The significance of computational morphological analysis for Zulu lexicography

Sonja E Bosch
Department of African Languages, University of South Africa, P O Box 392, UNISA 0003, South Africa
e-mail: boschse@unisa.ac.za

Laurette Pretorius
Department of Computer Science and Information Systems, University of South Africa,
P O Box 392, UNISA 0003, South Africa
e-mail: pretol@unisa.ac.za

March 2002

With regard to the significance of computational morphological analysis for Zulu, and more specifically for Zulu lexicography, the aims of this article are the following: Firstly, we discuss computational morphological analysis as an enabling technology in the field of natural language processing (NLP) with specific reference to lexicography. Secondly, we explain how a computational morphological analyser for Zulu is built by using the Xerox finite-state tools. Thirdly, we demonstrate the application of such an analyser by means of examples from Zulu. Finally, we argue that in Zulu lexicography, a computational morphological analyser is an indispensable tool for facilitating some of the basic outputs of electronic corpora.

Introduction

The importance of automatic linguistic analysis and interpretation of corpus data for computerised corpus-based lexicography is well-known (cf. Kruyt, 1995:119, and De Schryver & Prinsloo, 2000:95). A lexicographer's analysis of corpus data may be supported by various tools or enabling technologies, such as morphological analysers or lemmatisers, part-of-speech taggers, (text) tokenisers and syntactic parsers to mention a few. Kruyt (1995:127) emphasises that in lexicography such tools facilitate some of the basic outputs of electronic corpora,

> ... firstly by allowing queries at various linguistic levels, secondly by computing lexical patterns that are not easily observable by human analysis, and finally by the facility of concordance sorting according to relevance. If implemented in a lexicographer's workbench, the tools provide the lexicographer with the facility to have different views on large amounts of corpus data with a speed and flexibility that is inconceivable within the traditional method of manually arranging quotation slips.

In this article we concentrate on computational morphological analysis, a technology which facilitates the development of more practical applications such as grammatical tagging. Particularly in lexicography, grammatical tagging is an indispensable tool for corpus-based analyses, especially in the African languages with their rich morphological structures. By means of the grammatical tagging of a corpus, which is regarded as a crucial resource for lexicography, the corpus is enriched as a reusable source of

linguistic information for compiling or improving dictionaries.

Lawler & Dry (1998:175) confirm that,

> A program to parse words into their component morphemes is invaluable for languages with complex morphological structure, both to model and test a morphological analysis and to provide inflectional information required for syntactic analysis.

Although morphological analysis is in fact one of the most basic enabling applications, and is presupposed by other rule-based techniques of automatic language processing such as tokenising, part-of-speech tagging, parsing and so forth, the fact is that

> morphological analysers still remain to be written for all but the commercially most important languages. (Cole et al, 1997:96)

This is also the case for Zulu and most of the languages belonging to the Bantu language family, which have not featured significantly in natural language processing, (with the exception of Swahili, see Hurskainen, 1992) although the complex morphological nature of these languages particularly calls for specialised tools for automatic analysis of word forms.

Against this background, the aim of this article is twofold, namely to argue that in Zulu lexicography a computational morphological analyser is an indispensable tool for facilitating some of the basic outputs of electronic corpora, taking into account the challenges posed by a computational analysis of Zulu morphology; and to explain how a computational morphological analyser is built. In particular, we demonstrate the link between natural languages and the finite-state networks that provide the basis for computational morphology, and mention how such a computational morphological analyser can be constructed with the Xerox finite-state software tools (Beesley & Karttunen, forthcoming).

Challenges posed by a computational analysis of Zulu morphology

Languages are traditionally classified according to their morphological structure, which may range from isolating, through agglutinating to fusional and even polysynthetic systems. The languages belonging to the Bantu language family are predominantly agglutinating in nature, which means that the majority of words are formed by means of prefixes and suffixes that are built around a basic root.

This is clearly illustrated in the following examples from Zulu, which are formed from the basic root **-hamb-** 'walk/go/travel':

(1a) *umhambi (u-mu-**hamb**-i)* 'traveller'
(1b) *ngiyahamba (ngi-ya-**hamb**-a)* 'I am walking/going'
(1c) *kuhanjwa (ku-**hamb**-w-a)* 'It is being walked'
(1d) *isihambeli (i-si-**hamb**-el-i)* 'visitor'

It should be noted that Zulu, like all other Nguni languages, follows the convention of a conjunctive writing system. So what we observe in these examples, are linguistic words which each consist of a root surrounded by its affixes. Each word is of a poly-morphematic nature or consists of a number of bound parts or morphemes which can never occur independently as separate words. The two types of morphemes that are generally recognised, are roots and affixes. Roots, which are also called radical morphemes, always form the lexical core of a word, while affixes usually add a grammatical meaning or function to the word.

A further observation that can be made about the above examples is that the only constant core element in these words is the root. This is generally the case in predominantly agglutinating languages such as Zulu which are characterised by words consisting of more than one morpheme.

In the light of De Schryver & Prinsloo's comment (2000:97) that two of the basic outputs of an electronic corpus that are of interest to the lexicographer, are word frequency counts and concordance lines, it is essential that word roots

be identified in order to extract accurate frequency counts and concordance lines from an electronic corpus. This process calls for each word in the corpus to be morphologically analysed. From our examples it should become clear that especially in Zulu, as a conjunctively written language, scientifically reliable word frequency counts and concordance lines are in general difficult to obtain without (automated) morphological analysis.

Let us look at another example, namely that of the complex nature of the monosyllabic verb with the verb root **-dl-** 'eat' for instance:

(2a) *u**dl**ile (u-**dl**-ile)* 'he has eaten'
(2b) *usazo**dl**a (u-sa-zo-**dl**-a)* 'she will still eat'
(2c) *ku**dl**iwe (ku-**dl**-iw-e)* 'it was eaten'
(2d) *uyaba**dl**isa (u-ya-ba-**dl**-is-a)* 'she causes/helps them to eat'
(2e) *kwa(be)ku**dl**eka (ku-a-be-ku-**dl**-ek-a)* 'it was edible'

The number of occurrences of **-dl-** that would possibly be registered by performing a simple find and replace exercise on a text corpus of one million plus words is unimaginable. Consequently, if for arguments sake we wish to retrieve certain information from a corpus, for instance all monosyllabic verb roots in Zulu (such as **-y-; -f-; -ph-; -dl-; -mb-; -z-; -w-; -th-; -sh-; -lw-; -kh-; -n-; -b-; -ny-; -s-**) for purposes of lemmatisation, frequency analyses, concordances, or language learning, each word form in the corpus needs to be morphologically analysed, preferably automatically, in order to get reliable feed-back on the identification of the verb roots.

In this regard Hurskainen (1997:633) remarks:

> Many tasks in the process of retrieving language-specific information from text can be carried out in more than one way, by using suitable tools nowadays abundantly available. What cannot be substituted by commercial, shareware, or public domain tools is the morphological parser.

The reason is that morphological rules that are applicable to one language, are not necessarily applicable to another. Therefore, the morphological analysis needs to be language-specific. Two problems which are, however, central to any morphological analysis, are the following:

- **morphotactics**, also known as word-formation rules. Words are typically composed of minimal meaningful units, namely morphemes. The morphemes that make up words cannot combine at random but are restricted to certain combinations and orders. A morphological analyser needs to know which combinations of morphemes are valid.
- **morphological alternations**. The same morpheme may be realised in different ways depending on the environment in which it occurs. A morphological analyser needs to identify the correct form of each morpheme.

We will now pay attention to some of the implications of morphotactics and morphological alternations in Zulu.

Firstly, regarding the morphotactics or word formation rules, there are numerous possibilities of morpheme concatenations in Zulu word categories for which possible legal combinations would need to be determined. Let us briefly look at the case of verbs and nouns. Diagram 1 on the next page gives us an idea of possible morpheme combinations and legal orders that need to be recognised for the word category verb. The verb root may be preceded by at least seven prefixes, while various extensions and a terminative may be suffixed to the verb.

When it comes to nouns, some of the possible legal combinations that need to be determined, are illustrated in diagram 2 on the next page. Roots are preceded by a noun prefix which may be subdivided into a preprefix and a basic prefix, and may be followed by nominal suffixes such as the diminutive and augmentative, or even deverbative suffixes. Furthermore, preprefixes may be preceded by various morphemes such as locative or copulative prefixes.

Secondly, let us briefly consider the rules that

Diagram 1: Morphology of the verb

Hort	Neg	Subj	Neg	Asp	Tense	Obj/ Refl	Root	Ext	Term
	a	i		sa			buy 'return'	is + w	a
		ba	nga			ku	phek 'cook'		i
ma		si					akh 'build'	an	e
		ngi			zo	ni	siz 'help'		a

Diagram 2: Morphology of the noun

Loc Prefix	Pre-prefix	Basic prefix	Root	Nominal suffix	Deverb suffix	
	i	m	vu	kazi		'ewe'
	i	si	khathi	ana		'little time'
	u	mu	thwal		o	'load'
ku	u		baba			'to father'

determine the form of each morpheme in Zulu. Words are just concatenations of morphemes, but raw concatenation often results in abstract 'morphophonemic' words. There are 'alternations' between the raw concatenations on the abstract level and the desired final words on the surface level, which are best illustrated by the examples in the following diagrams:

Diagram 3a: Morphological analysis of *umzali* 'parent'

Underlying morphemic level						
Preprefix	+	Basic prefix	+	*-zal-*	+	deverb suffix
Abstract (morphophonemic) level						
u-	+	*-mu-*	+	*-zal-*	+	*-i*
Surface level						
u-	+	*-m-*	+	*-zal-*	+	*-i*

Diagram 3b: Morphological analysis of *sabanjwa* 'we were caught'

Underlying morphemic level						
Subject concord	+	*-bamb-*	+	pass ext	+	terminative
Abstract (morphophonemic) level						
sa-	+	*-bamb-*	+	*-w-*	+	*-a*
Surface level						
sa-	+	*-banj-*	+	*-w-*	+	*-a*

Diagram 3c: Morphological analysis of *ithutshana* 'small chance'

Underlying morphemic level						
Preprefix	+	Basic prefix	+	*-thuba*	+	Nom suffix
Abstract (morphophonemic) level						
i-	+	*-li-*	+	*-thuba*	+	*-ana*
Surface level						
i-	+	–	+	*-thutsh-*	+	*-ana*

The identification of the underlying root morpheme on the surface level in diagrams 3b and 3c is complicated by the fact that certain morphophonological processes have taken place. Consequently the method of comparing surface word forms and then identifying the indivisible, meaningful and recurrent element as the lexical item, becomes impractical and virtually impossible.

A morphological analyser has to take morphophonological processes, that is where changes in the sounds of morphemes are based on surrounding phones, into account. For instance, it needs to be specified in the analyser that palatalisation could occur with verbal extensions such as the passive; with noun suffixes such as the diminutive and so forth, as in the examples in diagrams 3b and 3c.

Other morphophonological processes which often occur across morpheme boundaries and need to be provided for in the analyser are vowel elision, vowel coalescence and consonantalisation, e.g.

- **Vowel elision**
  (3a) *ba-ya-osa* > *bayosa* 'they are roasting'
- **Vowel coalescence**
  (3b) *na-isi-khathi* > *nesikhathi* 'and time'
- **Consonantalisation**
  (3c) *uku-eb-a* > *ukweba* 'stealing'

In the following section it will be shown that the challenges posed by morphophonological processes as illustrated in diagrams 3a, 3b and 3c, may be addressed within the framework of computational analysis, for instance by means of finite-state networks in which both morphotactics (word-formation rules) and morphological alternations are modelled. The result is a system (or lexical transducer) that maps from surface word forms to their underlying base forms and vice versa. The underlying base forms consist of canonical representations of the morphemes of the words plus sequences of tags showing the morphological characteristics of the word forms.

Finite-state networks for computational morphology

Since the 1950s finite-state technology has been the subject of extensive research. Two of the main reasons for this are its mathematical elegance and the efficient implementation possibilities that it offers. In this section we briefly discuss the connection between natural languages and the finite-state networks that form the basis of finite-state computational morphology, emphasising the basic ideas, rather than a rigorous mathematical exposition, which may be found in the literature. See for example Xerox Research ... [sa], Natural Language ... 2002, Aho & Ullman (1972), Perrin (1990), Aho (1990), Cohen (1997) and Lewis & Papadimitriou (1998).

Linking languages and finite-state networks

The link between natural language and finite-state networks for computational morphology is demonstrated by means of an example. Consider a simple language Z consisting of two Zulu words, $Z = \{$*umuntu* 'person', *umngane* 'friend'$\}$. Let us now take a closer look at the network (network 1), called a *deterministic finite automaton* (DFA), that *defines* or *accepts* exactly this language.

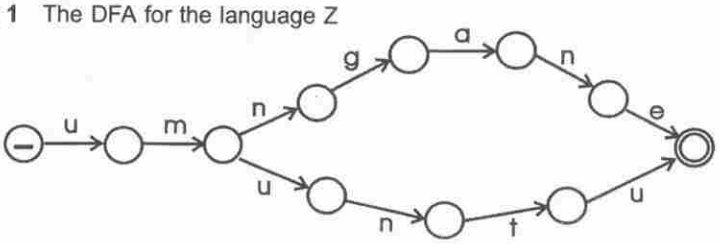
In our networks 1, 2, 3 and 4 the circles represent *states*, the arrows represent *arcs or transitions*, while the *labels* on the arcs represent the letters of the words of our language. The start state is denoted by the symbol -, and the final (halting) states by double circles. No two arcs leaving a state may have the same label. The ? denotes any other symbol not explicitly mentioned, and 0, the empty string.

In order to check whether any given string of characters belongs to our language, we feed it to the network matching input letters to arc labels, and if we reach the final state, the network having consumed all the characters in the string, we know that the string is a word in Z, otherwise it is not. Following this approach it should be clear that our network 1 accepts Z, but nothing else.

Having a network that accepts a language, however, does not bring us sufficiently close to an automated morphological analyser/generator. What we ideally need is a network which accepts *input*, for example a word that needs to be analysed, and then *outputs* the analysis of the word, or vice versa. This can be achieved by modifying a finite automaton to become a so-called *finite-state transducer (FST)*.

A finite-state transducer is a device similar to a finite automaton. However, while a finite automaton is only a language acceptor, a finite-state transducer does not merely accept words, but maps input strings in one language to output strings in another language. In general, the input alphabet and language, and the output alphabet and language of a transducer are different. The network for a deterministic finite-state transducer therefore also resembles that of a deterministic finite automaton, except that the label on the arrow (transition) is of the form a/w, or a:w, meaning 'if the input symbol is a, follow this arrow and emit output w'. The language with the a in its alphabet is often referred to as the *upper language*, and the one whose alphabet contains the w is called the *lower language*. Having label a on the arrow of a transducer is a shorthand notation for a/a, that is a is mapped to itself. Using the symbol 0 to denote the transition to the next state without giving any input or output is also a notational convention.

Network 1 The DFA for the language Z



One crucial and useful characteristic of finite-state transducers is that they may be combined in various ways, which again give rise to (more complex) finite-state transducers. This allows us to start with, for example a transducer for a one or two word language, and systematically expand it to a single transducer for the entire Zulu language.

Having introduced the concept of a finite-state transducer, we are ready to perform the task of automating morphological analysis/generation, in particular, the modelling of the appropriate morphotactics and alternation rules for Zulu.

Let us return to the language, *Z*, of our example.

From the discussion above, it follows that conceptually we need two networks - the one to take care of the morphotactics, the other to handle the alternation rules that apply. Networks 2 and 3 represent the morphological analysis/generation of the language Z.

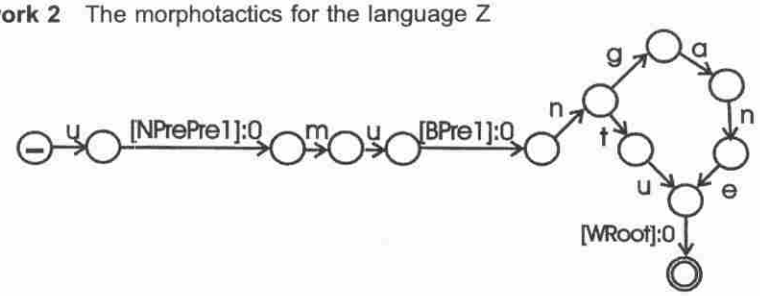
Network 2 shows the morphotactics for *Z*, while the alternation rule for *Z* may be implemented as network 3.

A closer look at network 3 shows that

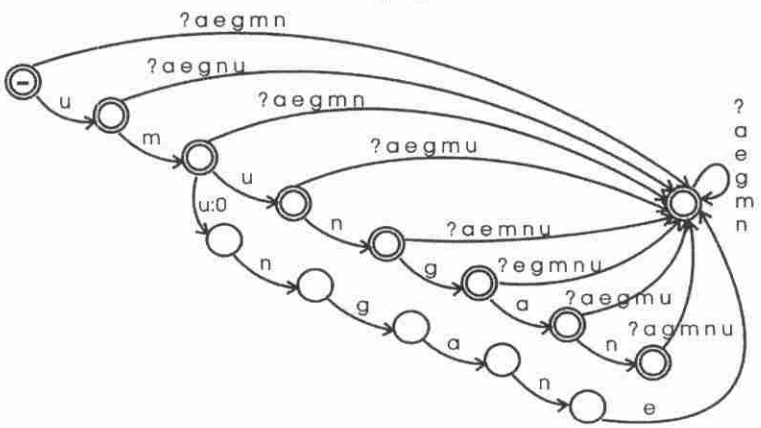
- the upper language is the universal language;
- all words that are not affected by the rule are mapped to themselves;
- words that are matched by the rule leave 0 (the empty string) on the lower side where u (in the specified context) is on the upper side (see label u:0);
- the rule is applied to all words in the lexicon at compile time, using the so-called composition operation.

By combining these two transducers (shown as networks 2 and 3), we obtain a third, shown as network 4, and called FST. FST is a computational morphological analyser/generator for *Z*, which is illustrated in diagram 4.

Network 2 The morphotactics for the language Z



Network 3 The alternation rule for the language Z



Network 4 The FST for the language Z

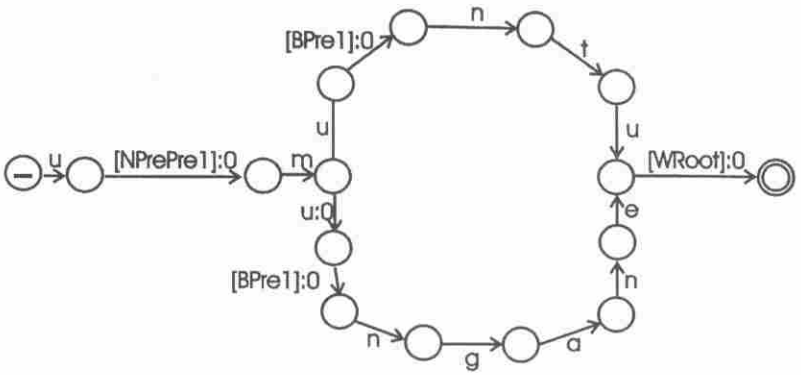


Diagram 4: The morphological analyser/generator for Z applied to *umngane*

u[NPrePre1]mu[BPre1]ngane[WRoot]

↑

FST

↓

umngane

The Xerox finite-state calculus and tools for computational morphology

Research in computational morphology in organisations such as Xerox Research Centre Europe, is based on the fundamental insight that the complexities of word-formation rules as well as morphological alternations, can be handled extremely efficiently with the help of finite-state networks.

The Xerox finite-state calculus (Beesley & Karttunen, forthcoming) is a powerful, sophisticated, state-of-the-art set of algorithms and programming languages for building finite-state solutions to a variety of problems in natural language processing. While a detailed discussion thereof is beyond the scope of this article, we briefly mention those software components we found useful in automating morphological analysis/generation for the Zulu language.

The **lexc** tool, for {lex}icon {c}ompiler, is used for specifying the required natural-language lexicon and the morphotactic structure of the words in this lexicon. The resulting network generates morphotactically well-formed but still rather abstract strings. Recall, for example, the lower side string *u m u n g a n e* in network 2 above. Such strings are sometimes called 'underlying' morphophonemic or, in finite-state tradition, lexical strings.

The **xfst** tool is used for formulating the alternation rules, required to map the abstract lexical strings into properly spelled surface strings as they occur in natural language, by means of regular expressions. A regular expression is a notion that is mathematically equivalent to an associated finite automaton. It is a representation that is particularly suited to human interpretation. Regular expressions are discussed in any book on formal languages and automata theory, and have found many practical applications in a great variety of software products.

Finally, the two finite-state networks are then combined together (using the operation of composition) into a single network, namely a so-called lexical transducer. The lexical transducer contains all the morphological information about the language being analysed, including derivation, inflection, alternation, compounding and so forth, as is shown in network 4, also referred to as the transducer FST for *umngane*.

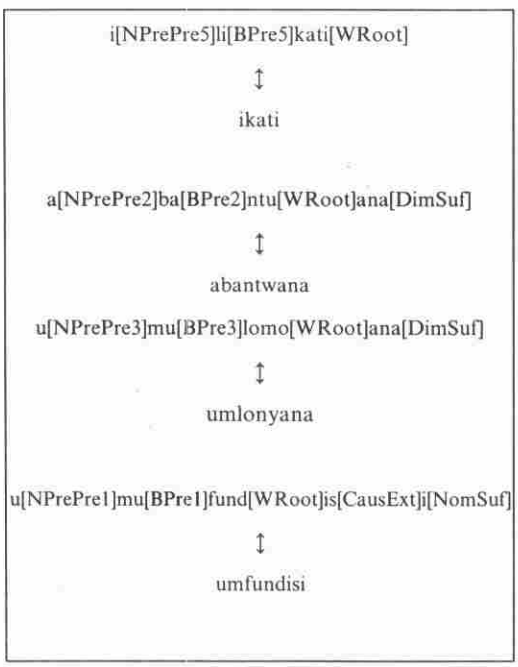
For a more detailed discussion on building and implementing these networks for Zulu nouns, see Pretorius & Bosch (2002).

Applying a morphological analyser/ generator for Zulu

Our *main objective* is the development of a computational morphological analyser/generator for Zulu, by means of the Xerox finite-state tools. The *research challenge* is to model the morphological structure of Zulu in such a way that *all* the Zulu words are *included* (generated) and *correctly analysed*, but that all character strings that do not represent words in the real language be excluded. In other words, our ultimate aim is that the lexc input should define a comprehensive lexicon of the Zulu language, as well as all the word formation rules (morphotactics) that apply in the language. The xfst input should contain all the alternation rules that are required to produce well-formed words in the Zulu language.

Diagram 5 shows examples of types of nouns that are already being handled by our current prototype. The symbol ↕ indicates the bidirectional use of the morphological analyser/generator:

Diagram 5: Bidirectionality of the morphological analyser/generator



Concluding remarks

The complex morphological structure of a language such as Zulu necessitates automated morphological analysis in order to make information retrieval from an electronic corpus truly scientific and reliable. We therefore conclude that for the purposes of Zulu lexicography a computational morphological analyser is an indispensable tool for facilitating some of the basic outputs of electronic corpora.

Acknowledgements

The authors are grateful to Dr Ken Beesley (XRCE) for constructive feedback and valuable suggestions on an earlier version of this article, as well as for his continued support and expert advice on the project.

The authors would also like to acknowledge the financial support of the National Research Foundation (NRF) for the project *Computer Analysis of Zulu Morphology*; and the contribution of Xerox Research Centre Europe (XRCE), whose software and training are a central component of this project.

References

Aho, A.V. 1990. Algorithms for finding patterns in strings. In: *Handbook of theoretical computer science Volume A: Algorithms and complexity*. J. van Leeuwen (ed.). Amsterdam: Elsevier.

Aho, A.V. & Ullmann, J.D. 1972. *The theory of parsing, translation, and compiling — Volume 1: Parsing*. Englewood Cliffs, N.J.: Prentice-Hall Inc.

Beesley, K.R. & Karttunen, L. Forthcoming. *Finite-state morphology: Xerox tools and techniques (preprint)*. Cambridge: Cambridge University Press.

Cohen, D.I.A. 1997. *Introduction to computer theory*. New York: John Wiley & Sons, Inc.

Cole, R., Mariani, J., Uszkoreit, H., Zaenen, A. & Zue, V. 1997. *Survey of the state of the art in*

human language technology. Cambridge: Cambridge University Press.

De Schryver, G-M. & Prinsloo, D.J. 2000. The compilation of electronic corpora, with special reference to the African languages. *Southern African Linguistics and Applied Language Studies*, 18:89–106.

Hurskainen, A. 1992. A two-level formalism for the analysis of Bantu morphology: an application to Swahili. *Nordic Journal of African Studies* 1(1): 87–122.

Hurskainen, A. 1997. A language sensitive approach to information management and retrieval: the case of Swahili. In: *African Linguistics at the Crossroads: Papers from Kwaluseni*. R.K. Herbert (ed.), pp 629–642. Köln: Rüdiger Köppe.

Kruyt, J.G. 1995. Technologies in computerized lexicography. *Lexikos*, 5:117–137.

Lawler, J. & Dry, H.A. 1998. *Using computers in linguistics*. New York: Routledge.

Lewis, H.R. & Papadimitriou, C.H. 1998. *Elements of the theory of computation*. Upper Saddle River, N.J.: Prentice-Hall Inc.

Natural Language Processing FAQ. 2002. [O]. Available: http://www.faqs.org/faqs/natural-lang-processing-faq/
Accessed on 2002/02/08.

Perrin, D. 1990. Finite automata. In: *Handbook of theoretical computer science Volume B: Formal methods and semantics*. J. van Leeuwen (ed.). Amsterdam: Elsevier.

Pretorius, L. & Bosch, S.E. 2002. Finite-state computational morphology - treatment of the Zulu noun. *South African Computer Journal* 28: to appear.

Xerox Research Centre Europe. [Sa]. [O]. Available:
http://www.xrce.xerox.com/competencies/content-analysis/fst/home.en.html
Accessed on 2002/02/08.