A framework for bootstrapping morphological decomposition

L. Joubert, V. Zimu, M. Davel and E. Barnard

Human Language Technologies Research Group
CSIR / University of Pretoria, Pretoria, 0001
ljjoubert@csir.co.za

Abstract

The need for a bootstrapping approach to the morphological decomposition of words in agglutinative languages such as isiZulu is motivated, and the complexities of such an approach are described. We then introduce a generic framework which can be employed for this task, and show a number of simple examples of its use for the decomposition of words in isiZulu. Initial thoughts on the process of rule induction are discussed.

1. Introduction

Human languages derive much of their power from the unbounded richness offered by the systematic combination of linguistic entities such as words, phrases and sentences [1] . At the root of this combinatorial tree lies the formation of lexical entities (usually referred to as "words") from their meaning-bearing constituents ("morphemes"). Morphological analysis is the task of extracting and interpreting these morphemes from words. Much as syntactic analysis attempts to decompose a sentence into its constituents, and assign categories to those constituents, morphological analysis (MA) aims at decomposing and labelling the components of words.

Languages differ widely with respect to the complexity of word morphologies: languages such as Turkish of Finnish, which, as a matter of course, construct words by the copious addition of prefixes and suffixes are known as agglutinative languages. Languages such as English or Japanese are considered to be non-agglutinative, since the composition of stems and morphemes is much less productive in those languages. Consequently, MA is much more important (and, generally, challenging) in agglutinative languages. Since languages in the Bantu family are all agglutinative in nature, the automation of MA is an important task in our goal to develop Human Language Technologies for all the languages of South Africa.

Automated MA has received much attention in the past two decades. A variety of approaches have been studied (see [2] for an overview); currently, the dominant approach is based on the so-called two-level approach proposed by Koskenniemi [3]. This approach brings many of the powerful tools of finite-state transducers to bear on MA, and has been applied successfully to numerous languages, including Finnish and kiSwahili [2]. Significantly for our purposes, Bosch and Pretorius have shown that the two-level formalism can be applied to isiZulu, and results in a powerful and accurate system for MA[4].

The biggest drawback of the two-level approach to MA is that it is both labour and skills intensive: a reasonably complete system for an agglutinative language may require several years' effort by skilled linguists and computer scientists. Since our goal is to develop open-source systems for MA in all the languages of South Africa, we would like to develop approaches that can produce acceptable MA with reduced requirements in terms of human resources - even if the resulting system is not quite as accurate or complete as the best systems. This is analogous to our creation of pronunciation dictionaries with reduced development costs; the key insight in that case was to combine human linguistic expertise with machine-learning algorithms in a bootstrapping approach. The current paper describes a framework that enables us to apply bootstrapping to MA.

In Section 2, we review the general principles of the bootstrapping approach. Section 3 describes the MA formalism at the heart of our approach, and provides some early results for isiZulu. Finally, Section 4 summarizes our view of the road ahead in the development of the bootstrapping approach to MA.

2. Bootstrapping for the creation of linguistic resources

Bootstrapping approaches are a useful way to quickly and cost-effectively create linguistic resources[5]. For example, when acoustic models are developed for a new target language, an automatic speech recognition system can be initialised with models from an acoustically similar source language, and these initial models improved through an iterative process in which audio data in the target language is automatically segmented and used to retrain the models [6]. Bootstrapping approaches are applicable to various language resource development tasks, specifically where an automated mechanism can be defined to convert between various representations of the data considered. In the above example, two represen-

tations are utilised: annotated audio data and acoustic models, and the mechanisms to move from one representation to the other, are well defined through the phone-alignment and acoustic-modelling tasks, respectively. Similarly, pronunciation dictionaries can be created by repeatedly converting between phonetic transcriptions of words (as corrected by human transcribers), and pronunciation rules (automatically induced from those transcriptions) [5].

To apply these ideas to MA, we need a formalism that is sufficiently powerful to represent the typical phenomena that occur in agglutinative languages, yet simple enough so that machine-learning approaches can induce sensible rules from limited samples. While the two-level formalism certainly meets the first requirement, we have not been able to develop learning algorithms that successfully extract two-level rules from examples. We have therefore designed an alternative formalism, as described below. As in acoustic training and pronunciation prediction, it is designed to be used in a closed-loop fashion, where a human "trainer" corrects the analyses found by the system. These corrections are used to update the system's rules, whereupon a further round of corrections and rule updates is initiated. This cycle continues until the MA system is deemed sufficiently accurate.

3. A simple finite-state formalism for MA

In order to develop a bootstrapping approach to MA, we need a formalism which makes it relatively easy for native-language speakers to specify facts about the morphology of a language, yet also supports the induction of rules that generalize from a given set of examples (as well as the set of rules already established). Note that these requirements tend to conflict with one another: rule induction is generally easiest if the rules are highly explicit, so that the relationship between rules and examples is easily derived. However, such explicit rules tend to be too long-winded for human consumption, and formalisms such as two-level morphology therefore support a more compact, implicit specification of the MA process. We therefore need to strike an appropriate compromise between the two requirements.

To this end, we have developed a morphological analyser that is defined in terms of three classes of elements:

1. Lists, which define all the word elements or morphemes (roots, prefixes, suffixes).
2. Rules, which capture the phonological alternations in the language using regular expressions.
3. Tables, which capture the morphosyntactics, defining how different word elements are combined to form words.

The syntax for the above elements are defined as a formal grammar, which is given in Fig. 1 in Extended Backus-Naur Form (EBNF) format.

For example, the specification in Fig. 2 describes some of the class-1 nouns in isiZulu.

3.1. List entry

A list entry is a comma-separated list of word elements. It is used to group together word elements that have similar behaviour in terms of morphosyntactics.

3.2. Rule entry

A rule entry describes the changes or alternations that result when two word elements are combined. It is assumed that alternations occur at the junction of two word elements, and that the alternation may influence both elements. The rule therefore has a pre-modifier (that determines the context and alternation of the first word element), and a post-modifier that determines the context and alternation of the second word element.

These rule modifiers are expressed in terms of regular expressions. A modifier consists of two parts, a matching regular expression used to determine the *context* in which the modifier is applicable, and a replacement regular expression that defines the alternation that takes place when the modifier is applied. This replacement regular expression may contain references to sub-expressions in the *context*, allowing for alternations where certain letters are dropped.

Thus the rule entry "(um)u, $1, [aeiou].*, $0" states that "umu" becomes "um" when joined with a word element that starts with a vowel, while the second word element remains unchanged. The "$" is used to refer back to marked sub-expressions (defined by parentheses). "$0" specifically refers to whatever was matched in the matching regular expression of the post-modifier. (Note that this simple example does not distinguish between stems consisting of a single syllable and multi-syllabic stems; such a distinction is easily captured in the regular-expression formalism, but makes the example unwieldy for explanatory purposes.)

A rule can optionally be composed of a sequence of rule entries. When a rule is applied in a table entry, each rule entry in the rule sequence is applied in succession.

3.3. Table entry

A table entry describes how different word elements (list or table entries) can be combined to form more complex word elements. Each table entry defines the two word elements to be combined, as well as the applicable rule that describes the alternations for the specific combination.

```
<list> ::= list <list_identifier> "=" <element> { "," <element> }
<list_identifier> ::= letter { letter | digit }
<element> ::= letter { letter }

<rule> ::= rule <rule_identifier> "=" <modifier>
        ", " <modifier> { ", " <modifier> "," <modifier> }
<rule_identifier> ::= letter { letter | digit }
<modifier> ::= regex "," regex

<table> ::= table <table_identifier> "=" <list_identifier>
        | <table_identifier> "," <rule_identifier> "," <list_identifier>
        | <table_identifier> { "," <rule_identifier> "," <list_identifier>
        | <table_identifier>  }
<table_identifier> ::= letter { letter | digit }
```

Figure 1: *EBNF specification for components of MA framework*

```
list umu = umu
list class1 = ntu, fana, ona

rule rule1 = (um)u, $1, [aeiou].*, $0,
                    (um)u, $1, .*[aeiou].*[^aeiou].*[aeiou].*, $0


table table1 = umu, rule1, class1
```

Figure 2: *Sample elements used to define a simple morphology in isiZulu*

3.4. Analyser operation

The analyser parses the file containing the list, rule and table definitions, and compiles the data into a tree. The edges of the tree map to the surface form of words, and the nodes or vertices map to the underlying form. Accepting nodes are formed when a list or table entry occurs as the last element in a table entry. Figure 3 shows an example of such a tree, built out of a subset of rules for isiZulu nouns.

When parsing a word for morphological decomposition, a depth-first search of the tree is attempted. Each character of the input is matched against edge values. When the input is consumed and the search ends in an accepting node, the stack of nodes visited in the search forms the underlying form of the input word.

These examples show that this approach is suitable to capture simple "local" morphological phenomena. They also suggest a straightforward approach to learning: whenever the "trainer" corrects an analysis (see Section 2), the smallest change to either make a rule fit the context, or to modify a rule which does fit, so that it produces the right output, is attempted. This proposed change is then tested against all the other examples already analysed by the system; if it conflicts with any of those, it is rejected and the next larger rule change is attempted.

An important factor in the success of this approach is the use of an appropriate metric to compare the magnitudes of different possible changes to the system of rules. For these local rules, it seems as if a simple symbol-counting metric is appropriate – that is, the magnitude of a change is defined to be the number of simple elements in the lists, rules, or tables that need to be added, modified, or deleted.

The ability to use back-references provides some ability to perform non-local processing as well, but we have encountered phenomena in isiZulu that are not easily handled in this manner. Also, learning becomes more complex when non-locality is allowed, since numerous rule changes can then produce comparable outputs from the analyser. It may be best to not allow automatic learning of such non-local rules – this matter is currently being studied.

4. Conclusion

Morphological analysis of lexical data is a fundamental task in Human Language Technologies. We have developed a relatively simple formalism that makes it possible to specify the phenomena that occur in the morphology of languages such as isiZulu. Although this formalism is significantly more involved than, for instance,

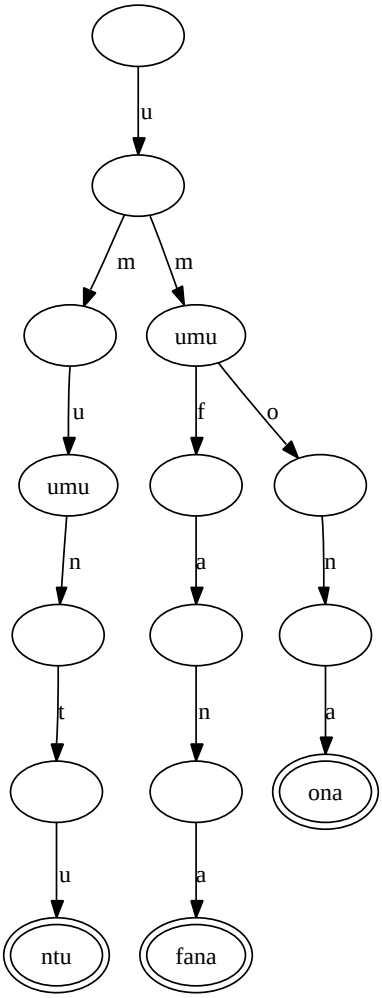

Figure 3: *A simple MA tree, constructed from the lists, rule and table in Fig. 2.*

the simple rewrite rules that form the basis of our system for bootstrapping pronunciations, it does seem suitable for the development of a method for rule induction. The added complexity of the proposed formalism makes it relatively easy to define a starting point for rule induction (whereas a less powerful formalism - e.g. the approach suggested by [7]) would probably not make much progress for an agglutinative language. This also implies that the "trainer" may possibly be require to develop a fairly sophisticated initial system for bootstrapping to succeed (in contrast to our pronunciation-prediction system, where even relatively unsophisticated "trainers" were able to produce highly accurate systems [8]). This may be an unavoidable consequence of the complexity of morphological analysis.

In order to test this hypothesis, and our overall approach, we intend to develop an explicit set of learning rules within our formalism, and then to write a user interface which makes it possible to refine the analyser in a standard bootstrapping approach. Applying this interface and learning formalism to isiZulu will allow us to assess the potential of the proposed framework.

5. Acknowledgements

This material is based upon work supported by the National Research Foundation under Grant number 2053242, and by the CSIR *Information Society Technologies Centre*. Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Research Foundation.

6. References

[1] D. Jurafsky and J. H. Martin, *Speech and Language Processing*, Prentice Hall, 2000.

[2] K.R Beesley and L. Karttunen, *Finite State Morphology*, CSLI Publications, 2003.

[3] K. Koskenniemi, "Two-level morphology: A general computational model for word-form recognition and production," Tech. Rep., University of Helsinki, Department of General Linguistics, 1983.

[4] L. Pretorius and S. Bosch, "Finite-state computational morphology-treatment of the zulu noun," *South African Computer Journal*, vol. 28, pp. 30–38, 2002.

[5] M. Davel and E. Barnard, "Bootstrapping for language resource generation," in *Proceedings of the 14th Symposium of the Pattern Recognition Association of South Africa*, South Africa, 2003, pp. 97–100.

[6] T. Schultz and A. Waibel, "Polyphone decision tree specialization for language adaptation," 2000.

[7] A. Van den Bosch and W. Daelemans, "Memory-based morphological analysis," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, ACL'99*, University of Maryland, USA, 1999, pp. 285–292.

[8] M. Davel and E. Barnard, "The efficient creation of pronunication dictionaries: Human factors in bootstrapping," in *Proceedings of the ICSLP*, Jeju, Korea, 2004.