQ4 (b.1), RQ4 (b.2), and RQ4 (b.3) were on the development of a generic pluralizer; RQ4 (b.4) was on the use of CFGs for verb conjugation; and RQ4 (b.5) was on the API. In Section 8.2, we showed that it is indeed possible to develop a generic pluralizer based on a combined morphology with syntax and semantics approach, which comprises generic pluralization rules but language-specific resources. The generic pluralizer achieved high levels of accuracy. We also showed that the same exceptions to pluralization identified for Runyankore and isiZulu are found in the five additional languages, as well as in other Bantu languages. In Section 8.3, we showed that CFGs can also be used to conjugate verbs in the five additional languages, though the use of a generic CFG for agglutinating Bantu languages is not possible. Finally, In Section 8.4, we showed the architecture for an API to generate text in agglutinating Bantu languages, which showed the generic and language-specific processes and resources.

The next chapter presents conclusions and directions for future work.

Chapter 9

Conclusions

9.1 Overview

This research proposed an approach to verbalize ontologies in a hitherto largely unexplored family of languages, Bantu languages. Our research had two main objectives: (1) investigate how a grammar engine can be used to verbalize ontologies in Runyankore; and (2) investigate how the same solution can be generalized to other agglutinating Bantu languages. These four research questions were the basis of our investigations:

RQ1: How can noun pluralization and verb conjugation be achieved in Runyankore?

RQ2: What are the Runyankore verbalization patterns for the selected $\mathcal{ALCQ}$ constructors?

RQ3: How can the Runyankore verbalization patterns be implemented as algorithms in a grammar engine to verbalize ontologies?

RQ4: How can the Runyankore and isiZulu approaches to ontology verbalization be generalized to other agglutinating Bantu languages?

In this chapter, we present the research achievements in Section 9.2, and the contributions to this area of research in Section 9.3. The areas remaining for research in future are presented in Section 9.4, and final remarks in Section 9.5.

9.2 Research Achievements

We carried out our research in two phases: we investigated the approach for a single language, Runyankore, and then investigated whether that approach is generalizable to other agglutinating Bantu languages. In each phase, we investigated the processes of noun pluralization in Chapter 3, verb conjugation in Chapter 4, and verbalization in Chapter 5. Our domain of interest was healthcare, so we took a knowledge-to-text approach, because the healthcare field is well represented in ontologies, and used healthcare-related examples in this thesis.

9.2.1 RQ1: How can noun pluralization and verb conjugation be achieved in Runyankore?

We found that an approach that combines morphology with syntax and semantics is necessary to pluralize nouns in Runyankore. This is due to the presence of nouns that have the same class prefix but belong to different noun classes, and are therefore pluralized differently. The first version of the pluralizer in Section 3.3 was based on the standard singular/plural pairings, and we repeatedly tested it and added new rules as errors were identified. Through this approach, we identified six exceptional cases during noun pluralization in Runyankore and developed algorithms for them. These are: compound nouns in Section 3.4.1, prefix exceptions in Section 3.4.2,

singular-only nouns in Section 3.4.3, mass nouns in Section 3.4.4, noun phrases with verbs in Section 3.4.5, and noun phrases with adjectives in Section 3.4.5. Our pluralizer achieved 100% on two datasets and over 98% on a larger dataset.

For verb conjugation, we first selected the tense and aspect to use by analyzing a corpus of drug prescription explanations, where the simple present tense, progressive aspect, and passive were identified. The participial present continuous tense was later recommended by a linguist as more appropriate for written communication. In Section 4.4, we found that Context-Free Grammars [188] can handle the selected tense and aspect using seven non-terminals. We used separate CFGs for the auxiliary and copulative in Section 4.4.1 because they deviate from the standard. Evaluation with two linguists showed that the majority of the 99 generated verbs were regarded as grammatically correct, with most errors resulting from the need for phonological conditioning, which we addressed in Section 4.5.

9.2.2 RQ2: What are the Runyankore verbalization patterns for the selected $\mathcal{ALCQ}$ constructors?

In Section 5.2, we selected the description logic $\mathcal{ALCQ}$ as sufficient for generating text for the healthcare domain. Specifically, we selected eight constructors for verbalization: subsumption ($\sqsubseteq$), conjunction ($\sqcap$), negation ($\neg$), existential quantification ($\exists$), universal quantification ($\forall$), maximum cardinality ($\leq$), minimum cardinality ($\geq$), and exact cardinality ($=$). In Section 5.3, we found that verbalization is affected by six factors:

(1) The noun class of the concept's name;

(2) The concept's category;

(3) Whether the concept is atomic or an expression;

(4) The quantifier used in the axiom;

(5) The position of the concept in the axiom; and

(6) The number restriction in the axiom

We developed verbalization algorithms for the selected constructors. The evaluation carried out with non-linguists showed that the singular form was preferred for all constructors except for the negation of roles.

9.2.3 RQ3: How can the Runyankore verbalization patterns be implemented as algorithms in a grammar engine to verbalize ontologies?

We combined the pluralizer, CFGs, and verbalization algorithms into a single Java implementation as a Protégé plugin. We used an XML-based annotation model to associate each concept and role in an ontology with its Runyankore translation, part-of-speech, and noun class. In Section 6.3, we developed the architecture of the grammar engine, which explains how crucial the noun class is to the verbalization process. In Section 6.5, we used the rate test and the human-based comparison test to evaluate the generated text with 100 non-linguists and two linguists. For the rate test, most of the generated text was regarded as grammatically correct and understandable by a statistically significant number of study participants; while the human-based comparison test resulted in all the generated text being regarded as human-authored by the majority of non-linguists.

9.2.4 RQ4: How can the Runyankore and isiZulu approaches to ontology verbalization be generalized to other agglutinating Bantu languages?

When investigating generalizability, we focused on the use of the bootstrap approach because some Runyankore verbalization patterns were bootstrapped from isiZulu ones. So because Runyankore and isiZulu belong to different Guthrie zones [123] (see Figure 7.1 in Section 7.2 in Chapter 7), we investigated both bootstrapping among languages in different zones (inter-zone bootstrappability), and bootstrapping among languages in the same zone (intra-zone bootstrappability). We selected five languages, two in the same zones as Runyankore and isiZulu, Luganda (J.10) and isiXhosa (S.40) respectively; and three from three different zones: chiShona (S.10), Kikuyu (E.50), and Kinyarwanda (J.60). In Section 7.3, we found that the same limitations that affect Meinhof's noun class system [136] when it is used in computational tasks identified in Runyankore and isiZulu are also present in these five languages. We then systematically regrouped the noun class system based on a syntactic and morphological approach in Section 7.4, which increased the number of NCs from 29 to 54.

In Section 7.5, we investigated bootstrappability by verbalizing five constructors from the DL $\mathcal{ALC}$ in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda. We found that the same factors that affect verbalization in Runyankore and isiZulu also apply to verbalization in these five languages, and showed how their verbalizations can be bootstrapped from Runyankore and/or isiZulu. Due to having two source languages, we developed a measure of bootstrappability in Section 7.6, in order to identify the most efficient source–target language bootstrap pair. The higher the bootstrappability value, the more the effort required for bootstrapping, implying inefficiency. Using this metric, we showed that intra-zone bootstrappability is mostly more efficient than inter-zone bootstrappability, and that inter-zone bootstrappability is mostly possible.

In Section 8.2, we investigated whether the same approach used to pluralize nouns in Runyankore and isiZulu can pluralize nouns in other agglutinating Bantu languages, we found that a combined morphology with syntax and semantic approach was also necessary for noun pluralization in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda. We found that when language-specific resources are used, the rules on pluralizing standard nouns and exceptions can be generalized. We developed a generic pluralizer which achieved high levels of accuracy in both verification and validation testing, with a minimum of 93% and up to 100%.

In Section 8.3, we investigated whether Context-Free Grammars (CFGs) can also be used to conjugate verbs in other agglutinating Bantu languages. By studying literature on the verbal morphologies of chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda, we found that CFGs can also be used to conjugate their verbs, and showed the CFGs and derivations for each language. However, due to differences in their verbal morphologies, we concluded that a generic CFG cannot be applied.

In Section 8.4, we investigated the possibility of developing an API to verbalize ontologies in agglutinating Bantu languages. We did this by identifying: (1) the language-specific processes, which are verb conjugation, verbalization patterns, and phonological conditioning; (2) the language-specific resources, which are the noun class system, ontology annotations, singular-only nouns, and prefix exceptions; and (3) the generic processes, which are the processes of noun pluralization, bootstrapping, annotating, and verbalizing. We showed the relationships between these components in the architecture in Figure 8.2.

9.3 Research Contributions

This research aimed to generate text in a grammatically complex and computationally under-resourced family of languages, agglutinating Bantu languages. The results of our research provide contributions in this area, both methodologically and resource-wise.

9.3.1 Surface Realization in Agglutinating Bantu Languages

Our research lies in the field of Natural Language Generation (NLG). Of the six NLG tasks, we focused on linguistic realization, because the first three (content determination, discourse planning, and sentence aggregation) are language-independent [63], and therefore not concerned with the linguistic complexity of Bantu languages. Referring expression generation was also not researched here because it has received a lot of attention over the years, and a number of stand-alone solutions exist [63]. For lexicalization, we applied a straightforward model by converting domain concepts into lexical items, because we were working in a well-defined domain [63]. We instead made contributions to the task of linguistic realization, particularly on generating the right morphological forms, which include verb conjugation, agreement, noun pluralization, verbalization, and phonological conditioning.

We selected ontologies as inputs, which have been used to generate text in English [4, 85, 93], as well as in highly synthetic languages like Latvian and Lithuanian [68, 69]. We have now contributed the verbalization of some Bantu languages to this field, and have shown our approach to be applicable, not just to a single Bantu language, but to be generalizable to other agglutinating Bantu languages. The benefit of working at the level of linguistic realization is that our approach is independent of a specific application, and can thus be shared among applications, as noted by Gatt and Krahmer [63].

9.3.2 Linguistic Approaches, Resources, and Tools

During the course of this research, we have developed several approaches, resources, and software tools that can be used for Bantu languages, where the existing ones were found to be inapplicable to Bantu languages due to their complex grammatical structure and under-resourced state. Our work has contributed to filling this gap.

Approaches

We have contributed the following approaches during this research:

(1) An approach to bootstrap Controlled Natural Languages (CNLs) for agglutinating Bantu languages based on a grammar engine in [139];

(2) An approach for handling phonological conditioning during noun pluralization in Section 3.5, verb conjugation in Section 4.5, and verbalization of a nasal compound in Section 5.5;

(3) Architectures for a grammar engine for a single Bantu language, Runyankore, in Section 6.3, for a generic pluralizer in Section 8.2, and for an API for agglutinating Bantu languages in Section 8.4; and

(4) A combined morphology with syntax approach to systematically regroup the NC system, so as to reduce its limitations when applied in a computational context.

We have contributed two architectures which show how computational resources can be developed for agglutinating Bantu languages, despite their complex grammatical structure. In Figure 6.1, we represented the architecture of the Runyankore grammar engine as a Protégé5X plugin, with Protégé only providing the user interface. The architecture includes the key processes of noun pluralization, verb conjugation, and verbalization, which all require phonological conditioning, as well as access to the NC system. It is an architecture that can apply to any other agglutinating Bantu language. In Figure 8.2, we showed the architecture for an API for verbalization of agglutinating Bantu languages. This architecture shows how to solve the problem of how to generalize when there are several language-specific aspects to consider, by showing how the language-specific and generic processes and resources interact to generate text.

One of the major contributions of our research is the regrouping of noun classes in Section 7.4, where we explained the rationale behind the regrouping. This should make it possible for other researchers of Bantu languages not covered in this research to also regroup the noun classes of their languages when attempting to develop language-independent computational tools. The most important thing about our noun class regrouping is that it fosters generalizability by eliminating non-determinism. With over 300 known Bantu languages, generalizing computational approaches is the best way to achieve wide coverage.

Software Tools

The following tools were developed and have been made available during this research:

(1) A Runyankore noun pluralizer[1];

(2) A Runyankore verbalizer as a Protégé5X plugin[2]; and

(3) A generic noun pluralizer[3].

Resources

We also contribute the following resources used and/or identified during this research:

(1) Noun pluralization datasets for chiShona, isiXhosa, Kikuyu, Kinyarwanda, Luganda, and Runyankore[4];

(2) Verbalizing the DL $\mathcal{ALC}$ in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda in Section 7.5; and for Runyankore, the DL $\mathcal{ALCQ}$ in Section 5.3;

(3) Derivation rules for CFGs [188] for the simple present tense with negation in chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda (see Section 8.3 in Chapter 8); and for Runyankore, additionally including the participial present continuous tense and the passive (See Sections 4.4 and 4.6 in Chapter 4); and

(4) Regrouped noun classes for chiShona, isiXhosa, isiZulu, Kikuyu, Kinyarwanda, Luganda, and Runyankore.

9.3.3 Metric for Measuring Bootstrappability

Our work to bootstrap the Runyankore verbalization patterns from the isiZulu ones was the first for this class of languages. We identified the need to measure the efficiency of bootstrapping when there are multiple source languages, and developed a metric for this in Section 7.6. This novel metric is based on the concepts of minimum edit distance [91] and linear gap penalty [66]. By investigating bootstrappability, our research also contributes verbalization patterns for chiShona, isiXhosa, Kikuyu, Kinyarwanda, and Luganda. These, in addition to Runyankore and isiZulu, can be used as source languages by researchers for a new language in this area, and our metric makes it possible to identify the most efficient source–target language bootstrap pair.

[1]The Runyankore noun Pluralizer is available at https://github.com/ThesisResources/RunyankorePluralizer.
[2]The Protégé plugin is available at https://github.com/runyankorenlg/RunyankoreNLGSystem.
[3]The generic pluralizer is available at https://github.com/runyankorenlg/Generic-Pluralizer.
[4]These datasets are available at https://github.com/runyankorenlg/Generic-Pluralizer.

9.3.4 Applicability to the Healthcare Domain

The healthcare domain was our focus in this research, due to the need to provide patients with additional information in the language that the patients understand. Efforts to address the communication gap between healthcare practitioners and patients in South Africa have resulted in mobile translators like Mobile Xhosa [138] and Mobile Zulu [139], which are composed of translations of basic medical phrases in English, isiZulu, and isiXhosa; or mobile assisted language learning tools targeted at teaching healthcare practitioners healthcare vocabulary in an indigenous language, for example, Northern Sotho [187]. The limitations of these approaches are: the mobile translators provide text that is fixed and cannot be updated as need arises; and the mobile assisted language learning tools require the healthcare practitioners to learn a new language. We proposed a dynamic approach, NLG for healthcare, which has successfully been applied elsewhere [31, 46, 53, 81, 117, 120] but not with a Bantu language. While our research has not resulted in a full NLG system for this task, we have contributed towards the task of linguistic realization, which overcomes the limitations originating from the complex grammatical structure of Bantu languages. Additionally, the healthcare text generated in Runyankore by verbalizing part of SNOMED-CT was regarded as grammatically correct and understandable by 66% of non-linguists, and was regarded as human-authored by 71% of non-linguists and by both linguists. This suggests that the generated text could be trusted by non-linguists as having been written by a human, and they would be able to understand it. Further, by generalizing the Runyankore approach to five more Bantu languages, and showing that it can be applied to even more, our research contributes a crucial component required to build an NLG system for healthcare messages in agglutinating Bantu languages.

9.4 Future Work

We have identified areas for future work, which we present below.

9.4.1 Runyankore NLG System

The grammar engine we have developed is a surface realizer for Runyankore. It would be ideal to develop a complete NLG system that accounts for all the NLG tasks.

9.4.2 Evaluating Generated Healthcare Messages in a Real-world Setting

Our domain of interest in this research was healthcare. Though we used examples of healthcare messages in this thesis, and the healthcare-based sentences in the Runyankore evaluation in Section 6.5 in Chapter 6 were regarded as grammatically correct, understandable, and human-authored by a statistically significant number of non-linguists, it would be ideal to test the efficacy of such generated text in a real-world, in order to assess the value of such personalized messages to Bantu language speakers.

9.4.3 Handling Phonological Conditioning in Generic Pluralizer

With the exception of isiZulu, all the errors found in the results of the generic pluralizer resulted from the need for phonological conditioning. Though the rules for phonological conditioning are largely language-specific, an investigation into how this can be achieved whilst maintaining generalizability would be ideal.

9.4.4 Better Grammar Formalism for Verb Conjugation

While CFGs were successfully used to conjugate verbs in Runyankore and isiZulu, and showed to also be applicable to five more agglutinating Bantu languages, a limited scope of the verbal

morphology had to be used in each case. This is because CFGs are not sufficient to account for the entire verbal morphology of Bantu languages, including phonological conditioning. And while, in theory, a very large CFG can be developed with additional rules to handle phonological conditioning, it would be a very inefficient approach. It would thus be ideal to develop a grammar formalism specific for the complex verbal morphology of Bantu languages.

9.4.5 Building API

We proposed an architecture for an API to verbalize ontologies in agglutinating Bantu languages. It would be ideal to develop an API based on this architecture and evaluate its output.

9.5 Final Remarks

The large number of indigenous languages in use in Africa has created a communication problem between health practitioners and the populations they serve, where healthcare practitioners are unable to speak their patients' first languages, especially in the rural areas where indigenous languages are still predominantly spoken. With existing technology-based efforts to solve this problem, we hope that our work can be applied to these existing efforts, in order to bridge the communication gap by generating localized medical text for millions of patients in Africa.

Bibliography

[1] AfLaT.org. *African Language Technology*. Accessed on 27/07/2019. 2019. URL: http://www.aflat.org/ (cit. on p. 5).

[2] Sarah Alkuhlani and Nizar Habash. "Identifying Broken Plurals, Irregular Gender, and Rationality in Arabic Text". In: *13th Conference of the European Chapter of the Association for Computational Linguistics*. 23-27 April, 2012, Avignon, France. ACL, 2012, pp. 675–685 (cit. on p. 26).

[3] Mohamed Altantawy et al. "Morphological Analysis and Generation of Arabic Nouns: A Morphemic Functional Approach". In: *7th International Conference on Language Resources and Evaluation (LREC'10)*. Ed. by Nicoletta Calzolari (Conference Chair) et al. Valletta, Malta: European Language Resources Association (ELRA), 2010. ISBN: 2-9517408-6-7 (cit. on p. 26).

[4] Ion Androutsopoulos, Gerasimos Lampouras, and Dimitrios Galanis. "Generating Natural Language descriptions from OWL ontologies: the NaturalOWL system". In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 671–715 (cit. on pp. 7, 23, 66, 129).

[5] Gabor Angeli, Percy Liang, and Dan Klein. "A Simple Domain-independent Probabilistic Approach to Generation". In: *2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2010, pp. 502–512 (cit. on p. 14).

[6] Krasimir Angelov and Aarne Ranta. "Implementing Controlled Languages in GF". In: *Workshop on Controlled Natural Language (CNL'09)*. Marettimo Island, Italy: Springer, 2009, pp. 82–101 (cit. on p. 83).

[7] K P Antony P J an Samon. "Computational Morphology and Natural Language Parsing for Indian Languages: A Literature Survey". In: *International Journal of Scientific and Engineering Research* 3 (3 2012) (cit. on p. 43).

[8] Allen Asiimwe. "Definiteness and Specificity in Runyankore-Rukiga". PhD thesis. Cape Town, South Africa: Stallenbosch University, 2014 (cit. on pp. 16, 17, 19, 26, 34).

[9] Franz Baader, Ian Horrocks, and Ulrike Sattler. "Description Logics". In: *Handbook of Knowledge Representation*. Ed. by F. van Harmelen, V. Lifschitz, and B. Porter. Elsevier, 2008. Chap. 3, pp. 135–180 (cit. on pp. 21, 39).

[10] Franz Baader and Werner Nutt. "Basic Description Logics". In: *The Description Logic Handbook: Theory, Implementation, and Applications*. Ed. by Franz Baader et al. New York: Cambridge University Press, 2003. Chap. 2 (cit. on pp. 21, 39, 51).

[11] Elizabeth Baertlein and Martin Ssekitto. "Luganda Nouns Inflectional Morphology and Tests." In: *Linguistic Portfolios* 3 (3 2014) (cit. on pp. 16, 17, 84, 85, 156).

[12] J. A. Bateman. "Enabling Technology for Multi-lingual Natural Language Generation: The KPML Development Environment". In: *Natural Language Engineering* 3.1 (1997), pp. 15–55 (cit. on p. 12).

[13] A. Belz. "Automatic Generation of Weather Forecast Texts Using Comprehensive Probabalistic Generation-Space Models". In: *Natural Language Engineering* 14.04 (2008) (cit. on pp. 2, 10).

[14] Abraham Bernstein and Esther Kaufmann. "GINO-A Guided Input Natural Language Ontology Editor". In: *International Semantic Web Conference*. Springer. 2006, pp. 144–157 (cit. on p. 22).

[15] Abraham Bernstein et al. "Ginseng A Guided Input Natural Language Search Engine for Querying Ontologies". In: *Jena User Conference*. Citeseer. 2006 (cit. on p. 23).

[16] Dianne C Berry, Tony Gillie, and Simon Banbury. "What do patients want to know: an empirical approach to explanation generation and validation". In: *Expert Systems with Applications* 8.4 (1995), pp. 419–428 (cit. on pp. 41–44, 46, 51).

[17] Dianne C Berry et al. "What do patients want to know about their medicines, and what do doctors want to tell them?: A comparative study". In: *Psychology and Health* 12.4 (1997), pp. 467–480 (cit. on pp. 41–44, 46, 51).

[18] Marcel Bollmann. "Adapting SimpleNLG to German". In: *13th European Workshop on Natural Language Generation (ENLG 2011)*. Nancy, France, 2011, pp. 133–138 (cit. on pp. 12, 82, 83).

[19] Sonja E. Bosch and Laurette Pretorius. "Building a Computational Morphological Analyser/Generator for Zulu using the Xerox Finite-State Tools". In: *Workshop on Finite-State Methods in Natural Language Processing, 10th Conference of the European Chapter of the Association for Computational Linguistics*. Budapest: Association for Computational Linguistics, 2003, pp. 27–34 (cit. on p. 5).

[20] Sonja E. Bosch and Laurette Pretorius. "Software Tools for Morphological Tagging of Zulu Corpora and Lexicon Development". In: *4th International Language Resources and Evaluation Conference*. Lisbon, Portugal, 2004, pp. 1251–1254 (cit. on p. 5).

[21] Sonja Bosch, Laurette Pretorius, and Axel Fleisch. "Experimental Bootstrapping of Morphological Analyzers for Nguni Languages". In: *Nordic Journal of African Studies* 17.2 (2008), pp. 66–88 (cit. on pp. 5, 82, 83).

[22] Nadjet Bouayad-Agha, Gerard Casamayor, and Leo Wanner. "Natural language generation in the context of the semantic web". In: *Semantic Web* 5.6 (2014), pp. 493–513 (cit. on pp. 1, 7–10).

[23] Paul Buitelaar et al. "Towards linguistically grounded ontologies". In: *European Semantic Web Conference*. Springer. 2009, pp. 111–125 (cit. on pp. 21, 23, 120).

[24] Joan Byamugisha, C. Maria Keet, and Brian DeRenzi. "Bootstrapping a Runyankore CNL from an isiZulu CNL". In: *5th Workshop on Controlled Natural Language (CNL 2016)*. Vol. 9767. Aberdeen, Scotland: Springer LNAI, 2016, pp. 25–36 (cit. on pp. 51, 66, 81, 83, 92).

[25] Joan Byamugisha, C. Maria Keet, and Brian DeRenzi. "Tense and Aspect in Runyankore using a Context-Free Grammar". In: *9th International Conference on Natural Language Generation (INLG 2016)*. Edinburgh, Scotland: ACL, 2016, pp. 84–88 (cit. on pp. 39, 42, 43, 95, 114).

[26] Joan Byamugisha, C. Maria Keet, and Brian DeRenzi. "Evaluation of a Runyankore grammar engine for healthcare messages". In: *10th International Conference on Natural Language Generation (INLG 2017)*. Vol. 4. Santiago de Compostela, Spain: ACL, 2017, pp. 105–113 (cit. on pp. 39, 68, 81).

[27] Joan Byamugisha, C. Maria Keet, and Brian DeRenzi. "Toward an NLG System for Bantu languages: first steps with Runyankore (demo)". In: *10th International Conference on Natural Language Generation (INLG 2017)*. Vol. 4. Santiago de Compostela, Spain: ACL, 2017 (cit. on p. 68).

[28] Joan Byamugisha, C. Maria Keet, and Brian DeRenzi. "Pluralizing Nouns in Agglutinating Bantu Languages". In: *27th International Conference on Computational Linguistics (COLING 2018)*. Santa Fe, New Mexico, USA: ACL, 2018, pp. 2633–2643 (cit. on pp. 82, 109).

[29] Joan Byamugisha, C. Maria Keet, and Langa Khumalo. "Pluralizing Nouns in isiZulu and Related Languages". In: *17th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2016)*. Vol. 9626. Konya, Turkey: Springer LNCS, 2016, pp. 271–283 (cit. on pp. 5, 26–28, 31, 34, 37, 81, 109–113, 122, 123).

[30] Diego Calvanese et al. "Ontologies and databases: The DL-Lite approach." In: *5th International Reasoning Web Summer School (RW 2009), Semantic Technologies for Information Systems*. Ed. by Sergio Tessaris and Franconi Enrico. Vol. 5689. Springer LNCS, 2009, pp. 255–356 (cit. on pp. 20, 21).

[31] J. Alison Cawsey, B. Ray Jones, and Janne Pearson. "The Evaluation of a Personalized Health Information System for Patients with Cancer". In: *User Modeling and User-Adapted Interaction* 10.1 (2000), pp. 47–72 (cit. on pp. 1, 2, 10, 24, 131).

[32] Catherine Chavula and C. Maria Keet. "Is Lemon Sufficient for Building Multilingual Ontologies for Bantu Languages?" In: *11th OWL Experiences and Directions Workshop (OWLED'14)*. Ed. by C. M. Keet and V. Tamma. Vol. 1265. Riva del Garda, Italy, 2014, pp. 61–72 (cit. on pp. 19, 24).

[33] David L Chen and Raymond J Mooney. "Learning to Sportscast: A Test of Grounded Language Acquisition". In: *25th International Conference on Machine Learning*. ACM. 2008, pp. 128–135 (cit. on p. 15).

[34] Guanyi Chen, Kees van Deemter, and Chenghua Lin. "SimpleNLG-ZH: A Linguistic Realization Engine for Mandarin". In: *11th International Conference on Natural Language Generation*. Tilburg, The Netherlands, 2018, pp. 57–66 (cit. on pp. 12, 82, 83).

[35] Philipp Cimiano et al. "LexInfo: A Declarative Model for the Lexicon-Ontology Interface". In: *Journal of Web Semantics* 9.1 (2011), pp. 29–51 (cit. on pp. 23, 120).

[36] Philipp Cimiano et al. "On the Role of Senses in the Ontology-Lexicon". In: *New Trends of Research in Ontologies and Lexical resources: Ideas, Projects, Systems*. Springer Berlin Heidelberg, 2013, pp. 43–62 (cit. on p. 79).

[37] Emilie Colin et al. "The WebNLG Challenge: Generating Text from DBPedia Data". In: *9th International Conference on Natural Language Generation (INLG 2016)*. Edinburgh, UK: Association for Computational Linguistics, 2017, pp. 163–167 (cit. on p. 22).

[38] D. M. Conway. "An algorithmic approach to English pluralization". In: *2nd Annual Perl Conference*. Ed. by C. Salzenberg. San Jose, USA: O'Reilly, 1998 (cit. on p. 26).

[39] N. Daoust and G. Lapalme. "JSreal: A Text Realizer for Web Programming". In: *Language Production, Cognition, and the Lexicon* (2014), pp. 363–378 (cit. on p. 83).

[40] Brian Davis et al. "Roundtrip Ontology Authoring". In: *International Semantic Web Conference*. Springer. 2008, pp. 50–65 (cit. on p. 23).

[41] Berardina De Carolis et al. "Generating recipient-centered explanations about drug prescription". In: *Artificial Intelligence in Medicine* 8.2 (1996), pp. 123–145 (cit. on pp. 41, 42).

[42] Rodrigo de Oliveira and Somayajulu Sripada. "Adapting SimpleNLG for Brazilian Portuguese Realisation". In: *8th International Conference on Natural Language Generation (INLG 2014)*. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2014, pp. 93–94 (cit. on pp. 12, 82, 83).

[43] Guy De Pauw, Gilles-Maurice de Schryver, and Janneke van de Loo. "Resource-light Bantu Part-Of-Speech Tagging". In: *Workshop on Language Technology for Normalisation of Less-Resourced Languages (SaLTMiL 8 - AfLaT 2012)*. Ed. by Guy De Pauw et al. Istanbul, Turkey, 2012, pp. 85–92 (cit. on p. 5).

[44] Guy De Pauw, Peter W. Wagacha, and Gilles-Maurice de Schryver. "Exploring the SAWA Corpus: Collection and Deployment of a Parallel Corpus English-Swahili". In: *Language Resources and Evaluation* 45.3 (2011), pp. 331–344 (cit. on p. 5).

[45] Fiorella De Rosis and Floriana Grasso. "Affective Natural Language generation". In: *Affective interactions*. Springer, 2000, pp. 204–218 (cit. on pp. 41, 42).

[46] Fiorella de Rosis, Floriana Grasso, and C. Dianne Berry. "Refining Instructional Text Generation after Evaluation". In: *Artificial Intelligence in Medicine* 17.1 (1999), pp. 1–36 (cit. on pp. 1, 2, 7, 24, 41, 73, 131).

[47] Rose-Marie Dechaine et al. "The Internal Syntax of Shona Class Prefixes". In: *Language Sciences* 43 (2013), pp. 18–46 (cit. on pp. 84, 85).

[48] Chrysanne Di Marco et al. "Authoring and generation of tailored preoperative patient education materials". In: *Personalisation for e-Health* 11.55 (2005), pp. 111–131 (cit. on pp. 1, 10, 24).

[49] Chrysanne Di Marco et al. "A Physician's Authoring Tool for Generation of Personalized Health Education in Reconstructive Surgery." In: *AAAI Spring Symposium: Argumentation for Consumers of Healthcare*. Stanford University, 2006, pp. 39–46 (cit. on pp. 1, 10, 24).

[50] Chrysanne Di Marco et al. "Authoring and generation of individualized patient education materials". In: *Conference of the American Medical Informatics Association (AMIA) Annual Symposium Proceedings*. Vol. 2006. American Medical Informatics Association. Washington D. C., 2006, p. 195 (cit. on pp. 1, 24).

[51] Chrysanne DiMarco, Graeme Hirst, and Eduard Hovy. "Generation by selection and repair as a method for adapting text for the individual reader". In: *Workshop on Flexible Hypertext, Eighth ACM International Hypertext Conference*. Southampton, U. K., 1997 (cit. on pp. 1, 24).

[52] Chrysanne DiMarco, David Wiljer, and Eduard Hovy. "Self-Managed Access to Personalized Healthcare through Automated Generation of Tailored Health Educational Materials from Electronic Health Records". In: *AAAI Fall Symposium on Virtual Health Interaction*. Washington D. C., 2009 (cit. on pp. 1, 2, 10, 24, 41).

[53] Chrysanne DiMarco et al. "HealthDoc: Customizing Patient Information and Health Education by Medical Condition and Personal Characteristics". In: *Workshop on Artificial Intelligence in Patient Education*. Glasgow, Scotland, 1995 (cit. on pp. 1, 2, 7, 10, 24, 131).

[54] Chrysanne DiMarco et al. "The Development of a Natural Language Generation System for Personalized e-Health Information". In: *12th World Congress on Helth (Medical) Informatics-Building Sustainable Health Systems (Medinfo 2007)*. Ed. by K. Kuhn, J. Warren, and LEONG. Tze-Yun. Vol. 129. Brisbane, Australia: IOS Press, 2007 (cit. on pp. 2, 10, 24, 41, 73).

[55] Ondřej Dušek and Filip Jurcicek. "Training a Natural Language Generator from Unaligned Data". In: *53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 451–461 (cit. on p. 15).

[56] Michael Elhadad and Jacques Robin. "An Overview of SURGE: A Reusable Comprehensive Syntactic Realization Component". In: *8th International Natural Language Generation Workshop: Posters and Demonstrations*. Association for Computational Linguistics, 1996 (cit. on p. 11).

[57] Basil Ell and Andreas Harth. "A Language-Independent Method for the Extraction of RDF Verbalization Templates". In: *8th International Natural Language Generation Conference (INLG)*. 2014, pp. 26–34 (cit. on p. 15).

[58] D. Espinosa, M. White, and D. Mehay. "Hypertagging: Supertagging for Surface Realization with CCG". In: *The Annual Meeting of the Association for Computational Linguistics and the Human Language Technology Conference (ACL-HLT'08)*. Columbus, Ohio, USA, 2008, pp. 183–191 (cit. on pp. 2, 11).

[59] Scott Farrar and D. Terence Langendoen. "An OWL-DL Implementation of GOLD: An Ontology for the Semantic Web". In: *Linguistic Modeling of Information and Markup Languages*. Dordrecht, Springer, 2009. Chap. 6 (cit. on pp. 21, 39).

[60] Franca Ferrari-Bridgers. "Luganda Verb Morphology: A New Analysis of the Suffixes '-ye' and '-a', and their Distribution Across the Indicative, Subjunctive, and Imperative Moods". In: *Studies in African Linguistics* 38.1 (2009) (cit. on pp. 18, 84, 85, 95, 114, 118).

[61] Claire Gardent et al. "Creating Training Corpora for Micro-Planners". In: *55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017 (cit. on p. 22).

[62] A. Gatt et al. "From Data to Text in the Neonatal Intensive Care Unit: Using NLG Technology for Decision Support and Information Management". In: *AI Communications* 22.3 (2009), pp. 153–186 (cit. on pp. 1, 2, 7).

[63] Albert Gatt and Emiel Krahmer. "Survey of the State of the Art in Natural Language Generation: Core tasks, applications and evaluation". In: *Journal of Artificial Intelligence Research* 61 (2018), pp. 65–170 (cit. on pp. 1, 2, 7–15, 72, 73, 129).

[64] Albert Gatt and Ehud Reiter. "SimpleNLG: A Realization Engine for Practical Applications". In: *12th European Workshop on Natural Language Generation (ENLG'09)*. Athens, Greece, 2009, pp. 90–93 (cit. on pp. 11, 12, 49, 79, 82, 123).

[65] Girija Goddol. "Natural Language Generation". In: *International Journal of Engineering Research and General Science* 3.2 (2015) (cit. on pp. 7, 8).

[66] O. Gotoh. "An Improved Algorithm for Matching Biological Sequences". In: *Journal of Molecular Biology* 16.2 (1982), pp. 705–708 (cit. on pp. 101, 102, 130).

[67] Rebecca Grollemund et al. "Bantu Expansion Shows that Habitat Alters the Route and Pace of Human Dispersals". In: *National Academy of Sciences* 112.43 (2015), pp. 13296–13301 (cit. on p. 19).

[68] Normunds Gruzitis and Guntis Barzdins. "Towards a More Natural Multilingual Controlled Language Interface to OWL". In: *ninth International Conference on Computational Semantics*. Association for Computational Linguistics. 2011, pp. 335–339 (cit. on pp. 21–23, 66, 129).

[69] Normunds Gruzitis, Gunta Nespore, and Baiba Saulite. "Verbalizing Ontologies in Controlled Baltic Languages". In: *4th International Conference on Human Language Technologies-The Baltic Perspective*. Riga, Latvia, 2010 (cit. on pp. 2, 21–23, 64, 66, 67, 129).

[70] Nicola Guarino. "Formal Ontology and Information Systems". In: *FOIS'98*. Trento, Italy: IOS Press, Amsterdam, 1998, pp. 3–15 (cit. on p. 21).

[71] Nicola Guarino, Daniel Oberle, and Steffen Staab. "What is an Ontology?" In: *Handbook on Ontologies*. Springer, 2009. Chap. 6, pp. 1–17 (cit. on pp. 2, 21).

[72] Malcolm Guthrie. *The Classification of the Bantu Languages*. London: Oxford University Press, 1948 (cit. on pp. 3, 19).

[73] Catalina Hallett, Donia Scott, and Richard Power. "Composing Questions Through Conceptual Authoring". In: *Computational Linguistics* 33.1 (2007), pp. 105–133 (cit. on p. 22).

[74] M. D. Harris. "Building a Large-Scale NLG System for an EMR". In: *The 5th International Conference on Natural Language Generation (INLG'08)*. Ohio, USA, 2008, pp. 157–160 (cit. on pp. 1, 2, 7).

[75] Johnson M Hart. "The Derivation Language of a Phrase Structure Grammar". In: *Journal of Computer and System Sciences* 12.1 (1976), pp. 64–79 (cit. on p. 43).

[76] Graeme Hirst. "Ontology and the lexicon". In: *Handbook on Ontologies*. Springer, 2009, pp. 269–292 (cit. on p. 79).

[77] Graeme Hirst et al. "Authoring and Generating Health-Education Documents that are Tailored to the Needs of the Individual Patient". In: *6th International Conference on User Modeling (UM'97)*. Ed. by Anthony Jameson, Cecile Paris, and Carlo Tasso. Chia Laguna, Sardinia, Italy: Springer Verlag, 1997, pp. 107–118 (cit. on pp. 2, 24).

[78] J. Hockenmaier and M. Steedman. "CCBank: A Corpus of CCG Derivations and Dependency Structures Extracted from the Penn Treebank". In: *Computational Linguistics* 33.3 (2007), pp. 355–396 (cit. on p. 11).

[79] Eduard Hovy. "Natural Language Generation". In: *MITECS* (1998) (cit. on p. 10).

[80] Arvi Hurskainen. "A Two-level Computer Formalism for the Analysis of Bantu Morphology: An Application to Swahili". In: *Nordic Journal of African Studies* 1.1 (1992), pp. 87–122 (cit. on pp. 5, 43).

[81] Mohammed Sazzad Hussain et al. "Nlg-based moderator response generator to support mental health". In: *33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM. Seoul, South Korea: ACM, 2015, pp. 1385–1390 (cit. on pp. 1, 2, 24, 73, 76, 131).

[82] Rebecca Hwa et al. "Bootstrapping Parsers via Syntactic Projection across Parallel Texts". In: *Natural Language Engineering* 11.3 (2005), pp. 311–325 (cit. on p. 82).

[83] International Health Terminology Standards Development Organization (IHTSDO). *Systematized Nomenclature of Medicine–Clinical Terms (SNOMED-CT)*. 2015. URL: `http://browser.ihtsdotools.org/` (cit. on pp. 2, 70, 72, 74).

[84] Srini Janarthanam and Oliver Lemon. "The GRUVE challenge: Generating Routes under Uncertainty in Virtual Environments". In: *13th European Workshop on Natural Language Generation*. 2011, pp. 208–211 (cit. on p. 14).

[85] Mustafa Jarrar, C. Maria Keet, and Paolo Dongilli. *Multilingual Verbalization of ORM Conceptual Models and Axiomatized Ontologies*. Tech. rep. Brussels, Belgium: Vrije Universiteit, 2006 (cit. on pp. 22, 66, 129).

[86] Mustafa Jarrar, C. Maria Keet, and Paolo Dongilli. *Multilingual Verbalization of ORM Conceptual Models and Axiomatized Ontologies*. Tech. rep. Brussels, Belgium: Vrije Universiteit, 2006 (cit. on p. 82).

[87] V. Jayan and V. K. Bhadran. "Difficulties in Processing Malayalam Verbs for Statistical Machine Translation". In: *International Journal of Artificial Intelligence and Applications (IJAIA)* 6 (3 2015) (cit. on p. 43).

[88] Lisa Jeon et al. "A Basic Sketch Grammar of Gikuyu". In: *Rice Working Papers in Linguistics* 6 (2015). Ed. by Robert Englebretson and Wambui Muringoo Wa-Ngatho (cit. on pp. 16–18, 84, 85, 99, 114, 116, 117, 123, 155).

[89] Mthuthuzeli Todd Jobela. "Negative Constructions in isiXhosa". MA thesis. Stellenbosch, South Africa: University of Stellenbosch, 2000 (cit. on pp. 84, 91, 92, 95, 114, 115).

[90] Aravind K Joshi and Yves Schabes. "Tree-adjoining Grammars". In: *Handbook of Formal Languages*. Springer, 1997, pp. 69–123 (cit. on p. 13).

[91] Daniel Jurafsky and H. James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. USA: Prentice Hall, Inc., 2007 (cit. on pp. 3, 7, 8, 10, 11, 43, 49, 101, 130).

[92] Kaarel Kaljurand. "ACEView–An Ontology and Rule Editor based on Attempto Controlled English." In: *5th Workshop on OWL Experiences and Directions (OWLED '08)*. 2008 (cit. on p. 23).

[93] Kaarel Kaljurand and Norbert E Fuchs. "Verbalizing OWL in Attempto Controlled English." In: *OWLED*. Vol. 258. Innsbruck, Austria, 2007 (cit. on pp. 2, 22, 66, 129).

[94] Francis Katamba. "Bantu Nominal Morphology". In: *The Bantu Languages: Routledge Language Family Series 4*. London: Taylor and Francis Routledge, 2003. Chap. 7, pp. 103–120 (cit. on pp. 15–18, 26, 53, 84, 89).

[95] C Maria Keet and Langa Khumalo. "Grammar Rules for the isiZulu Complex Verb". In: *Southern African Linguistics and Applied Language Studies* 35.2 (2017), pp. 183–200 (cit. on pp. 12, 43, 49).

[96] C. M. Keet, M. Xakaza, and L. Khumalo. "Verbalising OWL ontologies in isiZulu with Python". In: *14th Extended Semantic Web Conference (ESWC'17)*. Vol. 10577. Portoroz, Slovenia: Springer LNCS, 2017, pp. 59–64 (cit. on pp. 5, 23, 81).

[97] C. Maria Keet and Takunda Chirema. "A Model for Verbalizing Relations with Roles in Multiple Languages". In: *20th International Conference on Knowledge Engineering and Knowledge Management (EKAW'16')*. Ed. by E. Blomqvist et al. Vol. 10024. Bologna, Italy: Springer LNAI, 2016, pp. 384–399 (cit. on pp. 12, 23, 24, 68).

[98] C. Maria Keet and Langa Khumalo. "Basics for a Grammar Engine to Verbalize Logical Theories in isiZulu". In: *8th International Web Rule Symposium (RuleML'14)*. Vol. 8620. Prague, Czech Republic: Springer LNCS, 2014, pp. 216–225 (cit. on pp. 5, 12, 23, 66, 81, 92–94).

[99] C. Maria Keet and Langa Khumalo. "Towards Verbalizing Logical Theories in isiZulu". In: *4th Workshop on Controlled Natural Languages (CNL'14)*. Ed. by B. Davis, T. Kuhn, and K. Kaljurand. Vol. 8625. Galway, Ireland: Springer LANI, 2014, pp. 78–89 (cit. on pp. 2, 5, 12, 16, 17, 23, 64, 66, 67, 81, 90, 94).

[100] C. Maria Keet and Langa Khumalo. "Towards a Knowledge-to-Text Controlled Natural Language of isiZulu". In: *Language Resources and Evaluation* 51 (2017), pp. 131–157 (cit. on pp. 2, 5, 10, 24, 81, 95, 114).

[101] Alex Kimenyi. "Kinyarwanda Morphology". In: *Morphology: An International Handbook for Inflection and Word Formation*. Ed. by Geert Booij et al. Vol. 17.2. De Gruyter, 2004 (cit. on pp. 16, 17, 84, 85, 114, 117, 156).

[102] Alexander Koller and Ronald PA Petrick. "Experiences with Planning for Natural Language Generation". In: *Computational Intelligence* 27.1 (2011), pp. 23–40 (cit. on p. 13).

[103] R. Kondadadi, B. Howald, and F. Schilder. "A Statistical NLG Framework for Aggregated Planning and Realization". In: *Annual Meeting of the Association for Computational Linguistics (ACL'13)*. Sofia, Bulgaria, 2013, pp. 1406–1415 (cit. on p. 10).

[104] Ioannis Konstas and Mirella Lapata. "Unsupervised Concept-to-text Generation with Hypergraphs". In: *2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics. 2012, pp. 752–761 (cit. on p. 15).

[105] Ioannis Konstas and Mirella Lapata. "A Global Model for Concept-to-Text Generation". In: *Journal of Artificial Intelligence Research* 48 (2013), pp. 305–346 (cit. on p. 15).

[106] Kimmo Koskenniemi. *Two-level Morphology: A General Computational Model for Word-Form Recognition and Production*. Vol. 11. University of Helsinki, Department of General Linguistics Helsinki, 1983 (cit. on p. 43).

[107] Tobias Kuhn. "ACEWiki: A Natural and Expressive Semantic Wiki". In: *arXiv preprint arXiv:0807.4618* (2008) (cit. on p. 23).

[108] Tobias Kuhn. "A Survey and Classification of Controlled Natural Language(s)". In: *Computational Linguistics* 40 (01 2014), pp. 121–170 (cit. on pp. 21, 22, 52).

[109] M Anand Kumar, V Dhanalakshmi, and Dr KP Soman. "A Novel Algorithm for Tamil Morphological Generator". In: *ICON* (2010) (cit. on p. 43).

[110] I. Langkilde-Geary. "Forest-based Statistical Sentence Generation". In: *NAACL 2000 Student Research Workshop at the 6th Applied Natural Language Processing Conference (ANLP-NAACL'00)*. Seattle, Washington, USA, 2000, pp. 170–177 (cit. on pp. 2, 10).

[111] I. Langkilde-Geary and K. Knight. "HALogen Statistical Sentence Generator". In: *Annual Meeting of the Association for Computational Linguistics (ACL'02)-Demos*. Philadelphia, USA, 2002, pp. 102–103 (cit. on pp. 2, 10).

[112] Bettina Lanser. "Methods for Efficient Ontology Lexicalization for Non-Indo-European Languages: A Case of Japanese". PhD thesis. Germany: Bielefeld University, 2014 (cit. on p. 79).

[113] C. van der Lee, E. Krahmer, and S. Wubben. "PASS: A Dutch-to-text System for Soccer, Targeted Towards Specific Audiences". In: *The 10th International Conference on Natural Language Generation (INLG 2017)*. Santiago de Compostela, Spain, 2017 (cit. on pp. 7, 8, 10).

[114] L. Lepp et al. "Data-Driven News Generation for Automated Journalism". In: *10th International Conference on Natural Language Generation (INLG 2017)*. Santiago de Compostela, Spain, 2017, pp. 188–197 (cit. on p. 7).

[115] I. Vladimir Levenshtein. "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals". In: *Cybernetics and Control Theory* 10.8 (1966), pp. 707–710 (cit. on p. 101).

[116] Percy Liang, Michael I Jordan, and Dan Klein. "Learning Semantic Correspondences with Less Supervision". In: *Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*. Association for Computational Linguistics. 2009, pp. 91–99 (cit. on p. 14).

[117] Fredrik Lindahl. "Practical text generation in clinical medicine". In: *Chalmers Computer Science Department Winter Meeting*. Citeseer. Gullmarstrand, Sweden, 2005 (cit. on pp. 1, 2, 24, 131).

[118] Zachary C Lipton, Sharad Vikram, and Julian McAuley. "Generative Concatenative Nets Jointly Learn to Write and Classify Reviews". In: *arXiv preprint arXiv:1511.03683* (2015), pp. 1–11 (cit. on p. 15).

[119] David D MacDonald. "Issues in the Choice of a Source for Natural Language Generation". In: *Computational Linguistics* 19.1 (1993), pp. 191–197 (cit. on p. 9).

[120] Saad Mahamood and Ehud Reiter. "Generating Affective Natural Language for Parents of Neonatal Infants". In: *13th European Workshop on Natural Language Generation (ENLG 2011)*. Association for Computational Linguistics. Nancy, France: Association for Computational Linguistics, 2011, pp. 12–21 (cit. on pp. 1, 2, 7, 8, 10, 24, 131).

[121] Zola Mahlaza. "Grammars for Generating isiXhosa and isiZulu Weather Bulletin Verbs". MA thesis. Cape Town, South Africa: University of Cape Town, 2018 (cit. on pp. 12, 114, 120, 123, 124).

[122] Jouni Maho. "A Comparative Study of Bantu Noun Classes". PhD thesis. Goteborg, Sweden: Goteborg University, 1999 (cit. on pp. 5, 15–18, 26, 28, 33, 53, 85–87, 89, 122, 123).

[123] Jouni Filip Maho. *NUGL Online: The Online Version of the Updated Guthrie List, a referential classification of the Bantu languages*. 2009. URL: `http://goto.glocalnet.net/mahopapers/nuglonline.pdf` (cit. on pp. 3–5, 7, 19, 20, 81, 83, 96, 128).

[124] François Mairesse and Steve Young. "Stochastic Language Generation in Dialogue using Factored Language Models". In: *Computational Linguistics* 40.4 (2014), pp. 763–799 (cit. on p. 14).

[125] Massimo Marchiori. "Towards a People's Web: Metalog". In: *2004 IEEE/WIC/ACM International Conference on Web Intelligence*. IEEE Computer Society. 2004, pp. 320–326 (cit. on p. 23).

[126] Tomasz Marciniak and Michael Strube. "Classification-based Generation using TAG". In: *International Conference on Natural Language Generation*. Springer. 2004, pp. 100–109 (cit. on p. 14).

[127] Christian Matthiessen and John A Bateman. *Text Generation and Systemic-Functional Linguistics: Experiences from English and Japanese*. London and New York: Frances Pinter Publishers and St. Martin's Press, 1991 (cit. on p. 12).

[128] Nhira Edgar Mberi. "The Categorical Status and Functions of Auxiliaries in Shona". PhD thesis. University of Zimbabwe, 2002 (cit. on pp. 18, 84, 85, 99, 114).

[129] John P McCrae et al. "The Ontolex-Lemon Model: Development and Applications". In: *eLex 2017 Conference*. 2017, pp. 19–21 (cit. on p. 24).

[130] John McCrae, Dennis Spohr, and Philipp Cimiano. "Linking Lexical Resources and Ontologies on the Semantic Web with Lemon". In: *Extended Semantic Web Conference*. Springer. 2011, pp. 245–259 (cit. on pp. 23, 24).

[131] Chris Mellish et al. "A Reference Architecture for Natural Language Generation Systems". In: *Natural Language Engineering* 12.1 (2006), pp. 1–34 (cit. on p. 13).

[132] Dr AG Menon et al. "Amrita morph analyzer and generator for tamil: a rule based approach". In: *Tamil Internet Conference*. Coimbatore, India, 2009, pp. 239–243 (cit. on p. 43).

[133] Marie W Meteer. "Bridging the Generation Gap Between Text Planning and Linguistic Realization". In: *Computational Intelligence* 7.4 (1991), pp. 296–304 (cit. on p. 13).

[134] Guido Minnen, John Carroll, and Darren Pearce. "Robust, Applied Morphological Generation". In: *1st International Conference on Natural Language Generation*. Vol. 14. Association for Computational Linguistics. 2000, pp. 201–208 (cit. on pp. 37, 49).

[135] Saad Mohamood. *default-lexicon.xml*. Accessed on 08/02/2018. 2016. URL: https://github.com/simplenlg/simplenlg/blob/master/src/main/resources/default-lexicon.xml (cit. on p. 123).

[136] Linkie Mohlala. "The Bantu Attribute Noun Class Prefixes and their Suffixal Counterparts, with Special Reference to Zulu". MA thesis. Pretoria, South Africa: University of Pretoria, 2003 (cit. on pp. 5, 15–18, 128).

[137] Paul Molins and Guy Lapalme. "JSrealB: A Bilingual Text Realizer for Web Programming". In: *15th European Workshop on Natural Language Generation (ENLG)*. 2015, pp. 109–111 (cit. on p. 83).

[138] Saadiq Moolla. *Mobile Xhosa*. 2014. URL: http://www.mobilexhosa.org.za/ (cit. on p. 131).

[139] Saadiq Moolla. *Mobile Zulu*. 2014. URL: http://www.mobilezulu.org.za/ (cit. on pp. 129, 131).

[140] Jackson Muhirwe. "Computational Analysis of Kinyarwanda Morphology: The Morphological Alternations". In: *International Journal of Computing and ICT Research* 1.1 (2007), pp. 85–92 (cit. on pp. 18, 84, 85, 99, 114).

[141] Ramin Charles Nakisa and Ulrike Hahn. "Where defaults don't help: the case of the German plural system". In: *18th Annual Conference of the Cognitive Science Society*. San Diego, USA, July 12-15, 1996. 1996, pp. 177–182 (cit. on p. 26).

[142] Andrew Nevins. "Phonologically Conditioned Allomorph Selection". In: *The Blackwell Companion to Phonology* (2011), pp. 1–26 (cit. on pp. 33, 63).

[143] Wanjiku and Nganga. "Swahili Inflectional Morphology for the Grammatical Framework". In: *Human Language Technology for Development*. Alexandria, Egypt, 2011 (cit. on p. 11).

[144] F. Natalya Noy and L. Deborah McGuinness. *Ontology Development 101: A Guide to Developing your First Ontology*. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880. Stanford University, 2001 (cit. on p. 21).

[145] Derek Nurse. "Aspect and Tense in Bantu Languages". In: *The Bantu Languages: Routledge Language Family Series 4*. London: Taylor and Francis Routledge, 2003. Chap. 6, pp. 90–102 (cit. on pp. 18, 19, 39, 40, 124).

[146] Derek Nurse and Gerard Philippson. "Introduction". In: *The Bantu Languages: Routledge Language Family Series 4*. London: Taylor and Francis Routledge, 2003. Chap. 1, pp. 1–9 (cit. on pp. 2, 15–19).

[147] Rita Orji and Karyn Moffatt. "Persuasive Technology for Health and Wellness: State-of-the-art and Emerging Trends". In: *Health Informatics Journal* 24.1 (2018), pp. 66–91 (cit. on p. 24).

[148] K Parameswari. "An Improvized Morphological Analyzer cum Generator for Tamil: A case of implementing the open source platform APERTIUM". In: *Knowledge Sharing Event 1-Central Institute of Indian Languages*. Mysore, India, 2010 (cit. on p. 43).

[149] V. Plachouras et al. "Interacting with Financial Data Using Natural Language". In: *The 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'16)*. Pisa, Italy, 2016, pp. 1121–1124 (cit. on p. 7).

[150] Richard Power, Donia Scott, and Roger Evans. "What You See Is What You Meant: Direct Knowledge Editing with Natural Language Feedback." In: *ECAI*. Vol. 98. 1998, pp. 677–681 (cit. on p. 22).

[151] K. Rajan, V. Ramalingam, and M. Ganesan. "Machine Learning of Phonologically Conditioned Noun Declensions for Tamil Morphological Generators". In: *International Journal of Computer Engineering and Applications* 4 (3 2014) (cit. on p. 43).

[152] A. Ramos-Soto, J. Janeiro-Gallardo, and A. Bugarin. "Adapting SimpleNLG to Spanish". In: *10th International Conference on Natural Language Generation*. Association for Computational Linguistics, 2017, pp. 142–146 (cit. on pp. 12, 82, 83).

[153] A. Ramos-Soto et al. "Linguistic Descriptions for Automatic Generation of Textual Short-term Weather Forecasts on Real Prediction data". In: *IEE Transactions on Fuzzy Systems*. Vol. 22. 2015, pp. 44–57 (cit. on p. 7).

[154] Aarne Ranta. *Grammatical Framework: Programming with Multilingual Grammars*. Stanford: Center for the Study of Language and Information (CSLI) Publications, 2011 (cit. on pp. 11, 82, 123).

[155] Aarne Ranta, Yan Tian, and Haiyan Qiao. "Chinese in the Grammatical Framework: Grammar, Translation, and Other Applications". In: *8th SIGHAN Workshop on Chinese Language Processing (SIGHAN'08)*. Beijing, China, 2015, pp. 100–109 (cit. on pp. 11, 82, 123).

[156] E. Reiter et al. "Choosing Words in Computer Generated Weather Forecasts". In: *Artificial Intelligence* 167.1-2 (2005), pp. 137–169 (cit. on p. 7).

[157] Ehud Reiter. "An architecture for Data-to-text Systems". In: *Eleventh European Workshop on Natural Language Generation*. Association for Computational Linguistics. 2007, pp. 97–104 (cit. on p. 13).

[158] Ehud Reiter. "Natural Language Generation". In: *The Handbook of Computational Linguistics and Natural Language Processing* (2010). Ed. by A. Clark, Z. Fox, and S. Lappin, pp. 574–598 (cit. on p. 12).

[159] Ehud Reiter and Anja Belz. "An Investigation into the Validity of Some Metrics for Automatically Evaluating Natural Language Generation Systems". In: *Computational Linguistcs* 35.4 (2009), pp. 529–558 (cit. on pp. 72, 73).

[160] Ehud Reiter and Robert Dale. "Building applied Natural Language generation systems". In: *Natural Language Engineering* 3.1 (1997), pp. 57–87 (cit. on pp. 1, 2, 7–10, 12).

[161] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge: Cambridge University Press, 2000 (cit. on pp. 7, 8).

[162] Verena Rieser et al. "Adaptive Information Presentation for Spoken Dialogue Systems: Evaluation with Human Subjects". In: *13th European Workshop on Natural Language Generation*. Association for Computational Linguistics. 2011, pp. 102–109 (cit. on p. 14).

[163] Hazem Safwat and Brian Davis. "CNLs for the Semantic Web: A State of the art". In: *Language Resources and Evaluation* 51.1 (2017), pp. 191–220 (cit. on pp. 21, 22).

[164] C. Thilo Schadeberg. "Historical Linguistics". In: *The Bantu Languages: Routledge Language Family Series 4*. London: Taylor and Francis Routledge, 2003. Chap. 9, pp. 143–181 (cit. on p. 19).

[165] c. Thilo Schadeberg. "Derivation". In: *The Bantu Languages: Routledge Language Family Series 4*. London: Taylor and Francis Routledge, 2003. Chap. 5, pp. 71–89 (cit. on pp. 18, 124).

[166] Rolf Schwitter. "Creating and querying linguistically motivated ontologies". In: *Knowledge Representation Ontology Workshop, Conference in Research and Practice in Information Technology*. Vol. 90. Citeseer. Sidney, Australia, 2008, pp. 71–80 (cit. on p. 21).

[167] P Semman. "Natural Language Generation: An Overview". In: *Journal of Computer Science and Research (JCSCR)* 1.3 (2012), pp. 50–57 (cit. on p. 10).

[168] B. R. Shambhavi et al. "Kannada Morphological Analyser and Generator Using Trie". In: *International Journal of Computer Science and Network Security (IJCSNS)* 11 (1 2011) (cit. on p. 43).

[169] M. Steedman. *The Syntactic Process*. Cambridge, Massachusetts, USA: MIT Press, 2000 (cit. on pp. 11, 13).

[170] O. Stock et al. "Adaptive, Intelligent Presentation of Information for the Museum Visitor in PEACH". In: *User Modeling and User-Adapted Interaction* 17.3 (2007), pp. 257–304 (cit. on p. 7).

[171] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3104–3112 (cit. on p. 15).

[172] Elsabé Taljard and Gilles-Maurice de Schryver. "A Corpus-driven Account of the Noun Classes and Genders in Northern Sotho". In: *Southern African Linguistics and Applied Language Studies* 34.2 (2016), pp. 169–185 (cit. on p. 18).

[173] Jian Tang et al. "Context-aware Natural Language Generation with Recurrent Neural Networks". In: *arXiv preprint arXiv:1611.09900* (2016) (cit. on p. 15).

[174] Knut Tarald Taraldsen. "The Nanosyntax of Nguni Noun Class Prefixes and Concords". In: *Lingua* 120.6 (2010), pp. 1522–1548 (cit. on pp. 16–18, 84, 87, 155).

[175] Doreen Daphine Tayebwa. "Demonstrative Determiners in Runyankore-Rukiga". MA thesis. Norway: Norwegian University of Science and Technology, 2014 (cit. on pp. 17, 19, 95).

[176] C. Taylor. *A Simplified Runyankore-Rukiga-English Dictionary*. Kampala, Uganda: Fountain Publishers, 2009 (cit. on pp. 34, 35, 61, 123).

[177] Justus Turamyomwe. "Tense and Aspect in Runyankore-Rukiga: Linguistic Resources and Analysis". MA thesis. Norway: Norwegian University of Science and Technology, 2011 (cit. on pp. 18, 19, 39, 40, 43, 44, 49, 95).

[178] Rafael Valencia-Garcia, Francisco Garcia-Sanchez, and Dagoberto Castellanos-Nieves. "OWL-Path: An OWL Ontology-guided Query Editor". In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 41.1 (2010), pp. 121–136 (cit. on p. 23).

[179] K. Van Deemter, Emiel Krahmer, and M. Theune. "Real vs. Template-based Natural Language Generation: A False Opposition?" In: *Computational Linguistics* 31 (1 2005), pp. 15–23 (cit. on p. 10).

[180] Pierre-Luc Vaudry and Guy Lapalme. "Adapting SimpleNLG for Bilingual English-French Realisation". In: *14th European Workshop on Natural Language Generation (ENLG 2013)*. Sofia, Bulgaria: Association for Computational Linguistics, 2013, pp. 183–187 (cit. on pp. 12, 82, 83).

[181] Jette Viethen and Robert Dale. "The Use of Spatial Relations in Referring Expression Generation". In: *Fifth International Conference on Natural Language Generation (INLG'08)*. Association for Computational Linguistics. 2008, pp. 59–67 (cit. on p. 9).

[182] E. V. Vinu and P. Sreenivasa Kumar. "Ontology Verbalization using Semantic-Refinement". In: *Computing Research Repository (CoRR)* arXiv:1610.09964 (2016) (cit. on p. 21).

[183] R. van der Wal et al. "The Role of Automated Feedback in Training and Retaining Biological Reporters for Citizen Science". In: *Conservation Biology* 30.3 (2016), pp. 550–561 (cit. on p. 7).

[184] L. Wanner et al. "Getting the Environmental Information Across: From the Web to the User". In: *Expert Systems* 32.3 (2015), pp. 405–432 (cit. on p. 7).

[185] M. White and R. Rajkumar. "Minimal Dependency Length in Realization Ranking". In: *Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP'12)*. Jeju, South Korea, 2012, pp. 244–255 (cit. on pp. 2, 11).

[186] Lauren Wilcox et al. "Characterizing patient-friendly micro-explanationsof medical events". In: *SIGCHI Conference on Human Factors in Computing Systems*. ACM. Vancouver, Canada: ACM, 2011, pp. 29–32 (cit. on pp. 2, 24).

[187] Ilana Wilken, Febe de Wet, and Elsabe Toljard. "A Mobile Vocabulary Acquisition Application for Health Science Students: A Proposed Study". In: *Language Teaching, Learning ISCA Archive, and Technology (LTLT 2015)*. Leipzig, Germany, 2015, pp. 20–24 (cit. on pp. 1, 131).

[188] Zhiwu Xu, Lixiao Zheng, and Haiming Zhen. "A Toolkit for Generating Sentences from Context-Free Grammars". In: *International Journal of Software and Informatics* 5 (4 2011), pp. 659–676 (cit. on pp. 43, 47, 49, 69, 127, 130).

[189] Sina Zarrieß and Jonas Kuhn. "Combining Referring Expression Generation and Surface Realization: A Corpus-based Investigation of Architectures". In: *51st Annual Meeting of the Association for Computational Linguistics (Long Papers)*. Vol. 1. 2013, pp. 1547–1557 (cit. on pp. 14, 15).

[190] Jason Zentz. "Forming Wh-Questions in Shona: A Comparative Bantu Perspective". PhD thesis. Yale University, 2016 (cit. on pp. 16, 17, 84, 85, 154).

Appendix A

Evaluation of CFG Output

Please assess the correctness of the conjugated verbs below, according to their grammatical correctness. Place an asterix (*) in front of the one you consider to be incorrect. If you deem it necessary to explain your decision, you can type that in the space below the associated word.

nagyenda
nikagyenda
nimugyenda
ningyenda
takugyenda
tibakugyenda
tekugyenda
tinkugyenda
tizikugyenda
biine
eine
gwine
nyine
oine
tayine
tibaine
timwine
tinyine
nibavuga
ninvuga
novuga
takuvuga
tinkuvuga
titukuvuga
nibivugwa
nevugwa
ninvugwa
novugwa
nimushoma
ninshoma
noshoma
takushoma
tibakushoma
tinkushoma
titukushoma
biri
buri
eri
ndi
ori
ziri
tari
tiburi

Appendix B

Controlled Natural Language (CNL) Statements

(1) Fennodil is a well proven drug

(2) Fennodil and Nozydril

(3) Each Fennodil has no known side effects

(4) Each Nozydril is not addictive

(5) All Fennodil reduce some stomach acid

(6) All Nozydril relieve some painful symptoms

(7) All Fennodil are only taken with milk

(8) Each tablet has exactly 20 mg of Flavotine

(9) All tablets have some very high success rate

(10) All tablets only act directly on ulcers

(11) All Fennodil cause some nausea

(12) Each Nozydril is not taken with some alcohol

(13) All tablets have som adverse reaction with other prescribed medication

(14) All Fennodil act within a maximum of a week

(15) All tablets are taken at least 8 hours apart

Appendix C

Ethics Approval

UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Faculty of Science
University of Cape Town
RONDEBOSCH 7701 South Africa
E-mail: timm.hoffman@uct.ac.za
Telephone: + 27 21 650 5551

25 January 2017

Ms. Joan Byamugisha
Department of Computer Science

A Grammar Engine to Verbalize Ontologies in Runyankore

Dear Ms. Joan Byamugisha

I am pleased to inform you that the Faculty of Science Research Ethics Committee has approved the above-named application for research ethics clearance, subject to the conditions listed below.

- Implement the measures described in your application to ensure that the process of your research is ethically sound; and
- Uphold ethical principles throughout all stages of the research, responding appropriately to unanticipated issues: please contact me if you need advice on ethical issues that arise.

Your approval code is: **FSREC 079** – 2016

I wish you success in your research.

Yours sincerely

signature removed to avoid exposure online

Prof Timm Hoffman
Chair: Faculty of Science Research Ethics Committee

Cc: Dr Maria Keet & Dr Brian deRenzi(Supervisors)

Appendix D

Translated Questionnaire for Non-Linguists

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF CAPE TOWN
PRIVATE BAG X3
RONDEBOSCH 7701
SOUTH AFRICA

RESEARCHER'S TELEPHONE: +256 752702714
E-MAIL: joan.byamugisha@gmail.com, jbyamugisha@cs.uct.ac.za

Okwikiriza Okw'okwekundira kwejumba omu kucondooza
Omutwe gw'okucondooza: Okwejunisa kompyuta kukora ebibazo by'Orunyankore

Okushaba ku oyejumba omu kucondooza n'ebirugi ebi kurimu

Nooshabwa kwejumba omu kucondooza okurikuza kubamu abantu abarikumanya kushoma, kuhandiika hamwe n'okugamba Orunyankore. Ekigyendererwa ky'okucondooza oku n'okwenda kumanya ebibazo ebihairwe omu Runyankore byaba bihikire nainga birikukuratira ebiragiro by'enyombeka y'ebibazo nainga shi guraama y'Orunyankore hamwe n'okumanya yaaba nibyetegyerezibwa. Nyine okwikiriza ngu ahabw'obukugu obu oine noija kutuha amakuru agaraije kuba ag'omugasho omu kucondooza oku kandi nyine amasiko ku naiwe noija kugira ebi waayega omu kucondooza oku.

Okucondooza oku kuraatwazibwe
Omu kucondooza oku, noija kubuuzibwa ebi:

a. Okuha enteekateeka yaawe aha muringo, enyombeka y'ebibazo by'Orunyankore ebikwatiraine n'omubazi oguragiriirwe omushaho ahabw'okujanjaba oburwaire;
b. kushwijuma nainga shi kwetegyereza ebibazo ebikozirwe byaba nibikuratira enyombeka y'ebibazo nainga shi guraama y'Orunyankore;
c. kushwijuma yaaba ebibazo ebihairwe nibyetegyerezibwa; hamwe
d. n'okutaanisa ebibazo ebikozirwe abantu hamwe n'ebyo ebihangahangirwe kuruga omu bibazo ebikuhairwe, kandi oshoboorore okusharamu kwawe

Ebintu ebitaboneire ebirikubaasa kurugiirira aha kucondooza oku:

Noobaasa kuteekateeka ngu okushwijuma nikwija kuba nikukwata aha ku orikwetegyereza Orunyankore. Eki tikwo kiri. Kwihaho okushwijuma kuri aha nyombeka y'ebibazo ebikozirwe kandi niyo erikwija kwetegyerezibwa.

Ku waakwenda kuruga omu kucondooza oku:

Okwejumba kwawe omu kucondooza oku n'okw'okwekundira. Noobaasa kwanga kwejumba omu kucondooza oku, kandi nabwo noobaasa kukurugamu eshaaha yoona otabandize kuha enshonga yoona ahabwenki waakurugamu, kandi tihaine oraije kukuteekateekaho kubi nainga kukutwariza kubi waaba oteejumbire omu kucondooza oku nainga waakurugamu. Ku waakusharamu kuruga omu kucondooza oku, orikukora okucondooza tarikwija kukoresa amakuru goona agu akwihireho ataine rusa kuruga ahari iwe, oru omuhaire obwo otaireho

FIGURE D.1: Translated consent form and questionnaire given to non-linguists

Appendix E

English Version of Questionnaire for Non-Linguists

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF CAPE TOWN
PRIVATE BAG X3
RONDEBOSCH 7701
SOUTH AFRICA" "

RESEARCHER'S TELEPHONE: +256 752702714
E-MAIL: joan.byamugisha@gmail.com, jbyamugisha@cs.uct.ac.za

Informed Voluntary Consent to Participate in Research Study

Project Title: A Grammar Engine to Verbalize Ontologies in Runyankore

Invitation to participate, and benefits:

You are invited to participate in a research study conducted with persons who can read, write, and speak Runyankore. The study aim is to assess the grammatical correctness and understandability of the generated Runyankore text. I believe that your experience would be a valuable source of information, and hope that by participating you may gain useful knowledge.

Procedures:
During this study, you will be asked to:
a. Give opinions regarding the nature and structure of Runyankore messages about prescribed medication;
b. Assess the grammatical correctness of the generated text;
c. Assess the understandability of the generated text; and
d. Distinguish between human-authored and generated text, from among the provided sentences, and explain your decision

Risks:
There may be a misconception that your understanding of Runyankore is what is being assessed. This is not the case; rather the structure of the generated text is what is being assessed.

Disclaimer/Withdrawal:
Your participation is completely voluntary; you may refuse to participate, and you may withdraw at any time without having to state a reason and without any prejudice or penalty against you. Should you choose to withdraw, the researcher commits not to use any of the information you have provided without your signed consent. Note that the researcher may also withdraw you from the study at any time.

Confidentiality:
All information collected in this study will be kept private in that you will not be identified by name or by affiliation to an institution. Confidentiality and anonymity will be maintained as pseudonyms will be used.

What signing this form means:

FIGURE E.1: The original English consent form and questionnaire intended for non-linguists

Appendix F

Noun Class Systems for Five Agglutinating Bantu Languages

TABLE F.1: The chiShona noun class system, showing the subject, adjective, and possessive concords [190] [190]

No	Class Prefix	Grammatical Form	Subject Concord	Possessive Concord	Adjective Concord
1	mu-	singular	-a-	wa-	mu-
1a-	–	singular	-a-	wa-	mu-
2	va-	plural	-va-	va-	va-
2a-	vana-	–	-va-	va-	va-
2b	a–	singular	-va-	va-	a-
3	mu-	singular	-u-	wa-	mu-
4	mi-	plural	-i-	ya-	mi-
5	–	singular	-ri-	ra-	ri-
6	ma-	plural	-a-	a-	ma-
7	chi-	singular	-chi-	cha-	chi-
8	zvi-	plural	-zvi-	zva-	zvi-
9	n-	singular	-i-	ya-	n-
10	n-/dzi-	plural	-dzi-	dza-	n-
11	ru-	singular	-ru-	rwa-	ru-
12	ka-	singular	-ka-	ka-	ka-
13	tu-	plural	-tu-	twa-	tu-
14	u-	singular	-hu-	hwa-	u-
15	ku-	singular	-ku-	kwa-	ku-
16	ha-	locative	-ha-	ha-	ha-
17	ku-	locative	-ku-	kwa-	ku-
18	mu-	locative	-mu-	mwa-	mu-
21	zi-	singular	-ri-	ra-	zi-

TABLE F.2: The isiXhosa noun class system, showing the subject, adjective, and possessive concords [174]

No	Class Prefix	Grammatical Form	Subject Concord	Possessive Concord	Adjective Concord
1	u-m-	singular	-u-	wa-	m-
1a-	u-	singular	-u-	wa-	m-
2	a-ba-	plural	-ba-	ba-	ba-
2a-	oo-	plural	-ba-	ba-	ba-
3	u-m-	singular	-u-	wa-	m-
4	i-mi-	plural	-i-	ya-	mi-
5	i-/i-li-	singular	-li-	la-	li-
6	a-ma-	plural	-a-	a-	ma-
7	i-si-/i-s-	singular	-si-	sa-	si-
8	i-zi-/i-z-	plural	-zi-	za-	zi-
9	i-/i-n-	singular	-i-	ya-	i-n-
10	ii-/i-zin-	plural	-zi-	za-	zin-
11	u-/u-lu-	singular	-lu-	lwa-	lu-
14	u-bu-	singular	-bu-	ba-	bu-
15	u-ku-	singular	-ku-	kwa-	ku-

TABLE F.3: The Kikuyu noun class system, showing the subject, adjective, and possessive concords [88]

No	Noun Class	Grammatical Form	Subject Concord	Subject Prefix	Possessive Concord	Adjective Concord
1	mu-	singular	-a-	u-	wa-	mu-
1a-	–	singular	-a-	u-	wa-	mu-
2	a-	plural	-ma-	a-	a-	a-
2a-	–	plural	-ma-	a-	a-	a-
3	mu-	singular	-u-	u-	wa-	mu-
4	mi-	plural	-i-	i-	ya-	mi-
5	ri-/i-	singular	-ri-	ri-/-i-	ria-	ri-/i-
6	ma-	plural	-ma-	ma-	ma-	ma
7	ki-/gi-	singular	-ki-	ki-/-gi-	kia-	ki-/gi-
8	ci-/i-	plural	-ci-/-i-	ci-/-i-	cia-	n-
9	n-	singular	-i-	i-	ya-	n-
9a-	–	singular	-i-	i-	ya-	n-
10	n-	plural	-ci-	ci-	cia-	n-
10a-	–	plural	-ci-	ci-	cia-	n-
11	ru-	singular	-ru-	ru-	rua-	ru-
12	ka-/ga-	singular	-ka-	ka-/-ga-	ka-	ka-/ga-
13	tu-	plural	-tu-	tu-	tua-	tu-
14	–	singular	-u-	u-	wa-	mu-
15	ku-/gu-	singular	-ku-	ku-/-gu-	kua-	ku-/gu-
16	ha-	locative	-ha-	ha-	ha-	ha-
17	ku-/gu-	locative	-ku-	ku-/-gu-	kua-	ku-/gu-

TABLE F.4: The Kinyarwanda noun class system, showing the subject, adjective, and possessive concords [101]

No	Noun Class	Grammatical Form	Subject Concord	Possessive Concord	Adjective Concord
1	u-mu- singular	-a-	wa-	u-mu-	
2	a-ba-	plural	-ba-	ba-	ba-
3	u-mu-	singular	-u-	wa-	mu-
4	i-mi-	plural	-i-	ya-	mi-
5	i-, i-ri-	singular	-ri-	rya-	ri-
6	a-ma-	plural	-a-	ya-	ma-
7	i-ki-/i-cy-/i-gi-	singular	-ki-	cya-	ki-
8	i-bi-	plural	-bi-	bya-	bi-
9	i-/i-n-/i-nz-	singular	-i-	ya-	n-
10	i-/i-n-/i-nz-	plural	-zi-	za-	n-
11	u-ru-	singular	-ru-	rwa-	ru-
12	a-ka-/a-ga-	singular	-ka-	ka-	ka-
13	u-tu-/u-du-	plural	-tu-	twa-	tu-
14	u-bu-	plural	-bu-	bwa-	bu-
15	u-ku-/u-gu-	infinitives	-ku-	kwa-	ku-
16	a-ha-	locative	-ha-	ha-	ha-

TABLE F.5: Luganda noun class system, showing the subject, adjective, and possessive concords [11]

No	Noun Class	Grammatical Form	Subject Concord	Possessive Concord	Adjective Concord
1	o-mu-	singular	-a-	w-	o-mu-
2	a-ba-	plural	-ba-	b-	a-ba
3	u-mu-	singular	-gu-	gw-	u-mu-
4	e-mi-	plural	-gi-	gy-	e-mi-
5	e-li-/e-	singular	-li-	ly-	e-li-
6	a-ma-	plural	-ga-	g-	a-ma-
7	e-ki-	singular	-kii-	ky-	e-ki-
8	e-bi-	plural	-bi-	by-	e-bi-
9	e-n-	singular	-e-	y-	e-n-
10	e-n-	plural	-zi-	z-	e-n-
11	o-lu-	singular	-lu-	lw-	o-lu-
12	a-ka-	singular	-ka-	k-	a-ka-
13	o-tu-	plural	-tu-	tw-	o-tu-
14	o-bu-	plural	-bu-	bw-	o-bu-
15	o-ku-	singular	-ku-	kw-	o-ku-
16	wa-	locative	-wa-	wa-	w–
17	ku-	locative	-ku-	kw-	ku-
18	mu-	locative	-mu-	mw-	mu-
20	o-gu-	singular	-gu-	gw-	o-gu-
21	a-ga-	plural	-ga-	g-	a-ga-
23	e-	locative	-e-		–

Appendix G

Original 101 Wordlist Used as SetC

(1) Tree

(2) Plant

(3) Elephant

(4) Meal

(5) Meat

(6) Herbivore

(7) Wine

(8) Grape

(9) Region

(10) Bottle

(11) University

(12) Lecturer/proffessor

(13) Undergraduate student

(14) Course

(15) Doctor

(16) Dog

(17) Cat food

(18) Bird

(19) Owner

(20) Pet

(21) Toy

(22) Building

(23) Room

(24) Wall

(25) Cottage

(26) Window

(27) King

(28) President

(29) Council

(30) Committee

(31) Parliament

(32) Representative

(33) Mediator

(34) Nurse

(35) Operation

(36) Medicine

(37) Pill

(38) Prescription

(39) Diagnosis

(40) Symptom

(41) Stomach

(42) Leg

(43) Eye

(44) Camping

(45) Camera

(46) Hotel

(47) Beach

(48) Holiday

(49) Ice cream

(50) Swimming pool

(51) Mayonnaise

(52) Water

(53) Orange (the fruit)

(54) Green (the colour)

(55) Potato

(56) Key

(57) Airplane

(58) Car

(59) Money

(60) Police

(61) Tea

(62) Flood

(63) Ditch

(64) Field (in the sense of veld/grassfield)

(65) Mountain

(66) River

(67) Sea

(68) Fish

(69) Land

(70) Country

(71) Raspberry

(72) Samp

(73) Chicken

(74) Cafe

(75) Book

(76) Ball

(77) Worker

(78) Company (the one one works for)

(79) Society

(80) Box

(81) Pot (for cooking)

(82) Computer

(83) Pipe

(84) Pie

(85) News

(86) Speaker (the person)

(87) Shoe

(88) Sock

(89) Flower

(90) Walk

(91) Beauty

(92) Lamp

(93) Street

(94) Bag

(95) Niece

(96) Decoration

(97) Cigarette

(98) Painting

(99) Stick

(100) Guardian

(101) Blanket