Computational Analysis of Kinyarwanda Morphology: The Morphological Alternations.

JACKSON MUHIRWE*
Department of Computer Science
Faculty of Computing and IT,
Makerere University

For more than 30 years, there have been renewed interests in computational morphology resulting in numerous morphological tools. However the interest has always been on the politically and economically interesting languages of the world resulting in a wide language divide between the technologically rich and poor languages. Kinyarwanda language, a Bantu language spoken in East Africa is one of those under-resourced languages without any language technology tools. The two most essential components of most natural language applications are a morphological analyzer and a machine-readable lexicon. These two components are still lacking for Kinyarwanda and so many other under-resourced languages. The task of developing a morphological analyzer involves two problems: the morphotactics (word formation) and the morphological alternations. In this paper we are mainly concerned with the morphological alternations.

Categories and Subject Descriptors: J.5 [**Computer Applications**]: Arts and humanities -- language translation, and linguistics; I.2.7 **[Computing Methodologies**] Natural language processing -- Language generation, Language models, Language parsing and understanding, text analysis*;* I.2.1 [**Computing Methodologies**]: Applications and Expert systems – Natural Language Interfaces.

General Terms: Bantu, Kinyarwanda, natural language processing, computational linguistics, computational morphology

Additional Keywords and Phrases: Kinyarwanda morphological analysis, morphological alternations.
IJCIR Reference Format:
Jackson Muhirwe. Computational Analysis of Kinyarwanda Morphology: The Morphological Alternations.. International Journal of Computing and ICT Research, Vol. 1, No. 1, pp. 85 - 92
http:www.ijcir.org/volume1-number1/article10.pdf.

1. INTRODUCTION

The broad area of computational morphology is concerned with computational analysis and synthesis of words for eventual use in natural language applications. The ultimate goal of computational morphology is incorporating its products, the morphological analyzers into higher-level natural language applications. Such applications may include spell checkers and information retrieval systems. Currently there are two broad approaches to computation morphology: rule-based and data-based approaches. Rule-based approaches typically involve the use of grammatical rules to construct a morphological analyzer. On the other hand the data-based approaches use statistical information to develop a morphological analyzer. For this to happen, statistical approaches which are also known as memory based approaches require a large

* Author's Address: Jackson Mmuhirwe, Department of Computer Science, Faculty of Computing and IT, Makerere University, P.O Box 7062, Kampala, Uganda, *jmuhirwe@cit.mak.ac.ug*

International Journal of Computing and ICT Research, ISSN 1818-1139, Vol.1, No.1, pp. 85 - 92, June 2007

corpus. Due to the un availability for such vast resources for Kinyarwanda we chose the grammar based approaches using finite state networks. We chose the rule-based approach also because it gives an insight into the kind of grammatical rules one encounters while using Kinyarwanda. Past research has also shown that rule based approaches are superior to data based approaches to construction of morphological analyzers [Chanod and Tapanainen 1994].

Finite state methods have dominated computational morphology research since Johnson's ground breaking 1972 discovery that under certain constraints, phonological rules could be modeled using finite state methods [Johnson 1972]. Although Johnson [1972] was the first to realise that finite state machines could be used to model linguistics structures, the approach was never recognised until the late 1970s when Kay independently discovered that finite state machines can simplify the modelling of phonology and morphology. The invention of the two level phonology approach in 1983 by Koskenniemi was a major breakthrough in computational linguistics and this led to wide spread use of finite state machines in development of morphological analysers. Since the invention of the two level approach in 1983, finite state methods have successfully been employed to develop morphological analysers for many languages including English, French, Germany, Arabic, Spanish, Basque, Japanese, Korean and Swahili a Bantu language [Beesley and Karttunen 2003]. From the preceding list we see that computational morphological tools have mainly been built for the politically and economically interesting languages of Europe and some parts of Asia. The status quo according to Cole [1997] is that morphological analysers should be built for all but the commercially important languages. To emphasize this point, u-Dhonnchadha et al.[2003] reiterates the need to develop morphological analyser for all languages to avoid creating a language divide. We now have the technologically rich languages and poor languages. There are two main challenges involved in developing morphological analysers: the morphatactics and the morphological alternations. The morphotactics or sometimes known as the morphosyntax, is concerned with the strict rules that dictate how morphemes [smallest components of a word with a purpose or a meaning] are combined together to form words. Morphological alternations are concerned with the phonological / orthographical rules that are required to derive the surface representation of a word from its underlying representation. The problem of morphological alternations arises because sometimes a morpheme may have different realisations depending on its phonological environment and other morphemes that make up the word.
Focus in this paper, is on the phonological / orthographical rules that are required to derive the surface form of a word from its underlying /lexical level form.

2. KINYARWANDA MORPHOLOGY

Kinyarwanda, the national language of Rwanda, is a typical Bantu language classified by Guthrie [1971] as a group D60 Bantu language [D66], together with Kirundi, Giha, Vinza, Hangaza and Shubi. The language is spoken by over 20 million people, living mainly in the great lakes region of East and Central Africa. Its speakers include, Barundi (from Burundi), Giha, Bafumbira, Banyamulenge and the ethnic Banyarwanda in Masisi and Rutshuro in Northern Kivu of the Democratic Republic of Congo. Kinyarwanda major word categories are the nouns and verbs. Kinyarwanda nouns are composed of the preprefix, the prefix or sometimes called the class marker, the stem and the suffix. The noun morphology is mainly influenced by the noun classification system and the ensuing concodia agreement system. Kinyarwanda shares these properties with other Bantu languages. Kinyarwanda verb morphology is known to be more complex than the nominal. Typically, a verb will have multiple prefixes and suffixes surrounding the stem just like beads on a string. Prefixes have only grammatical information while suffixes have both grammatical and lexical information. The only obligatory morphemes in a verb are the subject agreement prefix, the stem and the final vowel which in most cases is the aspect marker. The optional morphemes include the proclitics *-nti* '*not*', *ni* 'if/when'; the tense aspect-modality morphemes; the morphemes *na* 'also'; the object pronouns which can be one or many; lexical verb extensions; grammatical suffixes; the enclitics *-ga;* and locative postclitics *-m´o, -h´o, or -y´o*. Kinyarwanda verbs, like most other Bantu language verbs, can have multiple object pronouns, multiple lexical verbal extensions and multiple grammatical suffixes. Lexical extensions such as *-agur-, -iir-, uur, -aang, iriz-*, etc. add lexical information such as inchoativity, iterativity, repetitivity, intensity, frequentativity, reversivity, etc. Grammatical morphemes such as the causative morpheme *-iish-*, the applicative morpheme *-ir-*, and the comitative/reciprocal morpheme *-an-*, can be added to any verb stem.

3. COMPUTATIONAL MORPHOLOGY

Computational morphology has been an active area of research for the last 25 years. Oflazer [1999] defined computational morphology as the study of the computational analysis and synthesis of words to be eventually used in natural language processing applications. Computational morphology is mainly concerned with systems that efficiently analyse and synthesize words. Although the beginning of the field of computational morphology could be traced to the 1950s, no practical systematic approach was in place till the ealy 1980s. The two-level morphology approach introduced by Koskenniemi in 1983 was the first practical approach to be developed. This approach was immediately accepted by most researchers and since then has been the dominant formalism for dealing with computational morphology [Sproat 1992 2006; Antworth 1990].

3.1 Finite State Morphology

The two main challenges in morphology is word formation (morphotactics) and morphological alternations. Word formation is the study about the principles that govern the combination of stems, affixes and other morphemes to form acceptable words for a given language. For example in English piti-less-ness is an acceptable word but piti-ness-less is not. Although in most languages, words are built by concatenation of morphemes there are also many languages that build words using non-concatenative processes such as interdigitation and reduplication.

The challenge of morphological alternations arises due to the fact that a morpheme may have different alternations depending on its phonological environment and the composition of the word. For example pity is realised as piti in the context of less, and die is realised as dy in dying.

Finite State Morphology is an attempt to account for these problems within the context of regular sets and regular relations. Although the two problems are related, the claim that one may be solved by finite state methods does not automatically extend to the other. Here we are mainly concerned with morphological alternations as they are manifested in Kinyarwanda.

3.1.1 Regular Expressions

Regular expressions are a formal language similar to Boolean logic formulae. Regular expressions denote sets and they have a simple syntax. These sets can be divided into two sets of strings (language) and sets consisting of pair of strings (relation).

The terms regular language and relation refer to sets that can be described by a regular expressions. Regular languages and relations are encoded into finite state networks with languages representing finite state automata and relation represented by transducers. Regular expressions contain two kinds of symbols: unary a, b, c etc.. and symbol pairs a:b, b:c, c:d, etc... Unary strings such as a,b,c etc.. denote strings while the symbol pairs denote relations.

A regular relation may always be viewed as a mapping between two regular languages. The a:b relation is a cross product of languages denoted by expressions a and b. In the a:b relation, the a is referred to as the upper language and b as the lower language of the relation. For simplicity purposes, the identity relation A:A is simply written as A. A transducer that encodes a regular relation may be used to map a string in the upper language to a string in the lower language and vice versa. There is no privileged direction.

Complex regular expressions can be constructed out of the simple one by means of regular expression operators. This is made possible because of the closure properties. Regular languages are known to be closed under concatenation, intersection, union, complementation and Kleene start. Details about these closure properties and the proofs are outside the scope of this paper. The interested reader is advised to refer to any book on automata. The syntax (though not the descriptive power) of regular expressions can be extended by defining new operators that allow commonly used expressions to be expressed in a concise way. These extensions have made very easy to apply regular expressions to language engineering.
For example, consider Koskenniemi [1983] restriction operator => originally introduced to for two-level phonological rules as we shall see in the next section.

A=> B_C The restriction of A only in the context of B_C

Here A, B, C denote regular languages. The above expression represents the language of strings that have a property that every occurrence of a string of A must be preceded by a string of B and followed by a string

of C. The advantage of the restriction operator is that it encodes in a very compact way a useful property that would otherwise be difficult to encode using the primitive operators. Using the primitive formulas the restriction operators is represented below.

A=>B_C = [~ [[~ [?* B] A ?*] | [?*A ~[C ?*]]]].

In the above expression, ? represents any symbol, * is the Kleene start, ~ is the complementation operator and [ ] are used for only grouping purposes.
Clearly, we can see from above that high level abstractions like (A=>B_C) easier to deal with that the equivalent but complex primitive formulas.

3.2 Two level Morphology

The two-level morphology approach to morphological analysis is a language independent general formalism for analysis and generation of word-forms. Kimmo invented this approach in 1983 [Koskenniemi 1983a]. Prior to the two-level approach was the Noam Chomsky' generative phonology approach. The generative phonology approach has two major weaknesses. It creates un-necessary intermediate levels and is also uni-directional. Kimmo decided to eliminate the intermediate levels. This created a new approach, which has only two levels, the lexical level and the surface level, hence the name Two-Level Morphology. This model has also an added advantage of being bi-directional, implying that both analysis and generation could be done using the same system, which was not possible with the earlier approaches which were uni-directional. Two-level morphology depends heavily on finite state methods, which are well known and are often described as elegant [Kartunnen and Bessley 2003]. Several compilers have been developed to deal with two level rules, but in this paper we shall use Twolc developed by Xerox. The choice for two-level morphology approach was not accidental. The two level approach has already successfully been used to develop a comprehensive morphological analyser for Swahili, a Bantu language.

3.2.1 Two level rules

Two level rules are generally of the form CP OP LC _ RC where
CP stands for correspondence part, OP stands for Operator and LC stands for Left Context
RC stands for Right Context.
There are four different kinds of rules that may be used to describe morphological alternations of any language.

1. a:b => LC _ RC. This rule states that lexical //a// can be realized as surface b ONLY in the given context. This rule is a context restriction rule.
2. a:b <= LC _ RC This rule states that lexical //a// has to be realized as surface b ALWAYS in the given context. This rule is a surface coercion rule.
3. a:b ⇔ LC _ RC this is a composite rule which states that lexical //a// is realized as surface be ALWAYS and ONLY in the given context.
4. a:b /<= LC _ RC This is an exclusion rule that states that lexical //a// is never realized as surface //b// in the given context.

These rules are extensions of regular expressions and they may be compiled into finite state acceptors either by hand or automatically using one of the many available two level rule compilers. For the purpose we used the Xerox finite state tools for testing the morphological alternation rules.

3.3 The Xerox Finite State Tools

This is a set of powerful, sophisticated set of algorithms and programming languages for building finite state solutions to a variety of problems in natural language processing. For the purpose of this paper we shall only look at only one tool: The two-level compiler.

3.3.1 Twolc

This is the two-level compiler. TWOLC is a high level declarative language designed for specifying alternation rules required in morphological descriptions. Gross irregularities and all base forms are also included in the Lexc source file. TWOLC source files are typically

text files written using notepad, Emacs or any other text editor. The Xerox finite state technology is based on three fundamental insights [Beesley and Karttunen 2003]:

- The morphotacts can be encoded using finite state networks;
- The alternation rules of each morpheme can be implemented as a finite state transducer;
- The lexicon network and the rule transducer can be combined together by composition to form a single network called a lexical transducer.

Lexical transducers constructed using the Xerox finite state technology is mathematically elegant, bi-directional and highly efficient. Lexical transducer applications have a potential for wide lexical coverage, use little memory space, being robust and commercially viable products.

4. MODELING KINYARWANDA MORPHOLOGICAL ALTERNATIONS

Kinyarwanda uses 26 alphabets (a b c d e f g h i j k l m n o p pf r s t ts u v y w z) in its current orthography. There are five vowels, a, e, I ,o u and the rest are consonants.
For the purpose of the two-level description we need to define the alphabet and other needed subsets using the Twolc syntax.
According to the twolc syntax definition of the alphabet follows the keyword alphabet and the required sets also proceed the keyword set. The exclamation mark '!' is used for commenting.

Alphabet: a b c d e f g h i j k l m N:m N:n n o p F:pf q r s t S:ts u v x y w z ;

There may also be need for definitions of sets that may be required by the rules

Sets
Vowels: V = a e i o u ;
Consonants: C = b c d g h j k l m n p q r s t v w x y z ;
Consonants excluding w: C2 = b c d g h j k l m n p q r s t v x y z ;
Consonants excluding n: C3 = b c d g h j k l m p q s t v w x y z ;
Round vowels: VR = e o;
Front vowels: VFR = i e;
Back vowels: VBK = o u;
Africates: A = f s;
Voiceless Consonants VS = c h k p s t ;

The keyword Rules marks the beginning of all the alternation rules in the source file.
According to the twolc two-level rules syntax the first part of the rule is the rule name and the other part is the rule itself. The rule name is written in double quotation marks and has to be unique for a given rule file. A rule is incomplete without the name.

4.1 Deletion rules:

Deletion rule concerns the deletion of –vowels, consonants and even syllables
//a// is deleted before any vowel in the initial position of a morpheme. Other vowels are glided as we shall see later.
Example
Aba-aana → *abaana*
Ama-ooko → *amooko*
Baa-uubatse → *buubatse*

4.2 !Deletion rules

"(1) a deletion"
a:0 <=> C _ V ;
"(2) y deletion"
y:0 <=> [y|w|z:j|g:z|d:z] _ ;
"(3) w deletion"
w:0 <=> w: _ ;

"(4) r deletion"
r:0 <= C3 _ [y [e|i]] ;
"(5) n deletion"
n:0 <=> _ [n|m] ;
"(6) u deletion" !This rule has been created due to rule 21
u:0 <=> :o _ C ;
"(7) k deletion"
k:0 <=> _ y:S ;
"(8) t deletion"
t:0 <=> _ y:s ;

4.3 Glide formation rule

his rule is about the formation of the glide. For example //i// becomes //y//
or //a// becomes //y// when if it is alone with a morpheme (This is about morpheme -a-
//u// becomes //w// if is alone on the morpheme.
i-angaaza → iyangaaza
u-wu-iishe → uwiiyishe
u-i-ang-a → wiiyaanga

"(9) a becomes y"
a:y <=> .#. _ V ?;
"(10) i becomes y"
i:y <=> _ [a|e|o|u];
"(11) u becomes w" !We need to deal with situations where u is following i e.g u-i-anga
u:w <=> [.#.|C2] _ V;
"(12) o becomes w"
o:w <=> C _ [a|e|i|u] ;
"(13) n becomes m"
n:m <=> _ [b|f|p|v|:p|:f|:v] ;
"(14) r becomes d"
r:d <= n _ ;
"(15) pf becomes f"
F:f <=> n _ ;
"(16) ts becomes s"
S:s <=> n _ ;
"(17) voiceless becoming voiced"
Ck:Cg <=> _ [V VS];
where Ck in (k t)
Cg in (g d)
matched;
"(18) h becomes p"
h:p <=> n:m _ ;
"(19) k becomes c"
k:c <=> _ :y ;

4.4 !Assimilation rules

"(20) i becomes e" ! Here a combines with the i to form e
i:e <=> a:0 _ C;
"(21) u becomes o" !a combines with u to form o
u:o <=> a:0 _ C ;
"(22) Y becomes h" ! S combines with y to form h
y:h <=> s _ ;
"(23) e becomes o" !When e of se unites with u they become o
e:o <=> s _ u:0;

```
"(24) y becomes ts" ! Here k combines with y to give birth to ts
y:S <=> k: _ ;
"(25) y becomes s"
y:s <=> t:0 _ ;
"(26) z becomes j" !In this rule z followed by y becomes j
z:j <=> _ y: ;
"(27) d and g become z"
g:z <=> _ y:0 ;
"(28) d and g become z"
d:z <=> _ y:0 ;
"(29) r becomes z"
r:z <= C3 _ y:0 ;
"(30) w and y position change"
w:y <=> C2 _ y:w ;
"(31) w and y position change"
y:w <=> w:y _ ;
```

5. DISCUSSION AND CONCLUSION

In this paper we have presented Kinyarwanda phonological /orthographical rules which form one of the basic two components required to develop a finite state two level morphology based morphological analyser. Some of the rules we have presented in this paper apply also to other Bantu languages, for example Dahl's law is applies to all Bantu languages. Since most rules are language dependant, the rules presented in this paper may not directly be applied to other Bantu languages. What can be reused is the idea and the formats used. Since the introduction of replace rules which look like the traditional rewrite rules, there has been debates on whether to use two level rules or the replace rules. Proponents of the two level rules would argue that the rewrite rules are outdated and should not be used today. On the other hand supporters of the replace rules, prefer them because of the close resemblance to traditional rewrites rules of Chomsky and Halle [1968]. Mathematically speaking both rules are the same. Mathematically, a network produced in one way is equally as good as finite state network produced in a different way. So the choice of two level rules or replace rules eventually is a matter of choice and human ease of use [Beesley and Karttunen 2003]. Two level rules where also chosen because they have already successfully been used to develop a comprehensive morphological analyser for Swahili, a Bantu language. Xerox tools are fast, elegant, well documented and modular and have been experimented on other Bantu languages. This is why we decided to use Xerox finite state tools.

The rules presented in this paper have been tested thoroughly well and are now part of a running Kinyarwanda morphological analyser. Over 90% of Bantu languages are known to be tonal languages therefore future work will go into incorporating tone rules into these rules as we apply them to the lexicon for a comprehensive analysis of Kinyarwanda and other Bantu languages.

REFERENCES

ANTWORTH, EVAN, L. 1990. PC-KIMMO: a two-level processor for morphological analysis. No. 16 in Occasional publications in academic computing. Dallas: Summer Institute of Linguistics.

BEESLEY, K. AND KARTTUNEN LAURI. 2003. Finite State Morphology: CSLI Studies in Computational Linguistics. Stanford University, CA: CSLI Publications.

CHANOD, J.P. &TAPANAINEN, P.(1994. "Tagging French:Comparing a Statistical and a Constraint-based Method." *Seventh Conference of the European Chapter of the Association for Computational Linguistics,* 149-156.

Chomsky, N., and Halle M. 1968. The sound Pattern of English. New York: Harper and Row. xiv, 470 pages. Reprinted 1991, Boston: MIT press.

U´I DHONNCHADHA E., NIC PH´AID´IN C., GENABITH, J.V. 2003. Design, implementation and evaluation of an inflectional morphology finite state transducers for Irish. Machine Translation, Springer Vol 18 Number 12pgs 173-193.

KOSKENNIEMI, K. 1983. Two-level morphology: a general computational model for word-form recognition and production. Publication No. 11. University of Helsinki: Department of General Linguistics.

OFLAZER, K. 1999. Morphological analysis. In: Van Halteren H (ed) Syntactic Wordclass Tagging. Dordrecht: Kluwer Academic Publishers. pp. 175205.
SPROAT, R.(1992). Computation and Morphology MIT press