Evaluation of the performance of a machine learning lemmatiser for isiXhosa

L Mzamo
24827304

Dissertation submitted in fulfilment of the requirements for the degree *Magister* in Computer and Electronic Engineering at the Potchefstroom Campus of the North-West University

Supervisor: Prof. ASJ Helberg
Co-supervisor: Prof. S Bosch

November 2015

DECLARATION

I hereby declare that all the material incorporated in this thesis is my own original work except where specific reference is made by name. This work has not been submitted for a degree at another university.

Signed:_____

Lulamile Mzamo

ACKNOWLEDGEMENTS

I suppose the closest kin suffer the most in these self-inflicted endeavours. To my boys, Naanda and Sangqa, I know that you have suffered the most in the time that I was doing this work. Know that this is just another example I give to you although you are the most hurt by it. Thank you for being the anchors and the raison d'être.

Lipolelo Masole, the faithful nanny to my children, thank you for your loyalty and extra hours of work.

Happy Mbuso Bhengu, words are inadequate to express my gratitude for your support over the course of these studies. Thank you for your understanding and for affording me space to finish these studies.

Mama wam, Nosebenzile Mzamo, indima yakho ayinakulinganiswa.

MaCirha, ngqokelela kaMlandeli Mzamo, nangamso!

Professors Helberg and Bosch, this work is complete only because of the guidance you provided, your rigor and the benefit of the doubt you afforded me. I cannot thank you enough.

Patience Luxomo, thank you for being my hiding place, my happy place and my sanctuary.

ABSTRACT

Human language resources (HLR) and applications currently available in South Africa are of a very basic nature, with lemmatisation being one of the basic. South African languages, except for English are considered underdeveloped when it comes to HLRs. The work detailed in this thesis is the development of a lemmatiser for one such language, namely isiXhosa. The previous benchmark in isiXhosa lemmatisation, which achieved 79.28%, was a rule-based lemmatiser implemented for the development of isiXhosa lemmatisation data. That data was used in this study.

IsiXhosa, one of the South African official languages belonging to the Bantu language family that are classified as "resource scarce languages", is the second largest language in South Africa with 8.1 million mother-tongue speakers, second only to isiZulu. IsiXhosa is closely related to languages such as isiZulu, Siswati and isiNdebele and the work done in it could easily be bootstrapped to these languages.

A lexicalised probabilistic graphical lemmatiser, the IsiXhosa Graphical Lemmatiser (XGL), was investigated, designed, implemented and evaluated against two benchmark lemmatisers, the CST Lemmatiser and the LemmaGen lemmatiser.

The investigation towards the XGL involved five objectives. The first objective was to establish good characteristics for an automatic lemmatiser for morphologically complex languages. This was achieved by reviewing existing research material on the lemmatisation of morphological complex languages. To establish the most appropriate lemmas for isiXhosa in the context of natural language processing, a study of the isiXhosa language morphology was done, and appropriate lemmas for each word category were identified. Exploring the training data answered the objective of establishing what good data features are for an isiXhosa lemmatiser. The objective of designing an isiXhosa lemmatisation model was realised through the implementation of XGL. The last objective, the evaluation of an isiXhosa lemmatisation model, was achieved through training and testing XGL, and comparing it to two benchmark lemmatisers, the CST Lemmatiser and the LemmaGen lemmatiser.

The XGL lemmatiser achieved the highest accuracy compared to the selected benchmark lemmatiser, with an accuracy rate of 83.19%.

KEY TERMS

Natural Language Processing, Human Language Technology, Machine Learning, Lemmatisation, IsiXhosa

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

ANN – Artificial Neural Network

ANOVA – Analysis of Variance

CFG – Context Free Grammar

CTexT – Centre for Text Technology

DAG – Directed Acyclic Graph

DCG – Definite Clause Grammar

FN – False Negative

FP – False Positive

FSA – Finite State Automata

GDX – Greater Dictionary of IsiXhosa

HCL – Helsinki Corpus of Swahili

HLT – Human Language Technology

HMM – Hidden Markov Models

KB – Kilobytes

KNN – K- Nearest Neighbour

LCS – Longest Common String

LiA –“Lemma-identifiseerder vir Afrikaans” [Lemmatiser for Afrikaans]

MBL – Memory-based learning

MBSMA – Memory-Based Swahili Morphological Analyser

MDL – Minimum Description Length

MSD – Morphosyntactic Description

NCHLT – National Centre for Human Language Technologies

NLP – Natural Language Processing

OoV – Out of Vocabulary

PBAC – Prototype Based Active Learning

PFCG – Probabilistic Context Free Grammars

POS – Part of Speech

RDR – Ripple Down Rules

RMA – Resource Management Agency

SVM – Support Vector Machines

TiMBL – Tilburg Memory-Based Learner

TN – True Negative

TP – True Positive

WER – Word Error Rate

XGL – IsiXhosa Graphical Lemmatiser

CHAPTER 1: INTRODUCTION

This thesis demonstrates that machine learning can be used to automate the lemmatisation of isiXhosa. Based on an understanding of the isiXhosa language and an analysis of existing isiXhosa lemmatisation data, a machine learning lemmatiser was designed, implemented, and evaluated against two benchmark lemmatisers.

1.1 Motivation

The Constitution of the Republic of South Africa (1996) recognises eleven (11) official South African languages. It also recognises the need for redress in that it requires that "all official languages must enjoy parity of esteem and must be treated equally". Knowledge and information, as well as the distribution thereof, are an important part of ensuring redress. In South Africa, the majority of text is in English and continues to be created and distributed in English. The other South African languages are considered under-resourced languages.

The above status in South African languages is reflected in the developments in human language technologies. Human language resources and applications currently available in South Africa are very basic. According to Groenewald (2009), this can be attributed to the dependence on Human Language Technology (HLT) expert knowledge, scarcity of data resources, lack of market demand for the African languages, and how the particular language relates to other more resourced languages. English benefits from world developments in HLT and Afrikaans has benefited, albeit to a lesser extent, because it is similar to Dutch. Lemmatisation is one of the basic tools in natural language processing (NLP). The work detailed in this thesis is the development of a lemmatiser for one of South Africa’s under-resourced languages, isiXhosa.

IsiXhosa is one of the South African official languages belonging to the Bantu language family which are classified as "resource scarce languages" (Groenewald, 2009).

IsiXhosa is an agglutinating and highly inflected language with affixes substituting for what would be important parts of speech in other languages. It has a "complex and productive derivational system" (Bennet, 1986 as quoted by Prinsloo, 2011), and its orthography is conjunctive. There has been work in computational linguistic tools for isiXhosa but the work has been limited (Sharma Grover et al. 2010).

IsiXhosa is closely related to languages such as isiZulu, Siswati and isiNdebele; therefore work done in it could easily be bootstrapped to these languages, as has been shown in Bosch et al. (2008).

1.1.1 IsiXhosa

IsiXhosa is one of the 11 official languages in South Africa. It has 8.1 million mother-tongue users, 16% of the South African population, and is second only to IsiZulu (Statistics South Africa, 2012:25). It is spoken primarily in the Eastern Cape province of South Africa and is the second most dominant language of the Western Cape Province.

IsiXhosa is a Bantu language, similar to isiZulu, Siswati and isiNdebele. IsiXhosa is ethnologically classified as S.41 in the Guthrie Nguni languages classification system (Maho, 2009; Lewis et al., 2014). The Glottocode for isiXhosa is "xhos1239". The Ethnologue (2014) and the Glottolog (Hammarstrom et al., 2014) are two of the genealogical classification systems used in world language classification. Other language classification methods use geographical origins, i.e. areal classification and typology (Bender, 2013:6).

IsiXhosa uses the same 27-letter alphabet (including <space>) and the ten numeral symbols, 0 to 9, as English, but some of the letters denote different sounds from their English representation, e.g. **c, q** and **x** (McLaren, 1948:1). IsiXhosa also uses the same punctuations as English.

Every isiXhosa syllable is open, i.e. ends in a vowel (Mncube, n.d.:1; McLaren, 1948:3). Vowels can stand as independent syllables if they are at the beginning of a word (McLaren, 1948:4). Some vowels, however, can be swallowed by a preceding consonant, which then gives that consonant a syllabic nature. This is common in the occurrence of *m* (Mncube, n.d.:1; McLaren, 1948:3). Consonants can be combined into various forms to produce different sounds, e.g. ***ngca***. These are called compound consonants (Boyce, 1844:3). This makes the syllabic structure of isiXhosa simple, with minor exceptions.

According to Pahl (1982:1), isiXhosa words are composed of morphemes, and an isiXhosa morpheme is seldom used alone as a word form. A morpheme is the smallest meaning bearing component of a word (Kosch, 2006). Each word has a root and affixes, i.e. suffixes, prefixes and circumfixes. A circumfix is the "simultaneous affixation of a prefix and suffix to a root or a stem to express a single meaning" (Kosch, 2006). An example of a circumfix in isiXhosa is the combination ***a****…**nga*** in isiXhosa negation, e.g. ***a****kahamba**nga** [he/she **did not** go].* Most roots, which are the meaning carrying constituents of words, consist of two syllables (Meinhof, 1932:36).

Examples of prefixes:

*Uyaphi? <**U**-ya-phi*
[Where are you going? -> you-going-where?]

*Balele <**Ba**-lel-e*
[They are sleeping -> They-sleep.]

Examples of suffixes:

*uhambile < u-hamb-**ile***
[he/she is gone] < she/she go-PAST-TENSE.SUFFIX]

*isityakazi < i-sitya-**kazi***
[a big dish < a dish-big]

*Injana < i-nja-**ana***
[A small dog < a-dog-small]

Examples of circumfixes:

*akalalanga < **aka**-lal-**anga.***
[he/she is not sleeping < she/he.is.not-sleep-NEGATION.SUFFIX.]

In the example above, the negative prefix *aka-* and the suffix *–anga* express a single meaning, namely negation.

Each of the affixes (i.e. prefixes, suffixes or circumfixes) is made up of one or more morphemes. Morphemes follow one another in an order prescribed for each word type (Louw et al., 1984).

IsiXhosa is an agglutinating and polysynthetic language in that it has many morphemes per word (Kosch 2006; Bender 2013). It is also fusional/inflectional because morpheme boundaries are fused and difficult to distinguish (Kosch, 2006).

1.1.2 Automated Lemmatisation for isiXhosa

Lemmatisation is "concerned with finding the lemma of a set of inflected word forms or with assigning lemmas to inflected word forms" (Spiegler, 2011).

In natural language processing lemmatisation is looked at in terms of inflection, as "a normalisation step on textual data, where all inflected forms of a lexical word are reduced to its common headword, the lemma" (Erjavec & Dzeroski, 2004). Jurafsky and Martin (2000) explain this by stating that in the context of natural language processing, a lemma represents a set of lexical forms with the same stem, the same major part-of-speech and the same word-sense.

A process similar to lemmatisation is stemming. For a particular paradigm, a stemmer simply finds the common substring among the paradigm word forms. The lemmatiser, in contrast, maintains the meaning. An example is that the lemma for "*better*", and "*best*" is "*good*" (Daelemans et al., 2009).

Prinsloo (2011) cites a popular morphologically inspired definition of lemmatisation as "the selection of a canonical form to represent a specific paradigm". This approach is supported by Manning and Schutze (1999:132) by saying that a lemma "imply disambiguation at the level of lexeme, such as whether a use of lying represents the verb lie:-lay - to prostrate oneself or lie: -fib". The work of Jones et al. (2005), shows the value of lemmatisation, at least in dramatically improving spell-checking for a highly inflected language, such as isiXhosa. This confirms that lemmatisers generally improve precision and recall in information retrieval, as stated by Jongejan and Dalianis (2009).

One of the earliest reports on automated morphological analysis of isiXhosa is that of Theron and Cloete (1997) on the automatic acquisition of a Directed Acyclic Graph (DAG) to model the two-level rules for morphological analysers and generators. The algorithm was tested on English adjectives, isiXhosa noun locatives and Afrikaans noun plurals. The algorithm was implemented for Afrikaans lemmatisation and achieved 5-fold validation accuracy of 93% for Afrikaans noun plurals (Russell & Norvig 2014).

The next lemmatisation work on isiXhosa was a supplement to spellchecking (Jones et al., 2005). The primary objective was to identify lemmas so that inflection could then be applied to increase the lexicon of the spellchecker. This approach increased the lexical recall of the spellchecker from 78.82% to 92.52%.

The last lemmatisation work for isiXhosa was used to generate isiXhosa lemmatisation data. The work was presented by Eiselen and Puttkammer (2014). The exercise reported a rule-based lemmatiser accuracy rate of 79.82% when measured against a gold standard.

1.2 Proposed Research Work

The work presented in this study covers investigating and implementing a machine learning based lemmatiser for isiXhosa, and its results will be compared to other machine learning lemmatisers that are freely available.

1.2.1 Research Hypothesis

It is expected that a machine learning lemmatiser specifically designed for isiXhosa will perform significantly better linguistically than existing lemmatisers in the lemmatisation of isiXhosa.

Therefore, the null hypothesis is that a machine learning lemmatiser specifically designed for isiXhosa will not perform significantly better linguistically than existing lemmatisers in the lemmatisation of isiXhosa.

1.2.2 Research Questions

The main research question, therefore, is "How does the performance of a machine-learning lemmatiser designed specifically with isiXhosa in mind compare with other machine-learning lemmatisers on the lemmatisation of isiXhosa?"

The main research question was addressed by answering the following questions:

(1) What characteristics do the most successful lemmatisers have?

(2) What is the appropriate lemma for isiXhosa in a Natural Language Processing context?

(3) What are good data features for an isiXhosa lemmatiser and how should they be structured?

(4) What is a good way to model an isiXhosa machine learning lemmatiser?

(5) How does the performance of a lemmatiser that implements the above model compare to existing similar lemmatisers on the lemmatisation of isiXhosa?

1.2.3 Research Objectives

The research questions posed above result in the following study objectives:

(1) To define the characteristics of a successful lemmatiser;

(2) To define the appropriate lemmas for isiXhosa in the context of Natural Language Processing (NLP);

(3) To determine good data features for the lemmatisation of isiXhosa;

(4) To design and implement a model for an isiXhosa machine learning lemmatiser, and

(5) To compare the implemented isiXhosa lemmatiser to existing machine learning lemmatisers.

The above objectives are expounded upon in the following section, section 1.3: *Research Methodology.*

1.3 Research Methodology

The work conducted was an experimental study. In an experimental study, an intervention is applied to a sample of a population, and the results of the interventions to the sample evaluated

and compared to one or more control interventions and generalised to the population (Welman et al., 2005:78; Rasinger, 2013:41).

In this study, the population is isiXhosa text and the sample is an existing isiXhosa lemma annotated corpus. The intervention being evaluated is a machine learning lemmatiser that was specifically designed for isiXhosa, and the control interventions are existing lemmatisers. The process of applying the lemmatisers involves training the lemmatisers using part of the corpus and evaluating them against a testing corpus.

The objectives stated above were achieved by following the phases below:

1.3.1 Literature Review

A thorough literature study was done on:

(1) Human language technology techniques;

(2) Lemmatisation techniques, and

(3) HLT measurement techniques.

The objective of the literature study was to gain an understanding of the broad field of human language technology, specifically the techniques that are used in the field, and then to focus on the techniques used in lemmatisation. This work is presented in chapter two of the document.

This study provided guidance on a good approach to implementing a lemmatiser for isiXhosa, on choosing good control lemmatisers that could be used for comparison purposes, and on how to measure and compare the performance of the lemmatisers.

1.3.2 Determining an appropriate lemma for isiXhosa in the Natural Language Processing context

To get to a lemmatiser that is specifically designed for isiXhosa, a study of the lemmatisation aspects of the language are required. Because isiXhosa is a morphologically complex language, it was important to do an analysis of the morphology of isiXhosa words to establish the most appropriate lemma form for each word category of isiXhosa.

As there is contention regarding the best approach to word categorisation for isiXhosa, a list of categories that would meet the needs of the study was adopted. Each word category was analysed and conclusions were made on what would be the best lemma form for that word category. This work is presented in chapter three of this document.

This analysis guided the work on feature selection.

1.3.3 Feature Selection

This study did not require data annotation as there was already isiXhosa lemma annotated data available on the Language Resource Management Agency's website[1] . This data conformed to the lemmas defined in the study on the appropriate lemma for isiXhosa for the natural language processing environment. Because isiXhosa is an affixing language, an analysis of the coverage of the different types of affixes was explored. The characterisation of the data was meant to find a heuristic that points to good features in the data that could be used in a lemmatiser for isiXhosa. This was primarily a statistical analysis. This work is presented in chapter four of the document.

From this study, a good combination of features was chosen and used in the design of the isiXhosa lemmatiser.

1.3.4 The isiXhosa Lemmatiser

A lemmatiser specifically for isiXhosa was designed, implemented and tested. The model of the lemmatiser was designed, and the model was implemented as a set of applications. The lemmatiser, as a machine learning lemmatiser, is trained using word lemma pairs, and generates the model from that input. The trained lemmatiser can then be used to lemmatise other isiXhosa words. This work is presented in chapter five of the document.

The objective of this work was to implement a lemmatiser that is specifically designed for isiXhosa.

1.3.5 Evaluation

To evaluate the performance of the isiXhosa lemmatiser against existing lemmatisers, experiments were set up and the results were captured and compared. Statistical comparisons that had been determined during the literature review were used to test the study hypothesis.

The experiments were set up to ensure the reliability and validity of the results. Reliability is a measure of the repeatability of an experiment, i.e. the method used repeatedly and providing consistent and stable measurements (Rasinger, 2013:28; Welman et al., 2005:9).

[1] The Language Resource Management website address is http://rma.nwu.ac.za

To ensure the validity of the result, the experiments were set up to ensure that the results could be compared without bias, using the 10-fold cross validation.

1.4 Thesis Structure

This document comprises seven chapters including this introduction.

Chapter two covers the literature review, which includes a study of Human Language Technology techniques and lemmatisation studies conducted in the recent past to establish what approach the best lemmatisers took and what characteristics they had.

Chapter three looks into the meaning of a lemma for isiXhosa in the context of natural language processing. The chapter starts by establishing a context for the language and its character. It then presents a hierarchy of word categories that provides an approach to the work. Each category is then discussed with a view to establishing what the best lemma should be for that word category.

Chapter four explores the data to find good features for use in a lemmatiser. The understanding of the language from the previous chapter guided the work. This chapter looks at the influence of the affix types on the identification of a lemmatisation strategy by calculating the cumulative coverage of each affix type and concludes by specifying what features would work for the automated lemmatisation of isiXhosa.

Chapter five presents the isiXhosa lemmatiser, which was designed and implemented from scratch. The chapter first explains the model, describes how the system works, and finally guides the reader on how to use the lemmatiser.

Chapter six details how the lemmatiser was evaluated. The chapter starts by detailing the experimental setup, including the data splits, motivates for the choice of control lemmatisers against which to benchmark the isiXhosa lemmatiser, and concludes by detailing the results.

Chapter seven summarises the work conducted, presents the main findings and reflects on future work.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

Natural language processing (NLP) is a scientific field concerned with creating techniques and methods for the processing of natural language both in audio format as in speech processing, and written format as in text processing (Manning & Schütze, 1999).

Natural language processing has concerned itself with both the analysis and synthesis of natural language. Examples of natural language analysis are word boundary identification in speech and morpheme identification in text; examples of synthesis are speech synthesis in speech and word form derivation in text.

Since the work in this study is on text, the author will constrain the rest of the document to text processing.

Bates (1995) categorises the fundamental challenges in natural languages processing and understanding thereof under syntax, semantics, pragmatics and discourse. Jurafsky and Martin (2000:4) extended this by prefixing the list with phonetics and phonology, and morphology. This study focuses on lemmatisation; therefore it will be confined to morphology. This chapter starts by detailing HLT techniques in general, it then zooms into techniques that have been used in lemmatisation, and finally suggest features for a machine learning lemmatiser.

2.2 HLT Techniques

Jurafsky and Martin (2000:5) categorises the elements of a natural language processing toolkit under "**state machines, formal rule systems, logic,** and **probability theory"**, and goes on to highlight a state space search algorithm and dynamic programming as among the most important elements.

This document categorises HLT techniques under knowledge-based/rules based, statistical based and hybrid systems.

2.2.1 Knowledge-Based/Rules Based

Rule-based systems implement language rules that have been defined by expert linguists. As linguistic work started by analysing words and grammar, morphosyntactic rules have been known for a long time and computational linguistics naturally started by using those rules. Chapter three will discuss the aspects of isiXhosa morphology that are relevant to this study.

The most prevalent rule-based systems are Finite State Automata, Context Free Grammars, First Order Logic and Definite Clause Grammars. These are detailed below.

2.2.1.1 Finite State Automata

A finite-state machine/automaton (FSA) is a computational machine that can be in one of a finite set of states. A state machine consists of three things, i.e., states, state transition functions and data. One of the states is the initial/start state. The state machine can have a number of final/termination states, where the finite-state machine is allowed to finally stop. The other states are intermediate states and the inability of the state machine to move from these intermediate states is considered a fault. Movement between states is determined by the state transition function/s. A transition function attached to a state takes the data as input and returns the next state based on the characteristics of the data.

State transition functions are modelled as regular expressions in morphological analysis applications (Jurafsky & Martin, 2000:21). The most widely used finite state automata system in HLT is the Xerox Finite State Automata system composed of a Finite-State Lexicon Compiler (`lexc`) and the Xerox Finite State tool (`xfst`) (Karttunen, 1993). The FSA has been used in the South African context for morphological analysis (Pretorius & Bosch, 2005; Jones et al., 2005) and lemmatisation (Brits et al. 2005).

2.2.1.2 Context-Free Grammars

Another form of rule-based systems is **context free grammars** (CFG). Context free grammars have been used for parsing sentences into phrases and terminals/words (Collins, 2003; Spiegler et al., 2010). For example:

```
S -> NP VP
NP -> D N
VP -> V NP
VP -> D V
with S=Sentence, VP=Verb Phrase, NP=Noun Phrase, N=Noun,
V=Verb and D=Determiner.
```

A sentence could then be parsed as follows:

```
John is going -> S=(N=John VP=(D=is V=going))
```

Context free grammars work at word level and are used for sentence parsing. Context free grammars are a model at parts-of-speech level and only represent relationships between categories.

2.2.1.3 First-Order Logic and Definite Clause Grammars

Yet another type of rules based system is **first-order predicate calculus** (Russell & Norvig, 2014:), also known as first order logic. First-order logic allows one to specify a set of truth statements, and then test to see if an assertion could be inferred from those truths. Inference in first-order logic provides for the querying of a system for cases where a particular input would be true (Russell & Norvig 2014:327). This makes for good morphological analyses.

Definite Clause Grammars (DCG) are a form of first order logic used in artificial intelligence and are mostly implemented in the Prolog language. Examples of DCG rules are:

`w --> n.`	`word`
`n --> iv, nst1.`	`Noun -> initial vowel + noun stem 1`
`nst1 --> npf, nst2.`	`Noun stem 1 -> noun prefix + noun stem 2`
`nst2 --> nr, dim.`	`Noun stem 2 ->: noun root + diminutive`
`npf --> n2.`	`Noun prefix`
`iv --> [a].`	`Terminal initial vowel`
`n2 --> [ba].`	`Terminal noun class 2 prefix`
`nr --> [ntu].`	`Terminal noun root`
`dim --> [ana].`	`Terminal diminutive`

A word could then be analysed as follows:

```
abantwana [children] ->  w=(n=(iv=[a], nst1=(npf=(n2=[ba]),
nst2=(nr=[ntu],dim=[ana])))) = a(iv)ba(n2)ntu(nr)ana(dim)
```

The *Ukwabelana corpus,* an isiZulu corpus, was generated with DCGs (Spiegler, 2011; Spiegler, et al. 2010).

2.2.2 Statistical Based HLT Techniques

Statistics and probability have played a large role in natural language processing. The initial statistical work was founded on information theory; it then progressed to the use of artificial intelligence techniques. Before going into the statistical techniques and the statistical nature of

language, one must differentiate between the supervised and unsupervised training of stochastic systems.

2.2.2.1 Learning System Training Modalities

Statistical based systems learn a model from data. This model is then used in prediction. The nature of the training data in relation to what the system needs to predict determines whether the system is supervised or not.

If the training data contains input/output pairs, then the system is a **supervised** system. If the learning system's training data does not contain output samples, then the system is **unsupervised**.

That said, it is very rare to get a fully unsupervised system because the exercise of developing the system implies supervision, albeit not from training data but from a human. In the validation of a system, some prediction samples are also used by its designer to validate and tune the system. This is another form of supervision. However, because the training algorithm itself does not have access to the prediction sample, this phenomenon is therefore referred to as **semi-supervised** training.

Having learnt about the learning modalities of statistical system, one can consider the statistical nature of language before looking at the statistical techniques used in NLP.

2.2.2.2 Zipf's Law

One of the fundamental characteristics of language is Zipf's law (Manning & Schütze, 1999; Zipf, 1945). Zip's law characterises the relationship between the frequencies of occurrence f of a type of language phenomena to its rank r in relation to others in the same category. For example, if the category "letters" is considered, each letter being the type in that category, then the relationship between the frequencies of occurrence of each letter to its frequency's rank among other letters, has the relationship:

$$f \propto \frac{1}{r} \qquad (1)$$

Letters on a corpus of isiXhosa data (van Huyssteen & Snyman, 2012) showed a distribution of letters that follows Zipf's law. This is shown in Figure 1 below, with '*' denoting <space>.

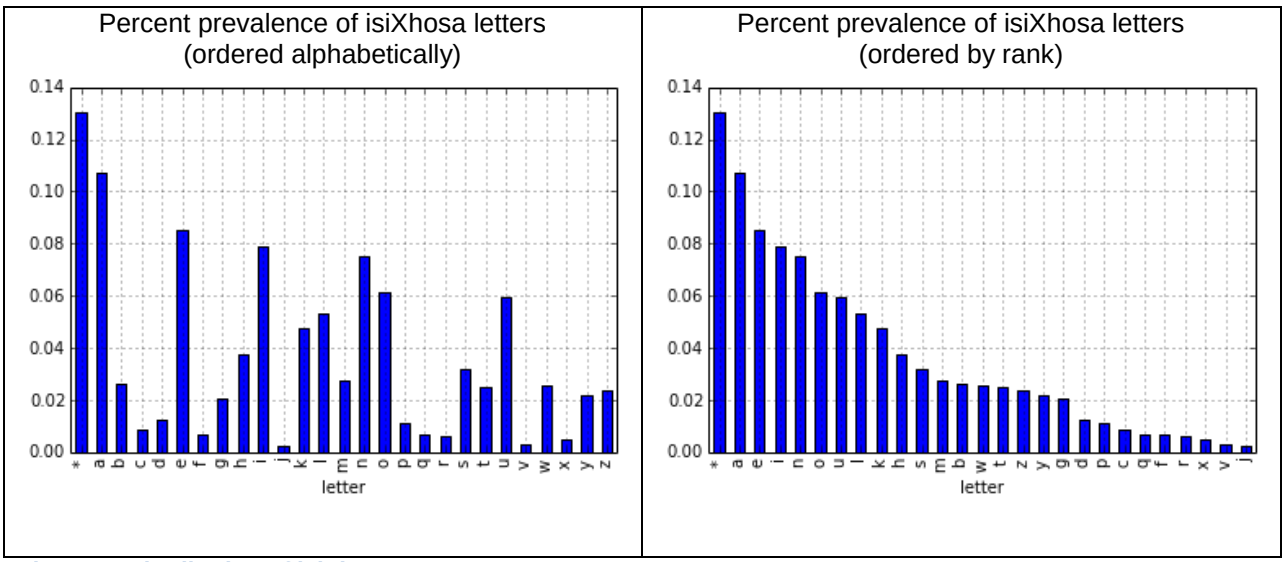

Figure 1: Distribution of isiXhosa Letters

2.2.2.3 Maximum Entropy

A number of information theory based tools have been used in natural language processing. The minimisation of mutual entropy has been used in language modelling (Manning & Schütze, 1999:73; Jurafsky & Martin, 2000:226) and syllabification (De Pauw & de Schryver, 2009). Of interest however, has been the use of maximum entropy in language modelling (Manning & Schütze, 1999:589; Berger et al., 1996), POS tagging, ambiguity resolution (Ratnaparkhi, 1998) and morphological analysis (Shalonova & Golenia, 2010).

"Entropy is a measure of uncertainty or diversity. The more we know about something, the lower the entropy" (Manning & Schütze, 1999:73). Given a number of models, the one with the lowest entropy has a better quality.

If given a model that predicts the future data with a probability distribution of *m(x),* even though the true probability distribution is *p(x),* the performance of that model can be calculated from data entropy *H(X*) and cross entropy between model predictions and actual readings *D(p||m)* as:

$$H(X,m) = H(X) + D(p \mid m) = -\sum_{x} p(x) \log m(x) \qquad (2)$$

A model that minimised H(X,m) improves prediction.

2.2.2.4 N-Grams

The N-grams language model asserts that the probability of a language token in a sequence can be computed from the preceding n-previous tokens if such probability estimates have been measured before. N-grams are used in handwriting recognition, augmentative communication

for the disabled, and spelling error correction (Jurafsky & Martin, 2000:192). They are also used in language modelling. N-grams model a probability P:

$$P(w_{n+1} \mid w_1 ....w_n) \qquad (3)$$

where $w_1 ....w_n$ is termed history of n-previous token in a sequence and w_{n+1} the option of a token being evaluated. Based on the length of the previous tokens chosen, one gets a unigram (*n=1*), bigram (*n=2*), trigram (*n=3*), and 4-gram (*n=4*), etc.

Because the n-gram probabilities are determined from type counts of a corpus, which is finite, there are valid types that are not in the corpus. This is referred to as sparseness and the counts of the missing types would be zero, which is an incorrect estimate. In addition, counting from the corpus produces poor estimates for near-zero probability types (Jurafsky & Martin, 2000:207). Compensating for these deficiencies is referred to as smoothing. There are a number of smoothing methods but the Good-Turing Discounting with back off is the most used.

Good-Turing Discounting is based on the assumption that bigrams are binomially distributed (Jurafsky & Martin, 2000:215), and it does a re-estimation of the n-gram probability of scarce tokens from the number of n-grams with higher counts. The smoothed count (c^*) is:

$$c^* = (c+1)\frac{N_{c+1}}{N_c} \qquad (4)$$

where *c* is the unsmoothed count and N_c the number of n-grams that occur *c* times.

An easier smoothing method is called Deleted Interpolation where P is calculated from all unigrams, bigrams and trigrams as follows:

$$\widehat{P} = \lambda_1 P(w_n \mid w_{n-1} w_{n-2}) + \lambda_2 P(w_n \mid w_{n-1}) + \lambda_3 P(w_n) \qquad (5)$$

where

$$\sum_i \lambda_i = 1 \qquad (6)$$

The easiest choice is $\lambda_1 = \lambda_2 = \lambda_3 = 1/3$.

2.2.2.5 Markov Models

Markov processes are stochastic state-space processes that satisfy the Markov property. A process is characterised by states and transitions between states. Stochastic processes have an associated probability of activation for each transition. The Markov property states that the selection of the next state or previous state in a process is dependent only on the current state. Such processes are regarded as memory-less. Mathematically, a Markov model satisfies the following probability equation:

$$P(X_{n+1} \mid X_1, X_2, ..., X_n) = P(X_{n+1} \mid X_n) \quad (7)$$

where X_{n+1} denotes the next state and X_n the current state. A Markov chain is another term for Markov processes.

Ordinarily all the states, transitions, and their probabilities for the Markov process are visible and determinable; however there are stochastic processes where the possible states are known but the transitions (and transition probabilities) between the states are not apparent. These processes can be modelled using Hidden Markov Models (HMM). HMMs are modelled as states, transitions and emission probabilities. The transition probabilities would determine the hidden model, and the emission probabilities show the visible output of the process.

Hidden Markov Models have been used in Morphological Analysis (Creutz & Lagus, 2005), parts-of-speech tagging (Van Eynde et al., 2000), information retrieval (Manning et al., 2009) and lemmatisation (Van Eynde et al., 2000).

2.2.3 Hybrid

Hybrid techniques use statistical methods but capitalise on existing linguistic knowledge.

2.2.3.1 Probabilistic Context-Free Grammars

Probabilistic context-free grammars (PCFG) add count of the prevalence of a particular rule in context-free grammars. These counts are used in calculating the probability of a particular sentence parse and selection of the most probable sentence parse tree. PCFGs have also been used in information retrieval (Manning et al., 2009:204) and sentence parsing (Gildea & Jurafsky, 2002; Manning & Schütze, 1999:382; Collins, 2003).

However, PCFGs have a number of limitations. The first limitation is the context insensitivity. An example cited by Russell and Norvig (2014:912) is the difference in the probabilities of "eat a banana" and "eat a bandanna". In a PCFG, the difference in the probabilities of the two words is

in P(Noun -> "Banana") and P(Noun -> "Bandanna"), and not the relationship between "eat" and "banana" or "bandanna". This is the motivation for **lexicalised PCFG**.

2.2.3.2 Probabilistic/Stochastic Definite Clause Grammar

Probabilistic definite clause grammars are very similar to PCFGs. They have been used in the sentence parsing of Vietnamese (Nguyen et al., 2013; Have, 2009). This adds statistical information to the clause grammar rules, which are then used in the disambiguation between competing rules.

2.2.4 Similarity Measure Techniques

In this text, two similarity measures are covered that have been used extensively, i.e. Minimum Description Length and Shortest Edit Distance. The nearest neighbour classifier that is dependent on similarity measures is also addressed.

2.2.4.1 Minimum Description Length

The idea of using the Minimum Description Length (MDL) in statistical natural language processing is based on the concept of "'equating 'learning' with 'finding regularity'" (Grunwald, 2005:3). MDL is concerned with finding an efficient code to represent a string of data, i.e. compression (Rissanen, 1978) or finding regularity in the data. What makes MDL appealing, is that it balances model fit and model generalisation.

Given a particular model, which is a code mapping to the data being observed, a model fit relates to how the model accurately represents the observed data. This is measured using the mean-squared-error σ^2 of the output of the model related to the observation. The lower the σ^2, the better the fit of the model to the observed data.

However, a model with a good fit to the observed data may provide a bad fit to future observations or more data from the same source. This is referred to as over-fitting. "Generalisation" is the ability of a model to fit new observations adequately. The MDL provides good generalisation because MDL penalises for model complexity. Given a model m_k such that $m_k \in M$, where M is a set of models, the MDL criteria for the most efficient model is:

$$L(s) = \min_k \left(-\log p(s \mid m_k) + k \log \sqrt{n} \right) \quad (8)$$

where $p(s|m_k)$ is the probability of the data given a particular model or cross entropy between the model and the data, *k* is the number of parameters the model uses and *n* is the size of the observed data *s*.

The MDL has been used in grammar inference (Grunwald, 2005:6), word clustering (Manning & Schütze, 1999:514), morpheme discovery (Creutz & Lagus, 2002; Creutz & Lagus, 2005) and the induction of morphology and lexical categories from text corpora (Chan, 2008).

2.2.4.2 Shortest Edit Distance

The Shortest Edit Distance/Minimum Edit Distance/Shortest Edit Script or the Levenshtein distance is a metric for measuring the difference between two strings. It is based on the fact that any string can be transformed to another string by using a series of character edit operations (insertion, deletion, substitution and swopping). The distance then is the count of these operations. A generalisation is to assign a weight to each operation, e.g. insertion=deletion=1, swopping=substitution=2, and to add the weighted operations. The Levenshtein distance is a special case where the operations are given a weight of one (Manning et al., 2009:58; Jurafsky & Martin, 2000:154). Another modality of the Levenshtein distance is to restrict the operations to insertion and deletion.

A distance measurement allows one to use numerous distance based algorithms, including regression and clustering (Chrupala, 2006).

However, even though Levenshtein distances have extensive use, the typology of the language may render these distances useless. An example is the distance between *engceni* [in the grass], *umgca* [a line], and *umnga* [an acacia tree] to *ingca* [grass], as shown in the table below:

Table 1: Discrepancy in Levenshtein Distances for isiXhosa Words.

Words	Levenshtein Distance from ingca -grass
umnga [acacia tree]	3
engceni [in the grass]	4
umgca [a line]	2

One can see from the above table that the words that are not related to *ingca* score better than a related word because with isiXhosa typology, for example, *umnga* is made up of three syllables: *u-m-nga*, while *ingca* is made up of two syllables, *i-ngca*.

2.2.4.3 K-Nearest Neighbour

The K-Nearest neighbour classifier is an unsupervised clustering system that uses a similarity measure to cluster items into cluster/classes of K items that are closest to each other.

K-Nearest neighbour has been used in morpheme induction in English (Belkin & Goldsmith, 2002) and lemmatisation has been used in Afrikaans (Groenewald, 2007).

2.2.5 Performance Evaluation Techniques

This section deals with the evaluation of the models and how their performance is measured i.e. how well the observations are predicted.

2.2.5.1 Perplexity

Entropy was discussed in section 2.2.2.3. Cross entropy can be used to measure the performance of a system.

Perplexity is sometimes used in the place of entropy and is calculated as follows:

$$perplexity(X, m) = 2^{H(X,m)} \quad (9)$$

where X is the set of input data, m is the model and $H(X, m)$ is a defined in Equation 2.

2.2.5.2 Accuracy and Error Rate

A simple measure of an algorithm's performance is accuracy and the error rate. They are defined as:

$$accuracy = \frac{T_{correct}}{T_{all}} \quad (10)$$

And

$$error - rate = \frac{T_{incorrect}}{T_{all}} \quad (11)$$

where $T_{correct}$ is the number of correct predictions, $T_{incorrect}$ the number of incorrect predictions, and T_{all} is the total number of prediction attempts.

2.2.5.3 F-Measure

The standard for evaluating performance of machine learning algorithms is precision, recall and the F-Measure (Jurafsky & Martin, 2000:578), because test data must cover the whole domain of the field and must contain instances the algorithm should identify as Positive and others as Negative. Positives are those data points that the algorithm should identify as hits, and the Negatives are those points that the algorithm should reject.

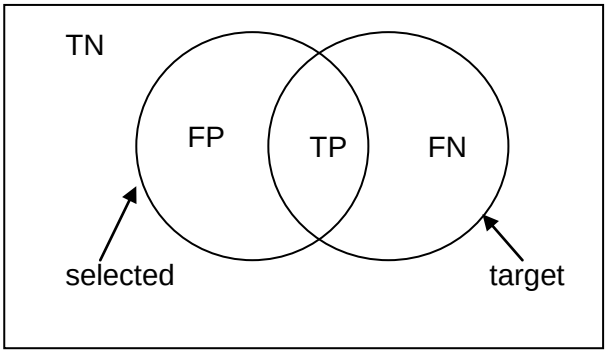

Figure 2: Graphical Representation of Precision and Recall Measures (Manning & Schütze, 1999:268)

In the graphic above TP stands for the count of True Positive, TN for the count of True Negatives, FP for the count of False Positives and FN stands for the count False Negatives.

"Precision" measures how well the algorithm correctly discriminates between Positives and Negatives. The formula for precision is:

$$precision = \frac{TP}{TP + FP} \qquad (12)$$

"Recall" is a measure of the coverage of the algorithm. It is synonymous with accuracy.

$$recall = \frac{TP}{TP + FN} \qquad (13)$$

Where *TP* is the count of true p

The *F-Measure* is a weighted average of precision and recall.

$$F - Measure = \frac{\left(\beta^2 + 1\right) \cdot precision \cdot recall}{\beta^2 \cdot precision + recall} \qquad (14)$$

A prevalent F-measure is the F1-Score, which is the F-Measure with $\beta = 1$. Most studies present the accuracy rate and the F1-Score.

2.2.5.4 Computing Resource Usage Evaluation

In addition to linguistic performance, HLT resources are measured on their use of computational resources. Results are mostly presented for execution time and memory usage. Execution time is mainly presented in seconds taken to lemmatise the testing set, and the memory in KB used by the lemmatiser.

Another approach is the normalisation of the results on the number of words in question. This is a better comparison when looking at comparing multiple batch sizes, e.g. comparing training time on different sized samples. Furthermore, because there are two stages to evaluating a lemmatiser: training and lemmatisation, it is important to compare the results for each stage. Therefore, typical computing resource usage metrics should be KB/word for memory usage and ms/word for execution duration and these should be presented for lemmatiser training and lemmatiser testing. Juršič et al. (2010) present their work in this format.

2.2.5.5 Hypothesis Testing

The t-test (Rasinger, 2013:192; Welman et al., 2005:231) is a widely used hypothesis testing metric for comparing two interventions. When comparing more than two interventions, the one-way Analysis of Variance (ANOVA) is used to check for statistical significant differences in the results of the interventions (Welman et al., 2005:237).

However, for evaluating HLT resources, the Wilcoxon signed-rank test (Wilcoxon, 1945) is the most appropriate. The Wilcoxon signed-rank test is recommended by Demšar (2006) who found that the widely used t-test was an inappropriate and statistically unsafe comparison test for classifiers. The generally accepted threshold for statistical significance is a p-value that is less than 0.05 (Rasinger, 2013:174).

2.2.6 Emerging Techniques

A number of emerging technologies are being used in HLT, particularly machine learning techniques.

Artificial Neural Networks (ANN) have been used in speech processing for a while (Jurafsky & Martin, 2000:267) and are seeing more use in text processing (Collobert et al., 2011).

Support vector machines (SVM) have been used in language identification (Botha et al., 2007) and in morphological analysis and disambiguation (Pasha et al., 2014).

Artificial Evolutionary techniques mimic nature's evolution processes in optimisation. One such evolutionary technique is the Genetic Algorithm. Some work has emerged using evolutional algorithms in grammar inference (Hrnčič et al., 2012).

2.3 Techniques Used in Lemmatisation

A number of techniques have been used in lemmatisation.

The lemmatisation problem can be looked at as a classification problem in that it is the classification of an inflectional morphosyntactic paradigm under one distinct class, namely the corresponding lemma. A more prevalent approach is to model lemmatisation with transformation classes. These classes define the transformation from word to lemma.

Automated lemmatisation can be done using linguistic rules or a data driven system. A hybrid system would be one using both linguistic rules and learning from training data.

A number of studies on automated lemmatisation have been reviewed, starting with rules based systems.

2.3.1 Rules Based Lemmatisation Work

Aduris et al. (1996) presented the morphologically based lemmatiser/tagger named EUSLEM for Basque. EUSLEM also used a lexical database. Basque is an agglutinative language with rich inflectional morphology. This morphological analyser is based on a two level morphology (Koskenniemi, 1984). The study, however, does not specify results.

Jones et al. (2005) cite the use of a lemmatiser in the development of the spelling checker for isiXhosa and how that improved the accuracy of the spelling checker from 78.82% to 92.52%.

Brits et al. (2005) presented work towards a rules based lemmatiser for Setswana. Preliminary results showed a 94% accuracy on a set of 500 verbs and a 93% accuracy on a set of 500 nouns.

Tamburini (2011) presents, AnIta, a morphological analyser based stemmer and lemmatiser for Italian, a morphologically complex language rich in inflection and derivation. Unlike isiXhosa, which is primarily prefixing in nature, Italian is predominantly suffixing (Tamburini, 2011). Italian is also disjunctive. AnIta uses the Helsinki Finite-State Transducer[2] package, and a lexicon of

[2] The Helsinki Finite-State Transducer software is available at http://www.ling.helsinki.fi/kieliteknologia/tutkimus/hfst/index.shtml

110 000 lemmas. The system recognised 97.2% of the tokens. For disambiguation, AnIta chooses the lemma based on prevalence.

The work of Suhartono et al. (2014) on the Indonesian language, Bahasa, is another implementation of a lemma dictionary and ordered rules in lemmatisation. As a language, Bahasa has both inflection and derivation. The language is circumfixing, prefixing and suffixing. The lemma dictionary was used for cases where the word-form is already a lemma. The rules were used to strip off the affixes and were defined from linguistic knowledge. The implementation achieved an accuracy rate of 98%; 57261 tokens were used in this study.

2.3.2 Data Driven Lemmatisation Studies

One of the earliest works on the automated morphological analysis of South African languages is the automatic acquisition of a Directed Acyclic Graph (DAG) by Theron and Cloete (1997) to model the two level rules for the morphological analyser and generators. The objective of the study was the generation of two level morphotactic rules from source-target word pairs. The algorithm used string edit sequences between the source and target pairs to generate the rules. Testing was done on English adjectives, isiXhosa noun locatives and Afrikaans noun plurals. All of the isiXhosa nouns presented to the system were inflected correctly to noun locatives. Afrikaans lemmatisation achieved a 5-fold validation accuracy of 93% for Afrikaans noun plurals when trained with 3935 nouns.

The work presented by Van Eynde et al. (2000) on the lemmatisation of Dutch-Flemish, starts by stating the base constraints for finding a lemma for a word. The first constraint is that the lemma must be an independently existing word form. The second constraint is that the pairing with lemma is performed on a word-by-word basis, meaning that each word must have a lemma. The last constraint is that each word has only one lemma. In this study, three existing lemmatisers were compared. The lemmatisers evaluated were a finite state transducer, a memory-based learning system, and a rule/lexicon-based system. The memory-based learning and rule/lexicon-based systems outperformed the other systems when verbs were excluded from the study at 3.6% word error rate (WER) compared to 4.8% and 5.8%. However, for all the word categories, the memory-based learning system was dismal at 18.2% WER with the rule/lexicon-based system excelling at 5.3% WER. The corpus used was 39304 word-lemma pairs and the test set was 2388 pairs in size.

Plisson et al. (2004) introduced the Induced Ripple-Down Rules (RDR) approach to word lemmatisation. RDRs were originally used for rule-based systems and resemble If-then-else statements with the most general rules appearing first and exceptions branching from them. An exception list then branches from each if-then paragraph. In essence, an RDR is part of a

hierarchy where the one level contains the rules and the following level contains exceptions to each rule, and so on. This work transformed the problem of lemmatisation into a classification problem, where the class is the transformation required to convert a word into a lemma. RDR takes as input, a lexicon of words with corresponding lemmatisation classes, automatically generates lemmatisation rules in the form of Ripple-Down Rules (RDR), and uses the generated rules to lemmatise words presented to it. The work was done on Slovene and the feature used was the suffix. To improve performance, the ripple-down rules were ordered so that the beginning of the list had shorter suffixes; 5-fold cross validation results showed an accuracy of 77%, which was an improvement at the time, with a training set of 5730 words.

Erjavec and Dzeroski (2004) conducted work on a machine learning supervised lemmatiser. The work was restricted to the Slovene open classes i.e. nouns, adjectives and verbs. Other word classes were not considered because they are closed classes. The tools used were existing Slovene tools. The work consisted of a Parts-of-Speech (POS) tagger, a morphological analyser and a lemmatiser. The tagger used was a trigram tagger, and the lemmatiser was an induced first-order decision list. A first order decision list is an ordered list of rules. The system induced the decision list from the input word form-lemma-MSD triples, where MSD stands for morphosyntactic, a feature structure showing the parts-of-speech and other morphosyntactic attributes of the word form (Juršič et al., 2010). To train the trigram tagger, 100 000 instances and 15 000 hand annotated word form-lemma pairs were used to train the lemmatiser. The lemmatiser achieved an accuracy rate of 92% on out-of-vocabulary (OOV) words.

Plisson et al. (2004) further modified the algorithm to handle exceptions better by recording words covered by a rule under that rule. With an increased corpus size from the previous work in Slovene, this study achieved performance levels of 91% accuracy when using only word form-lemma pairs for training. This work also included tests where the input included POS tags. This increased the 5-fold cross validation accuracy to 97.2%; 5720 word lemma pairs were used in training.

Groenewald (2007) presented a machine learning lemmatiser for Afrikaans named LiA (*Lemma-identifiseerder vir Afrikaans* [Lemmatiser for Afrikaans]). LiA is based on memory-based learning (MBL) and uses the Tilburg Memory-Based Learner (TiMBL), which was originally designed for Dutch. Learning in MBL, also known as instance-based classification algorithm (Mitchell, 1997:230), involves simply storing the learning instances in memory. The classification of a query involves the evaluation of the new query against stored instances in the nearest neighbour methods that use weighting of learning instances. TiMBL uses a distance-weighted Nearest Neighbour algorithm. LiA achieved over 91% accuracy, thanks to good feature selection, i.e. the right-alignment of the input to LiA; 56000 words were used for training.

The context sensitive lemmatiser, implemented by Chrupala (2006), is a pure data-driven lemmatiser. It used support vector machines (SVM) on the Shortest Edit Script on reversed words. For context, the lemmatiser used three preceding words and three following words. This lemmatiser assumed suffixal morphology, hence the reversed words. The system was tried on various European languages with varying results that correlate with the suffixal morphology assumption. The system performed well for out-of-vocabulary (OOV) words. The lemmatiser was trained with 70000 word lemma pairs, and was tested on 10000 word pairs.

The work done by De Pauw and De Schryver (2008) is interesting because it shows that a machine learning lemmatiser can be trained with the output of a rules based lemmatiser to produce superior results to the rule-based lemmatiser. De Pauw and De Schryver (2008) presented the Memory-Based Swahili Morphological Analysers (MBSMA), based on the modified memory based learning method of Van den Bosch and Daelemans (2009). It was trained using lemmas generated by a rule-based morphological analyser (SALAMA). The output of SALAMA, the Helsinki Corpus of Swahili (HCS), consists of word forms, lemmas and MSD. A sample of 97000 was extracted from the HCS and 10% was hand annotated to be a gold standard evaluation set. Two versions of the MBSMA were developed, one being the original character based analyser (MBSMA-c) and the other a syllable based analyser (MBSMA-s). The lemmas were generated by the HCS trained MBSMAs and the original HCS lemmas were evaluated against the gold standard and another analyser called Morfessor. The syllable based MBSMA had the lowest lemmatisation error rate at 11.7% followed by the HCS lemma output at 12%. The MBSMA-c performed at a word error rate (WER) of 13.6% with the Morfessor being the worst performer at 73.6%.

Jongejan and Dalianis (2009) presented a lemmatiser (CST Lemmatiser) that works with more than just suffixes because languages such as Dutch can include prefixing and infixing. The paper specifically states that the method used is not an obvious choice for agglutinated languages. This study also used a hierarchy of rules. Each rule was represented by the form: `affix0*affix1*…*affixK->insert0*insert1*…*insert.` The hierarchy is similar to Ripple-Down Rules in that for a child rule, the parent rule should hold true for candidate classes. Conflicts in lemmatisation were not handled, and the first lemma generated was accepted as the output. The implementation was compared to the suffix rules for 12 European languages. The implementation performed exceptionally well for Polish accuracy with a 24% improvement from suffix rules, but performed badly for Icelandic, with a drop in accuracy of 1.9%. The improvement in Polish is attributed to the inflectional paradigm of Polish being prefixal except for the superlative, which accounted for only 3.8% of the data. Also 23% of the data consisted of negation, which the prefixal and suffixal rules could not handle correctly. This

lemmatiser is openly sourced and freely available. Training was done with 3.8 million words in Polish.

The work of Daelemans et al. (2009) on the lemmatisation of Afrikaans, used the Prototype Based Active Learning (PBAC) method with modifications for linguistics. Active learning Prototype learning is modelled on human cognition, where it is understood that some words are more representative of the grammatical rules than others are. Such words are then prioritised over others based on certain criteria. This study used a combination of prevalence/frequency of a word, size/length and entropy as the criteria. The learning algorithm used was the memory-based leaner TiMBL. The modification to the generic PBAC was in the sequencing of training data. The generic PBAC sequencing involved starting with prototypical (most representative) data items first but, as in Daelemans et al., the less prototypical data items were prioritised. The intuition in the study was that "language processing tasks have highly disjunctive instance spaces". The study did not concentrate on the overall accuracy of the algorithm as that work had been proven by Groenewald (2006), but rather on the effect of prototypical ordering of the training data. The study showed that by reversing the sorting regime, one could reduce the number of training data instances required in Afrikaans by almost 20% and achieve the same accuracy as the non-prototype based learners.

Groenewald (2009) conducted a study on the technology transfer of LiA to the lemmatisation of Setswana. LiA was trained on 2947 lemma annotated Setswana words and achieved an accuracy of 64.05% after some minor modifications and additional work. The authors conceded that the training data was too small and that there is a possibility of improving the results of the lemmatiser if there were more data available.

Jursic et al. (2010) presented an even more enhanced Ripple-Down Rules lemmatiser called LemmaGen that was tested on 12 languages. The LemmaGen lemmatiser is an open source software (OSS). The lemmatisation of a new word is done in the same way as the most similar word form in the lexicon. The system also used the suffix feature, as it was also meant for European languages.

Ambiguation was done by choosing the most prevalent/frequent class. In a case of equal prevalence, the second most similar class was used. The original RDRs form a tree structure and are ordered, implying that the first rule to fire is accepted. To improve the efficiency and readability of rules, the LemmaGen implementation extends the RDR structure by imposing a similarity condition for an exception list, meaning that all the suffixes share the same k-1 characters, where k can be chosen. The Wilcoxon signed rank test (Wilcoxon 1945), instead of the general t-test which is discredited for this kind of work (Demšar 2006), was used to evaluate

LemmaGen against RDR and the CST Lemmatiser. LemmaGen outperformed the RDR and the CST for highly suffixal languages. It, however, did not provide a significant improvement in accuracy against RDR and CST for Italian with a 5-fold validation accuracy of 80.5 ± 0.25 % against RDR's 80.6 ± 0.27 % when implemented without POS tags. POS tagging improved the LemmaGen's accuracy beyond the performance of RDR and CST lemmatisers for all the languages, even though the accuracy of Italian was still the lowest. Training and Lemmatisation speeds improved by at least a factor of three. Romanian seemed odd in this study because it was slow in both training and lemmatisation. For Romanian, the CST was the slowest lemmatiser for training and the RDR the slowest in lemmatisation. This study presented normalised computer resource usage results.

Gesmundo and Samardzic (2012) present work on the lemmatisation of a number of European languages. The study evaluated approaching lemmatisation as a tagging task and the integration of context information. The study is conducted on eight European languages, including Hungarian and Estonian, which are agglutinative, and Slavic, which is a fusional language. The approach uses the transformation required to convert a word form to a lemma as the tag. The transformation is encoded as a four-tuple $<N_s, L_s, N_p, L_p>$ where N_p is a prefix to be removed, L_p a prefix to be inserted, N_s a suffix to be removed and L_s a suffix to be inserted. The system is trained on word-lemma pairs. For the study, a Bidirectional Tagger with Guided Learning (Shen et al., 2007) was implemented. The study achieved an accuracy of 96.5% for Hungarian, 95.8% for Estonian, and 96.1% for Slavic for lemmatisation without MSD. When MSD was incorporated, the results improved to 97.5% for Hungarian, 97.4% for Estonian and 98.1%. For Slovene, 110000 tokens were used in the study, and 10% of that was used for testing.

Agic et al. (2013) present work on the lemmatisation and POS tagging of Croatian and Serbian. The lemmatisers were trained with manually annotate word form-lemma pairs from the SETIMES.HR corpus of Croatian and the evaluation was done on both Croatian and Serbian. The intention was to capitalise on the similarity between Serbian and Croatian and the existing lemmatisers were trained. The lemmatisers used were the PurePos (Orosz & Novák, 2013), Tree-Tagger (Schmid, 1995), BTagger (Gesmundo & Samardžić 2012), all the lemmatisation capable of POS tagger, and the CST lemmatiser (Ingason et al., 2008). The CST Lemmatiser provided the best performance and achieved accuracy rates of 97.78% for Croatian and 96.30% for Serbian; 89 000 tokens were used for training and 8000 formed the test set.

2.3.3 Hybrid Lemmatisation Studies

Perera and Witte (2005) presented an algorithm for lemmatising German nouns depending on noun declension classes using an algorithm and a lexicon. This lemmatiser considers the context of the word and grammatical features and therefore uses a POS tagger and a noun-phrases chunker. The algorithm strips the word of affixes depending on the declension class. The lexicon is a self-generated lexicon of lemmas and morphological features that is used as a lookup table for disambiguation in cases where the algorithm returns two possible lemmas. The accuracy rate of the implementation was reported at 50% for the algorithm alone and at 98% for the algorithm and the lexicon.

The Lemmald lemmatiser for Icelandic was developed by Ingason et al. (2008). It achieved accuracies of 99.55% because it combines tested methods with a number of innovations. The lemmatiser is data-driven, but uses rules to improve performance. Training input to the lemmatiser was a triple of word form-lemma-POS tag. For evaluation, the system has a pre-processing step that guesses the POS tag of the word form and provides that as input to the lemmatiser. It also uses a hierarchical POS tagging and the Hierarchy of Linguistic Identities (HOLI). POS tagging improves lemmatisation by adding disambiguation. The HOLI innovation improves disambiguation. Training was done with 530 000 tokens and 60000 was used for testing.

Yet another hybrid is the Arabic stemming/lemmatiser by Bramhi et al. (2013). Their lemmatiser is based on a morphological analyser and root based stemming and has no disambiguation. The system uses a support vector machine.

2.3.4 Summary

Linguistic rules based systems have been based on finite state automata. The ones reviewed have achieved accuracy rates of up to 97%.

The prevalent data driven lemmatisation techniques primarily follow two technical streams, namely memory based learning/classifiers (MBLs) and Ripple-Down Rules (RDRs). Some work has used support vector machines (SVM), and efficiencies of up to 98% have been achieved with the RDRs; the MBLs have achieved accuracy rates of up to 92%. The features selected matter in both the RDR and MBLs, as shown by the improvement achieved in the lemmatisation of Afrikaans (Groenewald, 2007), and that shown by the reduction in lemmatisation accuracy of Italian for the suffixal LemmaGen lemmatiser (Juršič et al., 2010).

Hybrid lemmatisation techniques show the best results, especially for highly inflecting languages, as witnessed in the Lemmald lemmatisation of Icelandic (Ingason et al., 2008).

Most of the lemmatisers modelled the problem to be classification, where the relationship between a word and a lemma is characterised by a transformation class (Plisson et al., 2004; Groenewald, 2009; Jongejan & Dalianis, 2009)

For all three techniques, incorporating POS tags and a lexicon improved the results substantially, as witnessed in Lemmald (Ingason et al., 2008) and the work by Gesmundo and Samardzic (2012).

2.4 Conclusions

This chapter states what HLT objectives and techniques are used to achieve objectives. Techniques were classified into rule based techniques and stochastic techniques. The rules based techniques include Finite State Automata, Context Free Grammars and First-Order Logic. Stochastic techniques were divided into supervised and unsupervised techniques based on their learning data. The document highlighted Zipf's law, a characteristic of language. The Stochastic techniques covered were entropy maximisation, n-grams, and Markov Models, including Hidden Markov Models. The Hybrid techniques cited included Probabilistic Context Free Grammars and Stochastic Definite Clause Grammars.

The Minimum Description Length, Shortest Edit Distance and the K-Nearest Neighbour were cited as techniques that establish and use distance measures in HLT.

The performance evaluation of models was discussed and the linguistic evaluation methods mentioned were perplexity, accuracy and error rate, precision, recall, the F-measure. The computing resources evaluation metrics cited were execution duration in ms/word and the memory usage in KB/word. The Wilcoxon signed-rank test was specified as the appropriate hypothesis testing statistical method for classification systems.

The Artificial Neural Network, Support Vector Machines and Evolutionary algorithms were cited as artificial intelligence technologies that are seeing increased use in HLT.

Lemmatisation is being conducted using linguistic rules based systems, purely machine learning systems and hybrid systems. Rules based systems are primarily implemented using finite state automata. Purely machine lemmatisation techniques cover the use of memory-based learning (MBL) and Ripple-Down-Rules (RDRs). Incorporating MSD information, linguistic knowledge and a lexicon in the lemmatisation process was shown to improve lemmatisation performance.

This study suggests a machine learning lemmatiser that implements Ripple-Down-Rules and that incorporates a lexicon would provide good results for isiXhosa. This could be enhanced with Parts of Speech (POS) tags for better disambiguation.

CHAPTER 3: ISIXHOSA LEMMA FORMS IN THE CONTEXT OF NATURAL LANGUAGE PROCESSING

3.1 Introduction

To build a lemmatiser specifically for isiXhosa, one has to understand lemmatisation aspects of the language. In this chapter, the question of what is the appropriate isiXhosa lemma form for each category of isiXhosa words in the context of natural language processing (NLP) is investigated. According to Bender, "knowledge of linguistic structure leads to better features for machine learning … [and] also inform[s] error analysis of NLP systems" (2013:1). The understanding of isiXhosa in the context of NLP lemmatisation is therefore important. This chapter starts by providing a description of what a lemma actually is in isiXhosa, and because lemmas are specific to a word category, the document then enumerates word categories for isiXhosa. This is followed by details of what forms should stand as lemmas for the different word categories of isiXhosa.

A summary of the chapter is provided.

3.2 A lemma in the NLP context

Gouws and Prinsloo (2005:67) define lemmatisation in a dictionary context as "the selection of a specific form from a given paradigm to be used in a dictionary as the starting point for information retrieval". If a paradigm is the "systematic form-meaning correspondence between words in a language" (Booij, 2012:8), then a lemma in a dictionary context provides for a unique form that can be used to find a group of word-forms with similar meanings in a language. Lemmatisation, therefore, is the selection of that unique form that can be used to find a group of word-forms with similar meaning in a language. Examples are shown in Table 2.

The above definition of lemmatisation and a lemma is specific to dictionaries, but can also be applicable to general information retrieval by meaning. This explains the importance of a lemmatiser in natural language processing, and especially in searching through documents. In natural language processing lemmatisation is looked at in terms of inflection as "a normalisation step on textual data, where all inflected forms of a lexical word are reduced to its common headword, the lemma" (Erjavec & Dzeroski, 2004). Gesmundo and Samardžić (2012) clarify lemmatisation as “the task of grouping together word forms that belong to the same inflectional morphological paradigm and assigning to each paradigm its corresponding canonical form called lemma”.

Table 2: Paradigm Example

lemma	paradigm	translation
hamba	*ndiyahamba* *siyahamba* *uyahamba* *bayahamba* *iyahamba* *ziyahamba* *uhambile* *bahambile* *ndihambile* *sihambile* *ihambile* *zihambile* *abahambanga*	I'm going We are going He/she is going They are going Its going They (non-human objects) are going He/she left They went I went We went It went They went They did not go

A process similar to lemmatisation is stemming. For a particular paradigm, a stemmer simply finds the common substring among the paradigm word forms. The lemmatiser looks more for meaning. For example, the lemma for "better" and "best" is "good" (Daelemans et al., 2009).

Jurafsky and Martin (2000:195) explain that in the context of natural language processing, a lemma represents a set of lexical forms with the same stem, the same major part-of-speech and the same word-sense.

The issue of a lemma is even more conflated in isiXhosa text because it is a synthetic language. IsiXhosa morphology consists of bound and free morphemes, and morphemes from open and closed classes.

A morpheme is the smallest meaning-bearing component of a word. For example, the morphemes in the word "boys" has two morphemes, "boy" which means a male child, and "-s" which carries the plural meaning. Closed classes are those morpheme classes where all the morphemes in a class are known. Open morpheme classes are those that are subject to the productive nature of word formation. Bound morphemes cannot exist as words on their own, they need to be bound to another morpheme, whilst free morphemes do not need to be bound to another morpheme (Kosch 2006:5). In the "boys" example, "boy" is a free morpheme, as it can stand on its own, while "-s" is a bound morpheme because it requires something to bind to.

Morphemes are divided into roots and affixes. The root is the semantic carrying morpheme of a word. However, the root of an isiXhosa word is generally syllabically incorrect, e.g. the root of *babalekile* [they ran] is *balek*, hence the emphasis on the stem, *baleka*. In isiXhosa, most roots are bound morphemes which are not independently meaningful (Kosch, 2006). Stems are word

roots suffixed with a termination vowel (Louw et al., 1984), and hence the use of stems as the appropriate lemma for isiXhosa.

In summary, a lemma in isiXhosa in the context of natural language processing is a "form with the same stem, the same major part of speech and the same word-sense" (Jurafsky & Martin 2000: 195). Even in a synthetic language, this tri-criterion should be maintained.

3.3 Word Categories of isiXhosa

Because different categories of isiXhosa words are made up of free or bound morphemes, it is better to look at what a lemma would be for each particular word category. For a synthetic language, like isiXhosa, word categories are not straightforward.

Word categorisation in isiXhosa has evolved from the classic Germanic approach introduced by Boyce (1844) and later McLaren (1948) to that used in Pahl (1982) and later Louw et al. (1984), and the *Greater Dictionary of isiXhosa* (Pahl et al., 1989; Mini & Tshabe, 2003; Tshabe & Shoba, 2006).

The morphological approach of Louw et al. (1984) provides for 13 word categories for isiXhosa i.e. noun [*isibizo*], pronoun [*isimelabizo*] (absolute, demonstrative, quantitative, and emphatics absolute), adjective [*isiphawuli*], relative [*isibaluli*], enumerative relative [*isichazi sobalo*], possessive [*isimnini*], verbal [*isenzi*] (auxiliary verb [*isenzi esilabalabayo*]), copulative [*isibanjalo*], verbal relative [*isibaluli esakhiwe kwisenzi*], descriptive [*isihlomelo*], ideophone [*isifanekisozwi*], conjunction [*isihlahnganisi*] and interjection [*isikhuzo*].

The *Greater Dictionary of Xhosa*: Volume 3 (Q to Z) (Pahl et al., 1989) provides for a traditional nomenclature of isiXhosa word categories i.e. noun, pronoun, verb, qualificative, conjunction, adverb and ideophone.

The *Greater Dictionary of isiXhosa*: Volumes 1: (A to J) and 2: (K to P) (Mini & Tshabe, 2003:xxxiv; Tshabe & Shoba, 2006:xxv) list 13 word categories for isiXhosa i.e. noun [*isibizo*], verb [*isenzi*], relative [*isibaluli*], adjective [*isiphawuli*], copulative [*isibanjalo*], pronoun [*isimelabizo*], qualificative [*isichazi*l, possessive [*isimnini*], quantificative [*isahluli*], enumerative [*isiquki*], ideophone [*isifanekisozwi*], adverb [*isihlomelo*], and form word [*isikhapi*].

For the purposes of lemmatising isiXhosa, the following main word categories will be used in this document, i.e. noun, pronoun, qualificative, verb, copulative, descriptive, ideophone, conjunction and interjection.

However, there is sub-categorisation within a word category in isiXhosa, as witnessed in the many qualificatives in The *Greater Dictionary of Xhosa: Volume 3* (Pahl et al., 1989:xxxvi) and parts of speech in Louw et al. (1984:16). This work uses the following hierarchy of isiXhosa categories:

Figure 3: Hierarchy of IsiXhosa Word Categories used

- Noun (isibizo)
- Pronoun (isimelabizo)
    - Absolute pronoun (esoqobo)
    - Demonstrative (esokukhomba/ esokwalatha/ isikhombisi/ isalathisi/)
    - Quantitative pronouns
    - Superlative (esobalaselo)
    - Possessive pronoun
- Qualificative (isichazi)
    - Adjective (Isiphawuli)
    - Relative (Isibaluli)
    - Possessive (Isimnini)
    - Enumerative (esobalo)
- Predicate (Isivisa)
    - Verb (isenzi)
    - Copulative (Isibanjalo)
        - Indicative copulative (isibayiyo)
        - Qualificative copulative (isibanjani/ isibanjalo sochazo/ isichazi)
        - Locative Demonstrative Copulative (isibandawo/ izibanjalo-zalathandawo)
        - Possessive copulative (esinesibandakanyi/ isibanayo)
        - Distributive (esinokwaba)
- Descriptive/Adverb (isihlomelo)
    - Locatives (Descriptive of place) (isalathandawo)
    - Descriptive of Manner (Isihlomelo sobunjani)
    - Descriptive of Time
- Ideophone (Isifanekisozwi)
- Conjunction (Isihlanganisi)
- Interjection (Isikhuzo)

The hierarchy of categories is important, because in each main category there are lemmatisation nuances specific to the sub-categories.

3.4 IsiXhosa Lemmas Details

This section describes in some detail the appropriate lemma for each word. As nouns are the lead words in an isiXhosa sentence (McLaren, 1948:14), they will be addressed first.

3.4.1 Nouns

IsiXhosa nouns are characterised by noun prefixes that expose the class of the noun (Pahl, 1982:9; McLaren, 1948:16; Louw et al., 1984:18). The noun class structure is a system of

classifying Bantu nouns according to noun class prefixes that usually distinguish between singular and plural forms. In this study Meinhof's (1932) classification is adhered to.

Table 3: Noun Class Prefixes

Class	Singular		Plural
Class 1	*um-*	Class 2	*aba-*
1a	*u-*	2a	*oo-*
3	*um-*	4	*imi-*
5	*ili-*	6	*ama-*
7	*isi-*	8	*izi-*
9	*in-*	10	*izin-*
11	*ulu-*	10	*izin-*
14	*ubu-*		
15 and 17	*uku-*		

This noun class structure permeates isiXhosa as far as other word categories relate to nouns, as will be evident in the rest of the document. This is referred to as concordial agreement (Mncube, n.d:13; McLaren, 1948:16; Louw et al., 1984:17). McLaren (1948:27) presents the base of these concords as the noun prefix proper. Louw et al. (1984:) refers to these base concords as the basic prefix. The base prefixes are listed below.

Table 4: Noun Prefixes Proper/Basic Prefix

Class	Singular		Plural
Class 1 and 1a	*mu-*	Class 2 and 2a	*ba-*
3	*mu-*	4	*mi-*
5	*li-*	6	*ma-*
7	*si-*	8	*zi-*
9	*n-*	10	*zin-*
11	*lu-*	10	*zin-*
14	*bu-*		
15 and 17	*ku-*		

Other concords in isiXhosa are based on these basic prefixes, e.g. adjective, relative, copulative and enumerative.

Except for class 2a, one can see that the noun basic prefix is simply a drop of the pre-prefix from the noun class prefixes. Pahl (1982:9) and Louw et al. (1984: 19) define the pre-prefix as the initial vowel preceding the basic prefix, using the following examples:

u+m(u) > um, in umntu [a person]

i+li > ili, in ilitye [a stone]

a+ma > ama, in amanzi [water]

i+si > isi, in isitya [a dish]

Nouns can also be inflected with suffixes for diminution, exaggeration, etc. The only discernible common feature in an isiXhosa noun paradigm is a noun stem. Therefore, the appropriate lemma for an isiXhosa noun is the noun stem.

Umntu, abantu -> ntu
[A person, persons -> person]

Ilizwe, amazwe, isizwe -> zwe
[A nation, nations, nationhood -> nation]

IsiXhosa nouns can take an augmentative suffix -*kazi* or diminutive suffix, e.g. –*ana.* The lemma in such cases remains the noun stem as shown below.

Umntu + ana = Umntwana -> ntu
[A person + <Diminutive suffix> = child -> person]

Umthi + kazi = Umthikazi -> thi
[A tree + <augmentative suffix> = large tree -> tree]

Iindaba + ana = iindatyana -> daba
[News + <diminutive suffix> = minor news -> news]

3.4.1.1 Noun Concords

IsiXhosa words are characterised by their concordial element which specifies the noun class to which they refer (Louw et al. 1984: 57). The concordial element is a prefixal morpheme that relates the word to a noun class. This referential characteristic allows the word to be used in a place of a noun, a pronominal character.

These concordial morphemes are normally bound to stems of other word categories resulting in the word, e.g. "***ndi***-*vile*" [**I** heard], for subject concords, and *"u-**ndi**-vile*" [he/she heard **me]**, for object concord. These concords are a closed group. Table 5 and Table 6, are lists of subject and object concords.

There are other concords in isiXhosa, but those will be handled in the appropriate word category section. Of note is that a word with concordial agreement with a noun in isiXhosa allows for the omission of the noun in the use of the language, e.g.

u-hambile (verb)
he/she is gone

ba-hle (adjective)
they are beautiful

ndi-segumbini (locative)
I am in the room

Table 5: Subject Concords

Person/Class	Singular		Plural
1st Person	*ndi-*		*si-*
2nd person	*u-*		*ni-*
3rd person			
Class 1 and 1a	*u-*	Class 2 and 2a	*ba*
3	*u-*	4	*i-*
5	*li-*	6	*a-*
7	*si-*	8	*zi-*
9	*i-*	10	*zi-*
11	*lu-*	10	*zi-*
14	*bu-*		
15 and 17	*ku-*		

Table 6: Object Concords

Person/Class	Singular		Plural
1st Person	*-ndi-*		*-si-*
2nd person	*-ku-*		*-ni-*
3rd person			
Class 1 and 1a	*-m-*	Class 2 and 2a	*-ba*
3	*-wu-*	4	*-yi-*
5	*-li-*	6	*-wa-*
7	*-si-*	8	*-zi-*
9	*-yi-*	10	*-zi-*
11	*-lu-*		
14	*-bu-*		
15 and 17	*-ku-*		

This section does not define lemmas per se, but provides reference material for other sections of this chapter.

3.4.2 Pronouns

The following sub-categories of isiXhosa pronouns will be dealt with, i.e. absolute pronouns, demonstrative pronouns, quantitative pronouns and superlatives.

3.4.2.1 Absolute Pronouns

Absolute pronouns are a closed group of full words related to the noun classes they represent. According to Pahl (1982:37) and Louw et al. (1984:59) all absolute pronouns have the stem –na, e.g. thi-na, wo-na, so-na, ko-na, ye-na. The list of absolute pronouns is shown below.

Table 7: IsiXhosa Absolute Pronouns

Person/Class	Singular		Plural
1st Person	*m-na*		*thi-na*
2nd person	*we-na*		*ni-na*
3rd person			
Class 1 and 1a	*ye-na*	Class 2 and 2a	*bo-na*
3	*wo-na*	4	*yo-na*
5	*lo-na*	6	*wo-na*
7	*so-na*	8	*zo-na*
9	*yo-na*	10	*zo-na*
11	*lo-na*	10	*zo-na*
14	*bo-na*		
15 and 17	*ko-na*		

Because the absolute pronouns are a closed group, one could use the whole absolute pronoun as the lemma for itself or the stem *-na*. For this study, absolute pronouns will lemmatise to their stem *-na*.

Absolute pronouns are used as stems in the formation of some locatives, possessives, copulatives and adverbs (Pahl, 1982:38). In such cases, the absolute pronoun stem is dropped as shown below:

```
Na+yena > naye
Ya+zona > yazo
Nga+bona > ngabo
```

These other word categories will, however, be handled later in the document.

3.4.2.2 Demonstrative Pronouns

IsiXhosa also uses demonstrative pronouns. There are three types of demonstrative pronouns, i.e. for demonstrating at positions here [this], near [that] and at a distance [that yonder], e.g. *lo, lowo, lowaa* (Louw et al., 1984:60; McLaren, 1948:57; Pahl, 1982:39). These are also a closed set depending on the class of the related noun. The full list of demonstrative pronouns is shown below.

Table 8 : Demonstrative Pronouns from Louw et al. (1984:61)

Location :	this	there	That yonder
3rd Person			
Class 1 and 1a	*lo*	*lowo/loo*	*lowa/laa/lowaa*
2 and 2a	*aba*	*abo*	*abaya/abaa/aabayaa*
3	*lo*	*lowo/loo*	*lowa/laa/lowaa*
4	*le*	*leyo/loo*	*leya/laa/leyaa*
5	*eli*	*elo*	*eliya/elaa/eeliya*
6	*la*	*lawo/loo*	*lawa/laa/lawaa*
7	*esi*	*eso*	*esiya/esaa/eesiyaa*
8	*ezi*	*ezo*	*eziya/ezaa/eeziyaa*
9	*le*	*leyo/loo*	*leya/laa/leyaa*
10	*ezi*	*ezo*	*eziya/ezaa/eeziyaa*
11	*olu*	*olo*	*oluya/olwaa/ooluyaa*
14	*obu*	*obo*	*obuya/obaa/oobuyaa*
15	*oku*	*oko*	*okuya/okwaa/ookuyaa*
17	*apha*	*apho*	*phaya/phayaa*

Demonstratives are formed of the demonstrative element *la* and the class concord. As they do not have a stem and they are a closed set, it follows that the full word form should therefore be the lemma for demonstrative pronouns.

3.4.2.3 Quantitative Pronouns

Quantitative pronouns are characterised by the stems: *-nke, -dwa* and *–numera*l (Louw et al., 1984:68; Pahl, 1982:43). Examples are:

Table 9: Quantitative Pronouns

Quantitative stem	Examples	Translated
-nke	*so-nke*	all of us
	zo-nke:	all of them
-dwa	*ye-dwa:*	only him
	so-dwa:	us alone
-numeral	*so-babini:*	the two of us
	zo-mbini:	the two of them

Quantitative pronouns derived from ***–nke*** are known as Inclusive Quantitatives; those derived from ***–dwa*** are known as Exclusive Quantitatives and those using ***-numeral*** are known as Inclusive Numerals (Pahl et al., 1989:700).

For the quantitative pronoun, the lemma should be the stem, e.g.

so-nke -> nke

zo-nke -> nke

ye-dwa -> dwa

so-dwa -> dwa

so-babini -> bini

zo-ntathu -> thathu

3.4.2.4 Differentiative Pronouns

Differentiative pronouns are derived from differentiative qualificatives with stems ***–mbi*** [another] and ***–phi***, [which of these] (McLaren, 1948:67). The pronoun forms are the same as the differentiative qualificatives for those derived from the stem *–mbi*. However, for those derived from the qualificative stem *–phi*, the prefixes *a-, e-* and *o-* are added (Pahl et al., 1989:703). e.g.:

Akukho wumbi[There is no other].

*Ufuna e-siphi?[*which ones do you want?]

Sobona a-baphi?[which ones will we see?]

Ubone o-mphi? [which one did you see?]

The differentiative pronouns are a closed set that is related to the noun class.

Table 10: Differentiative Pronouns

Class	Singular		Plural
Class 1 and 1a	wumbi owuphi	Class 2 and 2a	bambi abaphi
3	wumbi owuphi	4	yimbi eyiphi
5	limbi eliphi	6	wambi awaphi
7	simbi esiphi	8	zimbi eziphi
9	*yimbi eyiphi*	10	zimbi eziphi
11	lumbi oluphi	10	zimbi eziphi
14	bumbi obuphi		
15 and 17	bumbi okuphi		

Because differentiative pronouns have distinct stems, they should lemmatise to that stem.

3.4.2.5 Superlative Pronouns

Superlatives or emphatic absolute pronouns are formed from absolute pronouns and are for emphasis (Louw et al., 1984:74). This pronoun shows that the particular noun is highlighted in

some manner (Pahl, 1982:39). The superlatives are specific to a noun class, only exist for third person and are shown in the table below:

Table 11: Superlative Pronouns (Louw et al., 1984:74; Pahl, 1982:39; Pahl et al., 1989:690)

Person/Class	Singular		Plural
3rd person			
Class 1 and 1a	oyena	Class 2 and 2a	abona
3	owona	4	eyona
5	elona	6	owona
7	esona	8	ezona
9	eyona	10	ezona
11	olona	10	ezona
14	obona		
15 and 17	okona		

Because superlative pronouns are a closed set, their lemma could be the full word form, or the absolute pronoun stem *–na*. For this exercise the absolute pronoun stem *–na* is chosen.

3.4.3 Qualificatives

Qualificatives describe or qualify a substantive. The difference in the types of qualificatives is in the origin of their stem. Except for adjectives and enumeratives, all the stems of qualificatives come from another word category like a verb or pronoun.

3.4.3.1 Adjectives

The stem of an adjective carries the characteristic of the substantive e.g. *–de* [height or length], or *–hle* [beauty or goodness].

Adjectives are characterised by a noun class concord prefix and a stem, e.g. ***om****-de* [tall one] , ***om****-futshane* [the short one], ***aba****-de* [the tall ones], ***aba****-futshane* [the short ones].

Negations adds a circumfix (Louw et al., 1984:81)., e.g.:

ongam*–d–****anga*** [the one who's not tall],

ongam*–funtshan–****anga*** [the one who's not short],

olungelu*–d–****anga*** [who is not tall]

The table below shows the adjective concords:

Table 12: List of Adjective Concords (Louw et al., 1984:77)

Class	Singular		Plural
Class 1 and 1a	*om-*	Class 2 and 2a	*aba-*
3	*om-*	4	*emi-*
5	*eli-*	6	*ama-*
7	*esi*	8	*ezi-*
9	*en-*	10	*ezin-*
11	*olu-*	10	*ezin-*
14	*obu-*		
15 and 17	*oku-*		

When an adjective is preceded by a demonstrative which precedes the noun being qualified, the adjective loses the initial vowel of the adjective concord (Louw et al., 1984:80), e.g.:

Loo mntu ***m****-hle*[that beautiful person]

Esi sitya ***si****-hle*[this nice dish]

The same happens when the adjective is used with a short negative verb (Louw et al., 1984:81) e.g.:

Andiboni bantu ***ba-****ninzi*[I don't see any people]

According to Pahl (1982:46) and Louw et al. (1984:78) there are only 19 adjective stems in isiXhosa and they are:

Table 13: isiXhosa Adjective Stems (Pahl, 1982:46; Louw et al., 1984:78)

-bi [ugly]	*-de* [long]	*-dala* [old]	*-khulu* [big]
-hle [beautiful]	*-fuphi/-futshane* [near]	*-tsha*[new]	*-ncini/ -ncinane/ -ncakane* [small]
-ni? [which]	*-ngaphi?* [how many?]	*-ninzi/ -nintsi/ -nintshi/ -ninji:* [many]	*-nje* [only]
-nye [alone]	*-nye* [one]	*-bini* [two]	*-thathu* [three]
-ne [four]	*-hlanu*: [five]	-thandathu [six]	

The lemma for adjectives should therefore be the adjective's stem as it is the persistent part in an adjective paradigm, e.g.:

Aba-futshane -> futshane

3.4.3.2 Relative

Relatives serve the same function as adjectives; the difference between the two is in morphology. The relative does not have a closed set of stems but may be formed from relative stems and from stems of other word categories (Louw et al., 1984:83). Therefore, a relative is a noun qualifier that may inherit stems from other word categories. This is done by prefixing a stem from another word category with the relative concord.

The relative concords for the different noun classes are listed below:

Table 14: Relative Concords (Pahl et al., 1989:685; Louw et al., 1984:84)

Person/Class	Singular		Plural
1st Person	*endi-*		*esi-*
2nd person	*o-*		*eni-*
3rd person			
Class 1 and 1a	*o-*	Class 2 and 2a	*aba-*
3	*o-*	4	*e-*
5	*eli-*	6	*a-*
7	*esi-*	8	*ezi-*
9	*e-*	10	*ezi-*
11	*olu-*	10	*ezi-*
14	*obu-*		
15 and 17	*oku-*		

The relative concord is used to form relatives from:

- Relative stems, e.g. ***obu****-mnyama* [that is black];
- Noun stems forming a copulative, e.g. ***ongu****-mthuthi* [that is a courier];
- Possessive forming a copulative, e.g. ***ela****-bo* [theirs];
- Descriptive stem as a copulative, e.g. ***el****-apha* [that is here];
- A connective, also forming a copulative, e.g. ***endi****-nayo* [that I possess].

For the purposes of a relative, it is best to choose the stem as the lemma. This does mean that relatives that derive from other word categories would have the same lemma as the other word category, e.g.:

> ***ongu****-mthuthi* [who is a courier], *aba-thuthi* [couriers] > *thuthi* [courier]

3.4.3.3 Enumerative

Louw et al. (1984:89), the *Greater Dictionary of Xhosa:* Volume 3 (Pahl et al., 1989:xxxvi) and Pahl (1982:45) consider that the enumerative is a qualificative that usually follows the substantive which it qualifies. However, the same *Greater Dictionary of Xhosa* lists the stems ***–mbi*** and "***-phi*?"** as both qualificative and pronouns in Addendum 15 (Pahl et al., 1989:702).

To the two stems listed above, Louw et al (1984) adds two more stems to the list of enumeratives, i.e. ***–nye*** and "***–ni*?"**

The stems ***–mbi*** and "***–phi*?"** are adequately covered under section *3.4.2.4: Differentiative Pronouns*, so the only consideration is for the latter two stems. The enumerative stems ***–nye*** and "***-ni***?" form words by being prefixed with the basic class prefix. This results in *–nye* being expressed as shown in Table 15, and "*-ni*?" being expressed as in Table 16.

Table 15: Enumeratives Based on *-nye*

Class	Singular		Plural
Class 1 and 1a	*omnye mnye*	Class 2 and 2a	*abanye banye*
3	*omnye mnye*	4	*eminye minye*
5	*elinye linye*	6	*amanye manye*
7	*esinye sinye*	8	*ezinye zinye*
9	*enye nye*	10	*ezinye zinye*
11	*olunye lunye*	10	*ezinye zinye*
14	*obunye bunye*		
15 and 17	*okunye kunye*		

Table 16: List of Enumeratives Based on –ni?

Class	Singular		Plural
Class 1 and 1a	*omni? mni?*	Class 2 and 2a	*abani? bani?*
3	*omni? mni?*	4	*emini? mini?*
5	*elini? lini?*	6	*amani? mani?*
7	*esini? sini?*	8	*ezini? zini?*
9	*eni? ni?*	10	*ezini? zini?*
11	*oluni? luni?*	10	*ezini? zini?*
14	*obuni? buni?*		
15 and 17	*okuni? kuni?*		

Table 17: Enumeratives based on –mbi and "-phi?"

Class	Singular	Class	Plural
Class 1 and 1a	*wumbi owuphi?*	Class 2 and 2a	*bambi abaphi?*
3	*wumbi owuphi?*	4	*yimbi eyiphi?*
5	*limbi eliphi?*	6	*wambi awaphi?*
7	*simbi esiphi?*	8	*zimbi eziphi?*
9	*yimbi eyiphi?*	10	*zimbi eziphi?*
11	*lumbi oluphi?*	10	*zimbi eziphi?*
14	*bumbi obuphi?*		
15 and 17	*bumbi okuphi?*		

Because both stems result in closed sets, enumeratives could either lemmatise to full form or the enumerative stem, *-mbi*, *-nye, "-phi?"* or "*–ni?".* For this study, the enumerative stem is chosen.

3.4.3.4 Possessive

To Louw et al. (1984:99), the *Greater Dictionary of isiXhosa:* Volume 3 (Pahl et al., 1989:xxxvi) and Pahl (1982:45), the possessive is a qualificative in that it qualifies either a noun or a pronoun with regard to possession.

Possessives modify a substantive with another substantive via a possessive concord. One is the possessor, the other the possessed. It is the possessor that is prefixed with the possessive concord.

Table 18: List of Possessive Concords (Louw et al., 1984:100)

Class	Singular		Plural
Class 1 and 1a	*wa-*	Class 2 and 2a	*ba-*
3	*wa-*	4	*ya-*
5	*la-*	6	*a-*
7	*sa*	8	*za-*
9	*ya-*	10	*za-*
11	*lwa-*	10	*za-*
14	*ba-*		
15 and 17	*kwa-*		

Examples are:

Umntu **we**-nkosi (wa + inkosi)[the chief's person]

Umfazi **wo**-mfundisi (wa + umfundisi)[the minister's wife]

Ukutya **ko**-mntu (kwa + umntu)[the person's food]

Possessives from pronouns are rather unique in that the absolute pronoun loses its stems, *–na*, when it converts to a possessive pronominal stem.

The possessive pronominal stems are listed in the table below:

Table 19: Possessive Pronominal Stems (Pahl et al., 1989:690)

Person/Class	Singular		Plural
1st Person	*-m*		*-ithu*
2nd person	*-kho*		*-inu*
3rd person			
Class 1 and 1a	*-khe*	Class 2 and 2a	*-bo*
3	*-wo*	4	*-yo*
5	*-lo*	6	*-wo*
7	*-so*	8	*-zo*
9	*-yo*	10	*-zo*
11	*-lo*	10	*-zo*
14	*-bo*		
15 and 17	*-ko*		

With possessive prefixes and possessive pronominal stems, one is able to generate a list of pronominal possessives. Such a list is tabled in Addendum 9: Pronominal Possessives of the *Greater Dictionary of Xhosa:* Volume 3 (Pahl et al., 1989:691).

With regards to selecting a lemma, the possessives could lemmatise to full form because they are a known closed set or possessives stem. The researcher chose the possessive stem in this study. Possessives derived from noun stems should be lemmatised to the noun stem, e.g.:

wenkosi [of the chief] > *nkosi* [chief]

yakhe[his/hers] > *khe*

yabo [theirs] > *bo*

3.4.4 Predicates

The predicate is the word that carries the central meaning of a sentence. In isiXhosa it “can be the statement, the relation or the declaration” (Louw et al. 1984: 111). In isiXhosa there are two types of predicates, i.e. verbs and copulatives.

Predicates are always used in relation to a substantive, be it a subject or an object. In isiXhosa this relation is reflected in the concord, the subject concord if related to the subject or object concord if related to the object. These are listed in Table 5 and Table 6.

Examples are:

	Subject Concord	Object Concord
Verbs	***u**-tyile* [he/she ate]	*u-**yi**-tyile* [he/she ate it]
Copulatives	***ku**-kutya* [it is food]	

Each of the isiXhosa predicates have complex structures and should therefore be handled individually.

3.4.4.1 Verbs

Verbs in isiXhosa are complex. They contain morphemes for person, number, class, mood, tense and form (McLaren, 1948:80), e.g. *ndi-**khawulez**-is-ile* [I rushed], *be-si-**khawulez**-is-ana* [we rushed each other], *a-ka-**khawuleza*** **[**he/she didn't rush]**,** *ba-ku-**khawilez**-is-wa* [they will be rushed].

IsiXhosa verb inflection can occur as a result of verb extension, conjugation, change in verb form, mood, aspect and tense. The paragraphs below give examples of the above.

3.4.4.1.1 Verb Extensions

IsiXhosa verbs can be extended to show relationships between the subject and the object and the intensity of the action. The isiXhosa extensions are: Applied, Neuter, Causative, Connective, Intensive, Passive and Reduplication (Louw et al. 1984: 122; Pahl 1982: 81)

Table 20: Verb Extension examples

Extension Type	Verb Examples
Basic verb stem	*bamb-a* [hold]
With applied extension	*bamb-**el-**a* [hold for]
With Neuter extension	*bamb-**ek-**a* [able to hold to]
With Causative extension	*bamb-**is-**a* [hold with]
With Connective extension	*bamb-**an-**a* [hold each other]
With Intensive extension	*bamb-**isis-**a* [hold tight]
With Passive extension	banj-***w-**a* [held]
With Reduplicated stems extension	*bamba-**bamba*** [hold many times]

As one can see above, stem is the part that persists among extensions.

3.4.4.1.2 Conjugation

The negation of a verb in isiXhosa is expressed with a bound morpheme, e.g. ***aka**-hamb-**ang**-a* [he/she did not go]. This is referred to as conjugation (Louw et al. 1984: 132). The positive conjugation of the above example would be ***u**-hamb-**ile*** [he/she went]. In verb conjugation the verb root persists.

3.4.4.1.3 Forms

IsiXhosa verbs have two forms, the potential and temporal forms (Louw et al. 1984: 133). The potential form of a verb indicates an ability, e.g. *ndi-**nga**-hamba* [I **can** go]. The temporal form indicated "when the action of the verb takes place", e.g. ***sa**-ku-hamba* [**when** we go]. Both forms have negation, e.g. ***a**-ndi-**nge**-hamb-**i*** [I **cannot** go], ***sa**-ku-**nga**-hamb-**i*** [**when** we **didn't** go].

Among the different forms of a verb the root, in the above case *hamb-*, is the most persistent morpheme of the verbs forms.

3.4.4.1.4 Moods

Verbs in isiXhosa have five moods. Verb moods express the relationship between the speaker and his/her view of the reality expressed by the predicate (Louw et al. 1984: 133). Even though mood is a semantic issue, it expresses itself in the morphology of the word. The **infinitive** and **imperative** moods are considered infinite moods, and the **indicative**, **participial**, and **subjunctive** moods are considered finite moods.

The main characteristic of infinite moods is that they do not have a subject concord and therefore do not relate to a subject of a sentence. An example is the infinitive verb, *uku-bona* [to see]. In addition, the infinitive verb has no indication of tense. An infinitive verb can act as both a noun and a verb. An imperative verb on the other hand does indicate tense but only the present tense, e.g. *bona* [see].

All the finite moods have subject concords. The indicative mood simply makes a statement, e.g. *u-ya-hamba* [he/she goes]. The participial specifies the conditions under which the predicate occurs ,e.g. *uhamba **e-cela*** [he/she goes begging]. The subjunctive shows an imagined reality, e.g. *uhambela ukuba **a-bonakal-e*** [he/she go to be noticed].

In all the verb moods, the root of the verb is the most pervasive morpheme.

3.4.4.1.5 Aspect

Aspect is a qualification of tense. It indicates if an action is complete or not, continuous or not or progressive or not (Louw et al. 1984: 135). There are two aspects therefore, progressive and exclusive. An example of a progressive verb is *u-sa-tya* [he/she is still eating], and the example of an exclusive verb is *se-le-ty-ile* [he/she has eaten].

The root again is the only morpheme that is unchanged among verbs aspect.

3.4.4.1.6 Tense

IsiXhosa verbs can fall in one of six tenses. The isiXhosa verb tenses are present, future, perfect, the remote past, the continuous past and the contingent tense (Louw et al. 1984: 134).

In most tenses the positive conjugation of the verb ends in the terminative morpheme *–a,* except for the perfect tense which uses the suffix *–ile*. Examples are *u-ya-bon-**a*** [he is seeing] and *u-bon-**ile*** [he saw] respectively. Most tenses have a negative conjugation which terminates with *–i.* The morphemes *–ka-* and *–nga-* are also used in negation.

IsiXhosa words have two forms per tense, the short form and the long form, e.g. ***ndi**-bona* [I see] and ***ndi-ya**-bona* [I see] respectively. The long form is formed by appending the morpheme –ya- to the subject concord.

The form of an isiXhosa verb in a particular tense is made of prefixes and suffixes that are defined by the conjugation, form, mood and aspect of the verb. The table below gives examples of isiXhosa verbs for different tenses.

Table 21: Examples of different isiXhosa tenses

Tense	Positive example	Negative example
Present tense	*ndi-**bon**-**a*** [I see], Ndi-ya-**bon**-a [I see]	*a-ndi-**bon**-i* [I don't see]
Future tense	*ndi-za ku-**bon-a*** [I will see]	*andi-zi-ku-**bon-a*** [I will not see]
Perfect tense	*ndi-**bon**-ile* [I saw]	*a-ndi-**bon**-ang-**a*** [I did not see]
Remote past tense	*wa-a-**bon**a* [I saw]	*a-ka-**bon**-ang-**a** **l**akazange wa-**bona*** [I did not see],
Continuous past tense	*ba-be-**bona*** [they used to see]	*ba-be-nga-**bon**-i* [they did not see]
Contingent tense	*bendiza ku-**bona*** [I was going to see]	*Bendingazi ku-**bona*** [I was not going to see]

The above table shows a few of the isiXhosa verb tenses. The table is not exhausting of all the different combinations of isiXhosa verb tenses, moods, forms etc. However it gives a good indication that the verb root is the central and pervasive morpheme in an isiXhosa word across the different tenses.

In Table 21 one should also notice verbs that precede other verbs. These are referred to as auxiliary verbs and should be treated the same as other verbs.

3.4.4.1.7 Conclusions on Verbs

As shown above isiXhosa verbs are complex, but are consistent in retaining the root of the verb. The common morpheme in a verb paradigm is therefore, the verb root. Therefore, a lemma for the isiXhosa verb should be the verb stem as it is the closest word to the verb root.

3.4.4.2 Copulatives

Copulatives are non-verbal predicates that specify what nature a substantive is, e.g. *yindlu* [it is a house]. Copulatives are derived from nouns, absolute pronouns, qualificatives, descriptives and conjunctions (Louw et al., 1984:216). In isiXhosa, copulatives have the potential to be a sentence (Mncube, n.d:59). It is worth noting that copulatives can be formed from quantitatives as well, however, these are expressed as word phrases, not as individual words (Louw et al., 1984:227). Because this study concerns itself with word lemmas, quantitative copulatives will not be considered because they are phrases.

Because copulatives are predicates, they have a positive as well as negative conjugation. IsiXhosa copulatives also have a personal and an impersonal form. These forms will be shown below for each copulative type.

Substantively derived copulatives are identificative copulatives (Louw et al., 1984:216), identifying the substantive regarding type, status and other qualities.

This section starts with pronoun-derived copulatives because pronoun derived copulatives are used in the negation of other copulatives.

3.4.4.2.1 Pronoun Derived Copulatives

Copulatives from absolute pronouns are formed by prefixing the absolute pronoun with the copula. The copula is a morpheme which connects a subject with any predicate other than a verb (McLaren, 1948:44) . There are different copulas for the different noun class pronouns, as shown in Table 22.

The absolute pronoun stem ***–na*** is dropped resulting in the list of absolute pronoun copulatives listed in Table 23.

Copulative negation is done by prefixing the positive copulative with a negation prefix *asi-* or *ayi-*, depending on the dialect of isiXhosa spoken (Pahl, 1982:167).

Table 22: Copula (Louw et al., 1984:220)

Person/Class	Singular		Plural
1st Person	*ndi-*		*si-*
2nd person	*ngu-*		*ni-*
3rd person			
Class 1 and 1a	*ngu-*	Class 2 and 2a	*nga-*
3	*ngu-*	4	*yi-*
5	*li-*	6	*nga-*
7	*si-*	8	*zi-*
9	*yi-*	10	*zi-*
11	*lu-*		
14	*bu-*		
15 and 17	*ku-*		

Table 23: Absolute Pronoun Derived Copulatives (Pahl, 1982:167; Louw et al., 1984:220)

Person/Class	Singular Positive	Negative	Plural Positive	Negative
1st Person	*ndi-m*	*asindi-m/ ayindi-m*	*si-thi*	*asisi-thi/ ayisi-thi*
2nd person	*ngu-we*	*asingu-we/ ayingu-we*	*ni-ni*	*asini-ni/ ayini-ni*
3rd person				
Class 1 and 1a	*ngu-ye*	*asingu-ye/ ayingu-ye*	*nga-bo*	*asinga-bo/ ayinga-bo*
3	*ngu-wo*	*asingu-wo/ ayingu-wo*	*yi-yo*	*asiyi-yo/ ayiyi-yo*
5	*li-lo*	*asili-lo/ ayili-lo*	*nga-wo*	*asinga-wo/ ayinga-wo*
7	*si-so*	*asisi-so/ ayisi-so*	*zi-zo*	*asizi-zo/ ayizi-zo*
9	*yi-yo*	*asiyi-yo/ ayiyi-yo*	*zi-zo*	*asizi-zo/ ayizi-zo*
11	*lu-lo*	*asilu-lo/ ayilu-lo*	*zi-zo*	*asizi-zo/ ayizi-zo*
14	*bu-bo*	*asibu-bo/ ayibu-bo*		
15 and 17	*ku-ko*	*asiku-ko/ ayiku-ko*		

Personal copulative pronouns are derived by prefixing the copulative pronoun with the subject concord of the appropriate person, e.g.:

U+nguye > unguye `[you are him]`

Ndi-ndim > ndindim `[I am me]`

Ni+nini > ninini `[you (Plural) are you]`

Ndi-nguye > ndinguye `[I am him/her]`

Impersonal copulatives derived from absolute pronouns are derived by prefixing a subject concord (Table 5) to the copulative pronoun. Again the absolute pronoun stem *–na* is dropped.

Table 24: Absolute Pronoun Derived Impersonal Copulatives (Louw et al., 1984:222)

Class	Singular Positive	Negative		Plural Positive	Negative
3rd person Class 1 and 1a	*ungu-ye*	*akangu-ye*	Class 2 and 2a	*banga-bo*	*abanga-bo*
3	*ungu-wo*	*awungu-wo*	4	*iyi-yo*	*ayiyi-yo*
5	*lili-lo*	*alili-lo*	6	*anga-wo*	*akanga-wo*
7	*usi-so*	*akasi-so*	8	*bazi-zo*	*abazi-zo*
9	*ndiyi-yo*	*asiyi-yo/ ayiyi-yo*	10	*sizi-zo*	*asizi-zo/ ayizi-zo*
11	*ulu-lo*	*asilu-lo/ ayilu-lo*	10	*sizi-zo*	*asizi-zo/ ayizi-zo*
14	*ubu-bo*	*asibu-bo/ ayibu-bo*			
15 and 17	*iku-ko*	*asiku-ko/ ayiku-ko*			

Copulatives are a distinct word category and should therefore have their own lemmas. Absolute pronoun derived copulatives are a closed set, albeit a large one and can therefore be lemmatised to the full word form. Another alternative is to lemmatise them to the subject concord, because they have dropped the pronominal stem *-na*. This reduces the number of lemmas in this category whilst keeping the lemma for absolute pronoun derived copulatives distinct from the absolute pronoun lemma. This is the approach this study will take. Therefore, absolute pronoun derived copulatives will be lemmatised to their stems, e.g.:

Nguye, asinguye, ndinguye, singuye, asinguye > ye
`[It is him/her, it is not him/her, I am s/he, we are s/he, we]`

Zizo, asizizo/ayizizo, zizizo, sizizo, ndizizo > zo
`[It is them, it is not them, we are them, I am them]`

3.4.4.2.2 Nouns Derived Copulatives

Noun derived copulatives are formed by prefixing the noun stem with a prefix. This prefix is derived from the copula (c.f. The absolute pronoun stem ***–na*** is dropped resulting in the list of absolute pronoun copulatives listed in Table 23.

Copulative negation is done by prefixing the positive copulative with a negation prefix *asi-* or *ayi-*, depending on the dialect of isiXhosa spoken (Pahl, 1982:167).

Table 22 on page 49) and the noun prefix (c.f. Table 4 on page 34) (Louw et al., 1984:217; Pahl, 1982:167).

Table 25: Noun Derived Copulative Prefixes (Louw et al., 1984:220)

Person	Singular		Plural
1st Person	*ndi-*		*si-*
2nd person	*ngu-*		*ni-*
3rd person			
Class 1	*ngu-*	Class 2	*nga-*
1a	*ngu-*	2a	*ngoo-*
3	*ngu-*	4	*yi-*
5	*li-*	6	*nga-*
7	*si-*	8	*zi-*
9	*yi-*	10	*zi-*
11	*lu-*	10	*zi-*
14	*bu-*		
15 and 17	*ku-*		

Personal noun-derived copulatives are derived by prefixing the noun-derived copulative with the subject concord. Negation is done by prefixing the noun-derived copulative with the prefix **asi-** e.g.:

Positive		Negative	
Personal	Impersonal	Personal	Impersonal
ndi-ngumntu (I am human)	*ngumntu* (It's a human)	*Andi-ngomntu/andi-mntu* (I am no human)	*asi-ngomntu/ asi-mntu* (It's not a human)
Si-zizinja (we are dogs)	*yinja* (it is a dog)	*asi-zozinja* (we are not dogs)	*asi-yonja /asi-nja* (it's not a dog)
U-yihagu (s/he is a pig)	*yihagu* (it's a pig)	*aka-yohagu* (s/he is not a pig)	*asi-yohagu /asi-hagu* (it's not a pig)
Ba-zizinja (they are dogs)	*zizinja* (its dogs)	*aba-zozinja* (they are dogs)	*asi-zozinja/ asi-zinja* (it's not dogs)

Impersonal copulatives are derived by prefixing a subject concord to the noun copulative.

Impersonal negation is done by preceding a noun with an absolute pronoun derived copulative of the related class, e.g. *Asinguye umntu [It is not a person]*. (Louw et al., 1984:223; Pahl, 1982:167) This form is also expressed as *asingomntu /asimntu [It's no person]*. The original extended negation is not shown in the example above, as these would be two words that would be lemmatised individually. However, the contracted form is shown, as it would need to be lemmatised. The second impersonal negation shown in the above example shows an even more contracted impersonal negation noun derived copulative form. This second negation contracted form is formed by removing the noun concord.

To establish what the lemma is for a noun derived copulative, the following three criteria had to be used: "the same stem, the same major part-of-speech and the same word-sense". In this case, the three criteria lead to the noun stem because it is the only stem available that

maintains the word-sense. Therefore, the noun stem is the lemma of a noun-derived copulative. Examples are:

Personal noun-derived copulatives paradigm	lemma
Bazizinja [they are dogs], *abazozinja* [they are not dogs], *ndiyinja* [I am a dog]), *andiyonja* [I am not a dog], *uyinja* [you are a dog], *awuyonja* [you are not a dog], *yinja, asizozinja*	*nja* [dog]
bazihagu [they are pigs], *abazohagu* [they are not pigs], *ndiyihagu* [I am a pig], *andiyohagu* [I am not a pig], *uyihagu*[you are a pig], *awuyohagu*[you are not a pig]	*hagu* [pig]

3.4.4.2.3 Demonstrative Derived Copulatives

Copulatives are also formed from demonstratives by prefixing the demonstrative with the same copula used in absolute pronoun derived copulatives (c.f. The absolute pronoun stem ***–na*** is dropped resulting in the list of absolute pronoun copulatives listed in Table 23.

Copulative negation is done by prefixing the positive copulative with a negation prefix *asi-* or *ayi-*, depending on the dialect of isiXhosa spoken (Pahl, 1982:167).

Table 22) (Louw et al., 1984:224)

Table 26: Demonstrative Derived Impersonal Copulatives from Louw et al. (1984:225)

	Location: this	there	that yonder
Class 1 and 1a	*ngulo*	*ngulowo/nguloo*	*ngulowa/ngulaa/ngulowaa*
2 and 2a	*ngaba*	*ngabo*	*ngabaya/ngabaa/ngaaabayaa*
3	*ngulo*	*ngulowo/nguloo*	*ngulowa/ngulaa/ngulowaa*
4	*yile*	*yileyo*	*yileya/yileyaa*
5	*leli*	*lelo*	*leliya/lelaa/leeliya*
6	*ngala*	*ngalawo/ngaloo*	*ngalawa/ngalaa/ngalawaa*
7	*sesi*	*seso*	*sesiya/sesaa/seesiyaa*
8	*zezi*	*zezo*	*zeziya/zezaa/zeeziyaa*
9	*yile*	*yileyo*	*yileya/yilaa/yileyaa*
10	*zezi*	*zezo*	*zeziya/zezaa/zeeziyaa*
11	*lolu*	*lolo*	*loluya/lolwaa/looluyaa*
14	*bobu*	*bobo*	*bobuya/bobaa/boobuyaa*
15	*koku*	*koko*	*kokuya/kokwaa/kookuyaa*

Negation is achieved by prefixing the impersonal copulative with the prefix `asi-` (Louw et al. 1984:226)

Table 27: Demonstrative Derived Impersonal Negative Copulatives from Louw et al. (1984:226)

Location	this	there	that yonder

Class 1 and 1a	*asingulo*	*asingulowo/asinguloo*	*asingulowa/asingulaa/asingulowaa*
2 and 2a	*asingaba*	*asingabo*	*asingabaya/asingabaa/asingaaabayaa*
3	*asingulo*	*asingulowo/asinguloo*	*asingulowa/asingulaa/asingulowaa*
4	*asiyile*	*asiyileyo*	*asiyileya/asiyileyaa*
5	*asileli*	*asilelo*	*asileliya/asilelaa/asileeliya*
6	*asingala*	*asingalawo/asingaloo*	*asingalawa/asingalaa/asingalawaa*
7	*asisesi*	*asiseso*	*asisesiya/asisesaa/asiseesiyaa*
8	*asizezi*	*asizezo*	*asizeziya/asizezaa/asizeeziyaa*
9	*asiyile*	*asiyileyo*	*asiyileya/asiyilaa/asiyileyaa*
10	*asizezi*	*asizezo*	*asizeziya/asizezaa/asizeeziyaa*
11	*asilolu*	*asilolo*	*asiloluya/asilolwaa/asilooluyaa*
14	*asibobu*	*asibobo*	*asibobuya/asibobaa/asiboobuyaa*
15	*asikoku*	*asikoko*	*asikokuya/asikokwaa/asikookuyaa*

Personal demonstrative copulatives are formed from impersonals by prefixing with the subject concord (Louw et al., 1984:226), e.g.:

> *Ungulo* [he/she this one], *ungulowa* [you are that one], *sileli* [we are this one]

Negation of personal demonstrative copulatives is achieved by prefixing the negative **a-** morpheme to the subject concord, e.g.:

> *Akangulo* [he/she is not this one], *awungulowa* [you are not that one], *asileli* [we are not this one]

The lemma of the demonstrative copulatives is the demonstrative itself. This maintains the word sense.

3.4.4.2.4 Quantitative Derived Copulatives

Quantitatives can be used to form copulatives by prefixing them with the subject concord (Louw et al., 1984:9228), e.g.:

> *babodwa* [they are alone]
>
> *ababodwa* [they are not alone]

For quantitative derived copulatives, the lemma should be the quantitative stem as it maintains the word-sense of the paradigm.

3.4.4.2.5 Qualificative Derived Copulatives

Copulatives can be formed from three types of qualificatives, i.e. adjective stems, relative stems and enumerative stems (Louw et al., 1984:216).

3.4.4.2.5.1 Adjective stems derived copulatives

Copulatives are formed from adjective stems by prefixing the stem with the copula (Louw et al., 1984:). Personal forms are further formed by prefixing the derived copulative with the subject concord when forming the first and second persons. Negation is formed by prefixing the negative a- followed by the subject concord and the copula to the adjective stem. The table below shows personal adjectival stem copulatives derived from the *–khulu* [big] adjective stem.

Table 28: Adjective stem derived copulatives (Pahl 1982: 171; Louw et al. 1984: 220)

Person/ Class	Singular Positive	Negative		Plural Positive	Negative
1st Person	*ndi-m-khulu*	*a-ndi-m-khulu*		*si-ba-khulu*	*a-si-ba-khulu*
2nd person	*u-m-khulu*	*a-ku-m-khulu*		*ni-ba-khulu*	*a-ni-ba-khulu*
3rd person					
Class 1 and 1a	*m-khulu*	*a-ka-m-khulu*	Class 2 and 2a	*ba-khulu*	*a-ba-ba-khulu*
3	*m-khulu*	*a-wu-m-khulu*	4	*mi-khulu*	*a yi-mi-khulu*
5	*li-khulu*	*a-li-li-khulu*	6	*ma-khulu*	*a-ka-makhulu*
7	*si-khulu*	*a-si-si-khulu*	8	*zin-kulu*	*a-zi-zin-kulu*
9	*in-kulu*	*a-yin-kulu*	10	*zin-kulu*	*a-zi-zin-kulu*
11	*lu-khulu*	*a-lu-lu-khulu*	10	*zin-kulu*	*a-zi-zin-kulu*
14	*bu-khulu*	*a-bu-bu-khulu*			
15 and 17	*ku-khulu*	*a-ku-ku-khulu*			

Impersonal copulatives are formed adjectives by prefixing them with the same copula as used in nouns (Louw et al., 1984:230), e.g.:

```
Ngo-m-dala [it's the old person]

Ye-mi-dala [it's the old ones]
```

Negation is achieved in the same manner as the noun (Louw et al., 1984:230; Pahl, 1982:171),e.g.:

```
Asi-ng-om-dala [it’s not the old one]

Asi-ng-aba-dala [It’s not the old ones]
```

For a particular adjective, the adjective stem is the only consistent part in the adjective stem copulative paradigm, making the adjective stem the lemma.

3.4.4.2.5.2 Relative stem derived copulatives

Personal positive copulatives from relative stems are formed by prefixing them directly with the subject concord. Personal negative copulatives are formed by prefixing the positives relatives stem formed copulative with the negative `a-` (Louw et al., 1984:230; Pahl, 1982:171).

Table 29: Relative Stem Derived Copulatives (Pahl 1982: 171; Louw et al. 1984: 230)

Person/Class	Singular Positive	Negative		Plural Positive	Negative
1st Person	*ndi-banzi*	*a-ndi- banzi*		*si- banzi*	*a-si- banzi*
2nd person	*u-banzi*	*a-ku-banzi*		*ni- banzi*	*a-ni- banzi*
3rd person					
Class 1 and 1a	*u-banzi*	*a-ka- banzi*	Class 2 and 2a	*ba-banzi*	*a-ba- banzi*
3	*u-banzi*	*a-wu- banzi*	4	*i-banzi*	*a-yi-banzi*
5	*li-banzi*	*a-li- banzi*	6	*a-banzi*	*a-ka-banzi*
7	*si-banzi*	*a-si- banzi*	8	*zi- banzi*	*a-zin- banzi*
9	*i-banzi*	*a-yi- banzi*	10	*zi- banzi*	*a-zi- banzi*
11	*lu- banzi*	*a-lu- banzi*	10	*zi- banzi*	*a-zi- banzi*
14	*bu-banzi*	*a-bu-banzi*			
15 and 17	*ku-banzi*	*a-ku-banzi*			

Impersonal relative copulatives are derived from full relatives using the same copula as that prefixed to the noun (Louw et al., 1984:232), e.g.:

Ngo-bomvu [it's the red person]

Ye-mhlophe [it's the white one]

Ze-zi-mhlophe [it's the whine ones]

Impersonal negative is achieved by prefixing the impersonal relative copulative with *asi-*, e.g.:

Asi-ngo-mhlophe [it's not the white person]

Asi-ye-mhlophe [it's not the white one]

Asi-ze-zi-mhlophe [it's not the white ones]

From the above, it is clear that the only stable part in the relative stem formed copulative paradigm is the relative stem. The relative stem therefore must be the lemma of the copulative construction formed from the relative stem.

3.4.4.2.5.3 Enumerative derived copulatives

As mentioned before, there are five types of enumerative stems, i.e. –nye, -ni?, -phi, -mbi, and numeral stems (Louw et al. 1984: 232).

The impersonal copulatives for –nye and the numeral stems are derived by prefixing the enumerative stem with the subject concord. The personal copulative is derived by prefixing the enumerative stem with the subject concord to the impersonal form.

Table 30: Examples of Enumerative Stem Derived Copulatives

Person/ Class	Singular Positive	Negative		Plural Positive	Negative
1st Person	*ndi-m-nye*	*a-ndi-m-nye*		*si-ba-bini*	*a-si-ba-bini*
2nd person	*u-m-thathu*	*a-ku-m-thathu*		*ni-ba-ne*	*a-ni-ba-ne*
3rd person					
Class 1 and 1a	*m-ni*	*a-ka-m-ni*	Class 2 and 2a	*ba-phi*	*a-ba-ba-phi*
3	*m-mbi*	*a-wu-m-mbi*	4	*mi-phi*	*a-yi-mi-phi*
5	*li-hlanu*	*a-li-li-hlanu*	6	*ma-thandathu*	*a-ka-thandathu*

The lemma for the enumerative should be the enumerative stem, as it carries the meaning.

3.4.4.2.6 Descriptive Derived Copulatives

Descriptives form copulatives by having their stems prefixed with a subject concord. The prefix is a subject concord for locative derived positives and the negative a- followed by the subject concord for negative locative (Louw et al., 1984:236).

For descriptives the lemma should be the descriptive stem, as this carries the meaning.

3.4.4.2.7 Conjunction Derived Copulatives

Copulatives can also be formed from conjunctions by prefixations (Louw et al., 1984:239). The conjunction stem should be the lemma as it maintains the meaning.

3.4.4.2.8 Conclusions on Copulatives

Copulatives are derived from a number of word types. For some of the copulatives the root from which they are derived from persists. In this case, the stem from the source word category should be the lemma. In the case of copulatives derived from absolute pronouns where the absolute pronoun stem *–na* is dropped, the best one can do is carry the subject concord as the lemma of the copulative.

3.4.5 Descriptive

According to Louw et al. (1984:24), a descriptive is a word that describes a predicate or other descriptive in terms of manner, time and place (locatives).

Descriptive of manner are derived from adjective stems, relative stems, nouns and pronouns by prefixing then with the descriptive prefixes, *ka–* , *kaku–*, *na–*, *kuna–*, *nga–*, *njenga–*, and *nganga–*. (Louw et al., 1984:254). Descriptives of manner should be lemmatised to their stem, as the stem carries the meaning.

Descriptives of time are derived by prefixing the time noun stem with *e-*. These descriptives should also be lemmatised to the meaning carrying stem.

3.4.6 Adverbs

The isiXhosa adverb describes the manner, place or time of a noun or another adverb (Mncube, n.d:51).

There are a few simple adverbs in isiXhosa. IsiXhosa mostly uses nouns in the locative case, e.g. ***e****-bu-suku* [at night], ***ku****-ba-ntu* [to the people], ***e****-mthi-****ini*** [to/at the tree] or by preceding nouns with the preposition *nga-* e.g*. iya* ***nga****-sekunene* [go left]. For manner, they are used by prefixing *ka-* to an adjective stem, e.g. ***ka****-de* [for long], (McLaren, 1948:142). To describe time, the prefixes *nge-, ngo- and ku-* are used to form adverbs, e.g. ***nge****-Cawa* [on Sunday], ***ngo****-Mvulo* [on Monday], ***ngo****-mso* [tomorrow], ***ku****-sasa* [in the morning], ***ku****-qala* [first].

There is also a set of simple adverbs in a closed set, e.g. *phandle* [outside] and *phesheya* [across the sea].

Simple adverbs should therefore be lemmatised to their word forms, e.g, *futhi* [again], *phakathi* [inside], but prefixal adverbs should be lemmatised to the stem to which they are bound, e.g. *ngoMvulo* [on Monday] > *Mvulo* [Monday].

3.4.7 Conjunctions

Conjunctions are words introducing a sentence or linking up two sentences (Louw et al., 1984:262).

IsiXhosa has few primitive conjunctions. Most isiXhosa conjunctions are either verbal forms (e.g. *ukuze* < *ukuze* [then], *ndaye* < *ukuya* [and I], *ngokungathi* < *ukuthi* [like], nouns (e.g. *xa* < *ilixa* [when], *mhla* < *umhla* [the day when], and pronouns (e.g. kaloku [by the way] < oku [this], okoko < oko [that]), especially the pronoun *oko* (McLaren, 1948:149; Pahl, 1982:205).

Some conjunctions are phrases, e.g. *ngoko ke* [therefore] and *ke kaloku* [when].

Even though there seems to be productivity in conjunctions, that productivity is exhausted; conjunctions are a closed set, albeit a large set. Because of this, conjunctions should be lemmatised to full word form.

3.4.8 Interjections

Interjections are isolated words that express exclamation.

There are original interjections such as ewe! [yes], *hayi bo*! [no], and *heke*! [expressing approval]. Pahl (1982:207) provides a substantial list of these that could be considered almost as a closed list.

Interjections can also be derived from other word categories, e.g. nouns, copulatives, conjunctions, adverbs and even short sentences.

For the purposes of this study, original interjections should lemmatise to full word forms as they are almost a closed list. Interjections derived from other word categories are a problem because it is difficult to identify them without looking at the context. This means that they should simply be lemmatised to the source category lemma.

3.4.9 isiXhosa NLP Lemma in Summary

It has been shown above that the best lemmas for the natural language processing of isiXhosa vary by parts-of-speech.

Most word categories should be lemmatised to the stem from which they are derived. For the few word categories that are not based on stems, the full word form should be chosen.

As evident in pronouns and words derived from other word categories, affixes used in the derivation process are concatenated morphemes. This concatenation of morphemes to form an affix is also expressed in negation.

3.5 Conclusions

In the introduction to this chapter, an explanation is given, in broad terms, of what a lemma in isiXhosa is. A classification of isiXhosa words categories is then provided as a structured way of deciding on an isiXhosa lemmas.

Appropriate forms for isiXhosa lemmas are described. More lemmas are made up of stems, and words that are not stem derived are kept as full words. There are classes of words that cannot be categorised as definitely closed but have stabilised, like original interjections. Such should be treated as a closed set and lemmatised to full word forms.

Therefore word categories that lemmatise to stems are nouns, absolute pronouns, quantitative pronouns, differentiative pronouns, superlative pronouns, possessive pronouns, all qualificatives, verbs, all copulatives except those demonstrative derived, descriptives and prefixal adverbs. Categories that lemmatise to full word forms are demonstrative pronouns,

distributive pronouns, demonstrative derived copulatives, simple adverbs, conjunctions and interjections.

It is very important to note that getting to an isiXhosa stem involves the removal of the prefix and suffix from the word, and ensuring that the terminal vowel of the stem is corrected. For full word lemmas, no removal of affixes is required.

CHAPTER 4: FEATURE SELECTION

4.1 Introduction

This section reviews the data that was used in the study. The chapter starts by detailing the source of the data and then describes the characterisation of the data. The characterisation is meant to find some heuristics of the data so that good features can be used. The conclusion specifies what the good features for the study could be together with supporting justifications.

4.2 Source of Data

The data used in the study is the Lemmatisation Corpus of the IsiXhosa NCHLT Annotated Text Corpora (South African Department of Arts and Culture & Centre for Text Technology (CTexT) North-West University South Africa 2013), a product of the NCHLT Project on Text Resources conducted by the North-West University's Centre for Text Technology (CTEXT) and the Republic of South Africa's Department of Arts and Culture (Eiselen & Puttkammer, 2014). The data was generated using rules from a study conducted by Bosch et al. (2006).

The data is available via the South African Language Resource Management Agency website[3].

4.3 Data Exploration

The corpus consists of two lemma annotated files, a 50000 word corpus of word form-lemma pairs and a 5000 word corpus of word-lemma pairs for testing purposes. The 50000 word corpus will be referred to as the general corpus, and the 5000 word corpus will be referred to as the testing corpus from this point onwards. The general corpus will be used for training and the testing corpus will be used for evaluation. The following analysis was conducted on the general corpus.

On analysis of the general corpus, it was found that it consisted of 44608 tokens. The tokens included punctuations and empty lines to separate sentences. Punctuations and empty lines had "Null" lemmas.

After the removal of empty lines and punctuations, 36526 tokens were left.

[3] Language Resource Management Agency website is http://www.rma.nwu.ac.za.

4.3.1 Method

The data analysed consisted of full word forms and corresponding lemmas. The analysis approached the problem of lemmatisation as a classification problem where the classes are transformation codes between the words and the lemma. The exercise assumed that a word form could be transformed to a corresponding lemma by identifying which affixes should be removed and which should be inserted and combined into transformation classes. An explanation of transformation classes is detailed below. The affixes identified to be removed were prefixes and suffixes, together with the corresponding insertions required if any.

For each affix type to be removed, the following data analysis was conducted.

(1) Rank the affix in ascending order of prevalence in the data.

(2) Calculate the accumulated affix coverage up to a particular rank. Given affixes ranked by prevalence in the data, Cumulative Ratio/Coverage for rank k is a fraction of how many words are covered by the k most prevalent affixes. This gives an indication of how many affixes matter in the design of the lemmatiser.

(3) Do a percentile statistical analysis.

(4) Do an estimate of the most significant number of affixes that could be used.

(5) Make remarks on the significance of the tail affixes.

The prefixes, suffixes and circumfixes were analysed (see below for the results).

The word length was also analysed to see if it would matter in the lemmatisation strategy.

4.3.1.1 Transformation Classes

A transformation class is a sequence of characters that specifies what should be replaced at the beginning and end of the word to reduce it to its lemma. A good, albeit rare, example is the reduction of the word, *ekuqinisekiseni [when strengthening]*, to its lemma, *qina [harden]*. The transformation class is *Leku>Risekiseni>a*. The L denotes the transformation to be done at the beginning of the word and R indicates the transformation to be done at the end of the word. What follows the "L" or "R" is a sequence of characters that need to be removed from the word. The ">" then denotes the beginning of the characters with which the sequence of characters will be replaced. In the example, *Leku>Risekiseni>*a means that the sequence of letters *eku* at the beginning of the word must be replaced with an empty

string and the sequence of letters *isekiseni* at the end of the word must be replaced with the letter "*a*". Below are a few illustrations:

Table 31: Examples of Transformation Classes

Word	Lemma	Transformation class
esetyenziswayo	sebenza	Lesety>sebRiswayo>a
ixesha	xesha	Li>
elithatyathwayo	thabatha	Lelithaty>thabRwayo>a
azisiwe	azisa	Riwe>a

4.3.2 Prefixes

4.3.2.1 Overall Prefix Coverage

The data consisted of 3020 unique prefixes. The table below (Table 32) shows the top 10 prefixes, the number of tokens with the prefix, and the cumulative ratio of the coverage of the prefixes.

Table 32: Top 10 Prefixes, their counts and cumulative coverage

	Prefix	PrefCount	CumRatio
1	ku	1635	0.044763
2	i	1545	0.087061
3	e	822	0.109566
4	uku	778	0.130866
5	u	754	0.151509
6	um	484	0.164759
7	ii	430	0.176532
8	ezi	420	0.188030
9	a	366	0.198051
10	o	328	0.207031

Table 33 shows the preffix percentiles of the cumulative coverage.

Table 33: Percentiles of overall prefix coverage

count	3020.000000
mean	0.772772
std	0.095234
min	0.044763
2.5%	0.491203
5%	0.588934
10%	0.672116
25%	0.756968
50%	0.806590
95%	0.843784
97.5%	0.845850
max	0.847917

The maximum coverage of the prefixes is 84.8%. As one can see, half of the prefixes cover 80.6% of the data, accounting for 95% of the prefix covered data. Figure 4, below, shows the cumulative coverage of the prefixes ordered by prevalence.

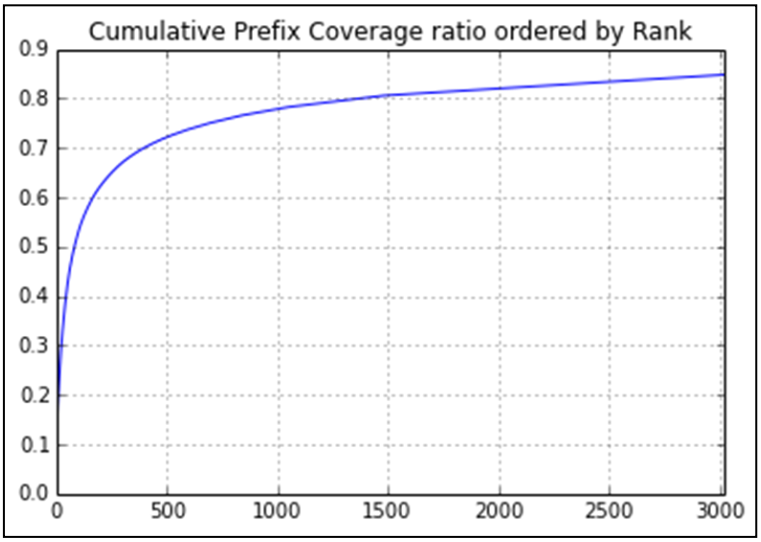

Figure 4: Prefix coverage

As one can see from the graph, the graph follows a straight line from the 50% percentile.

4.3.2.2 Prefix Only Coverage

From the data characterised by having only prefixes, 2419 unique prefixes were identified. The table below (Table 34) shows the top 10 prefixes, the number of tokens with the prefix, and the cumulative ratio of the coverage of the prefixes.

Table 34: Top 10 Prefixes, their counts and cumulative coverage

	Prefix	PrefCount	CumRatio
0	i	1252	0.034277
1	ku	1135	0.065351
2	u	548	0.080354
3	uku	519	0.094563
4	um	469	0.107403
5	ii	420	0.118902
6	e	336	0.128101
7	ama	317	0.136779
8	isi	272	0.144226
9	ye	263	0.151426

Table 35 below shows the prefix percentiles of the cumulative coverage.

Table 35: Percentiles of Prefix Only Coverage

count	2419.000000
mean	0.580038
std	0.077598
min	0.034277
2.5%	0.339399
5%	0.427271
10%	0.499485
25%	0.568718
50%	0.607348
95%	0.637138
97.5%	0.638793
max	0.64044

The maximum coverage of the prefixes in prefix only data is 64%. As one can see, half of the prefixes cover 60.7% of the data, accounting for 95% of the prefix covered data. Figure 5, below, shows the cumulative coverage of the prefixes for data containing only prefixes.

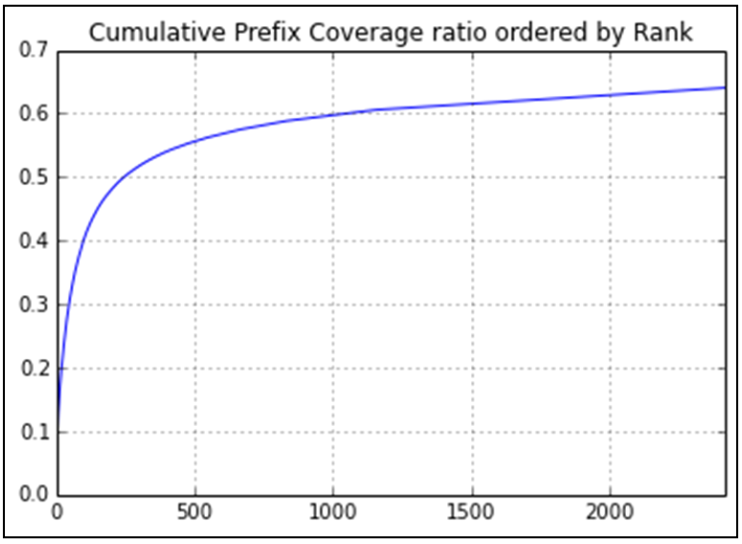

Figure 5: Prefix coverage in Prefix Only Data

As one can see from the graph, the graph follows a straight line from the 50% percentile.

4.3.3 Suffixes

4.3.3.1 Overall Suffixes Coverage

The data consisted of 311 suffixes. For the entire data containing suffixes, the table below (Table 33) shows the top 10 suffixes, the number of tokens with the suffix and the cumulative ratio of the coverage of the suffixes:

Table 36: Top 10 Suffix, their counts and cumulative coverage

	Suffix	SufCount	CumRatio
1	e	1047	0.028665
2	wa	646	0.046351
3	yo	435	0.058260
4	eka	394	0.069047
5	we	345	0.078492
6	ileyo	334	0.087636
7	i	291	0.095603
8	isa	290	0.103543
9	eni	217	0.109484
10	ela	199	0.114932

The table below (Table 37) shows the suffix percentiles of the cumulative coverage.

Table 37: Percentiles of suffix coverage

count	311.000000
mean	0.195242
std	0.026620
min	0.028665
2.5%	0.107998
5%	0.141967
10%	0.173329
25%	0.196504
50%	0.204950
95%	0.209399
97.5%	0.209611
max	0.209823

The figure below (Figure 6) shows the cumulative coverage of the suffixes. The maximum data coverage of the suffixes is 21%. As one can see, half of the suffixes cover 20.5% of the covered data, which account for 97.6% of the suffix covered data.

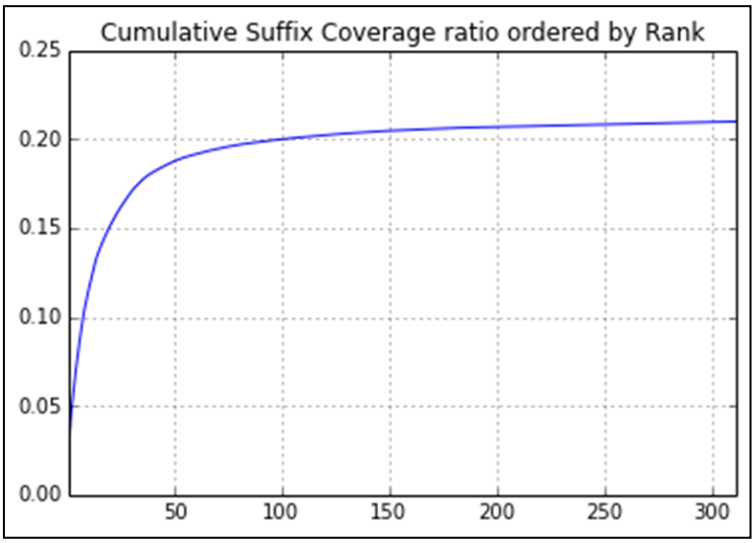

Figure 6: Suffix cumulative coverage

As one can see from the graph, the graph also follows a straight line from the 50% percentile.

4.3.3.2 Suffix only coverage

From the data that only had suffixes, 29 suffixes were identified. For the data suffix only data, the table below (Table 38) shows the top 10 suffixes, the number of tokens with the suffix and the cumulative ratio of the coverage of the suffixes.

Table 38: Top 10 Suffix, their counts and cumulative coverage

	Suffix	SufCount	CumRatio
0	e	25	0.000684
1	be	10	0.000958
2	isa	6	0.001122
3	ele	4	0.001232
4	eleyo	3	0.001314
5	eleni	3	0.001396
6	na	3	0.001478
7	ise	3	0.001561
8	ye	3	0.001643
9	eni	2	0.001697

The table below (Table 39) shows the suffix percentiles of the cumulative coverage.

Table 39: Percentiles of suffix coverage

count	29.000000
mean	0.001823
std	0.000444
min	0.000684
2.5%	0.000876
5%	0.001024
10%	0.001210
25%	0.001561
50%	0.001971
95%	0.002316
97.5%	0.002335
max	0.002354

The figure below (Figure 7) shows the accumulate coverage of the suffixes. The maximum data coverage of the suffixes is 0.24%.

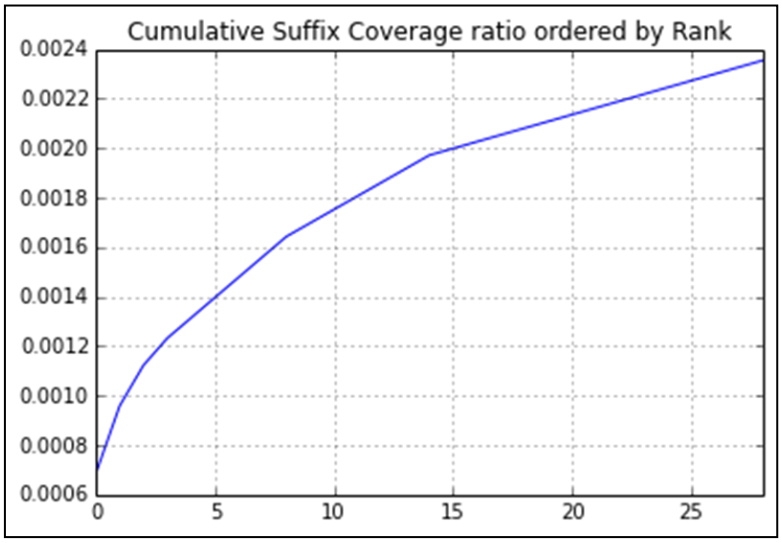

Figure 7: Suffix cumulative coverage for suffix only data

As one can see, half of the suffixes cover 0.19% of the covered data, which accounts for 79.6% of the suffix only covered. The graph then follows a straight line from there.

4.3.4 Circumfix Coverage

From the data, 2504 circumfixes were identified. The table below (Table 40) shows the top 10 circumfixes, the number of tokens with the suffix and the cumulative ratio of the coverage of the suffixes.

Table 40: Top 10 Circumfixes, their counts and cumulative coverage

	Prefix	Suffix	CircCount	CumRatio
15	ku	eka	205	0.005612
44	si	e	110	0.008624
51	u	e	101	0.011389
52	eku	eni	94	0.013963
53	a	e	94	0.016536
70	aba	i	76	0.018617
73	i	e	71	0.020561
80	e	ileyo	66	0.022368
85	ku	wa	63	0.024092
87	ngoku	yo	62	0.025790

The table below (Table 41) shows the suffix percentiles of the cumulative coverage.

Table 41: Percentiles of circumfix coverage

count	2504.000000
mean	0.163584
std	0.036508
min	0.005612
2.5%	0.066999
5%	0.087290
10%	0.111605
25%	0.146895
50%	0.173205
95%	0.204042
97.5%	0.205755
max	0.207469

As one can see, half of the suffixes cover 17% of the covered data, which account for 82% of the suffix only covered.

The figure below (Figure 8) shows the accumulate coverage of the circumfixes. The maximum data coverage of the circumfixes is 20.7%. As one can see from the graph, the graph also follows a straight line from the 50% percentile.

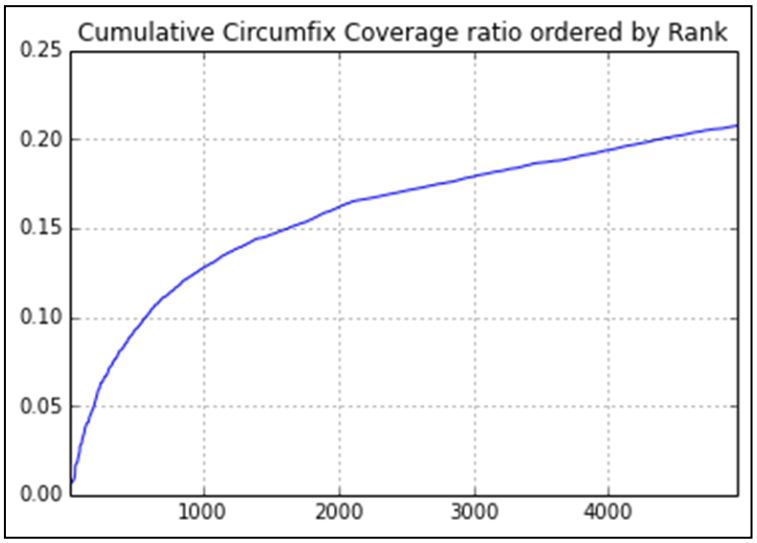

Figure 8: Circumfix cumulative coverage for Suffix Only data

4.3.5 Classes

The data consisted of 5131 distinct transformation classes. The table below (Table 42) shows the top 10 classes, the number of tokens with the class and the cumulative ratio of the coverage of the classes.

Table 42: Top 10 Classes, their counts and cumulative coverage

	Class	ClassCount	CumRatio
0	0	5435	0.148798
1	Li>	1249	0.182993
2	Lku>	1135	0.214067
3	Lu>	548	0.229070
4	Luku>	481	0.242238
5	Lum>	469	0.255079
6	Lii>	420	0.266577
7	Lama>	317	0.275256
8	Le>	314	0.283853
9	Lisi>	272	0.291299

The table below (Table 43) shows the class percentiles of the cumulative coverage

Table 43: Percentiles of class coverage

count	5131.000000
mean	0.896413
std	0.107256
min	0.148798
2.5%	0.587766
5%	0.683718
10%	0.768275
25%	0.866438
50%	0.929776
65%	0.950843
95%	0.992978
97.5%	0.996489
max	1.000000

The maximum data coverage of the classes is 100.%. As one can see, half of the classes account for only 93% of the suffix covered data. To achieve coverage of 95% of the data covered by the classes, one would use 65% of the classes, which is 3335 classes.

Figure 9 below shows the cumulative coverage of the classes. As one can see from the graph, the graph also follows a straight line from the 50% percentile.

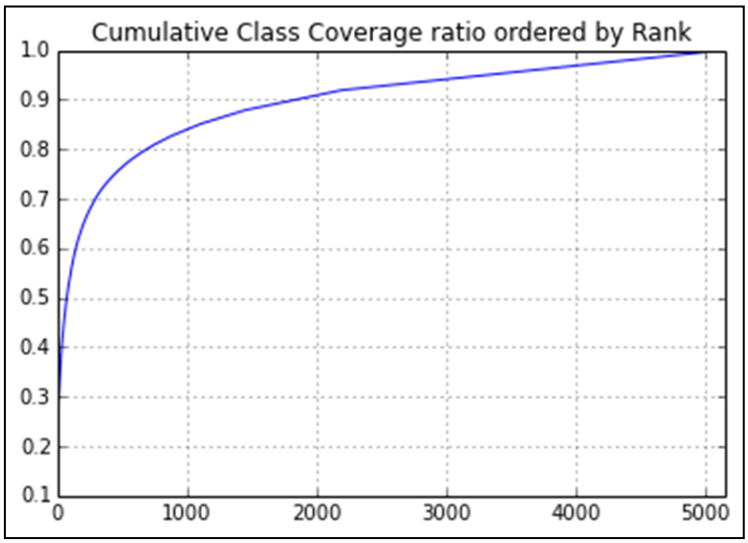

Figure 9: Classes cumulative coverage

4.3.6 Word Length

Another feature of the words that is readily available is the word length. An exploration of the word lengths in relation to the lemma was done.

A Pearson correlation of the length of the full word to the combined length of the affixes was performed.

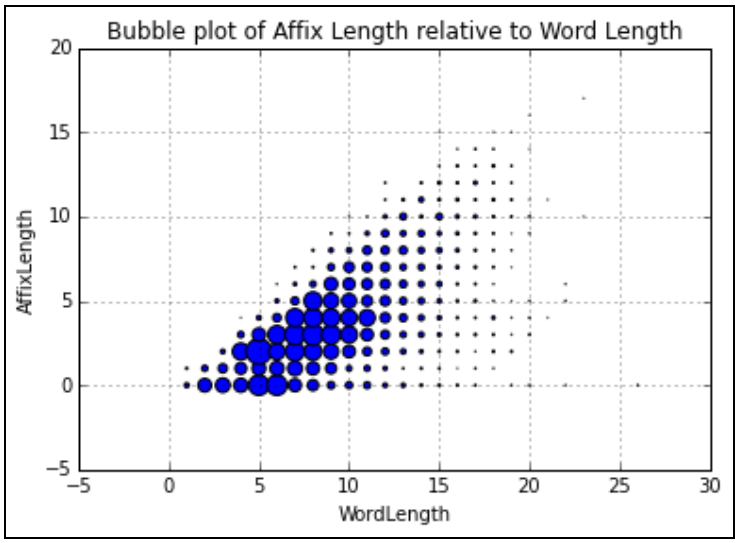

Figure 10: Bubble Plot of Affix Length relative to Word Lengths

As one can see from the graph in Figure 10, there is a relationship between the affix length and word length. The measured correlation was 68.1% correlation between the length of the affixes and the word length. This is a strong correlation, and implies that the length of a word can also be used to decide on the right transformation class for a word

4.4 Conclusions

The data confirms that isiXhosa is a prefixal language with 3020 prefixes active in 84.8% of the data. Of this, 64% of the data was characterised by prefixes only. There were 311 suffixes overall, active in 21% of the data. Only 0.24% of the data was characterised by suffixes only. Some of the prefixes and suffixes combine into 2504 circumfixes that cover 20.7% of the data. However, there are 5131 classes overall, covering all of the data. Of that, 14.9% of data lemmatised to the full word forms.

Table 44: Affix counts, and their maximum data coverage

Affix	Count	Maximum Coverage
Prefixes	3020	84.8%
Suffixes	311	21%
Circumfixes	2504	20.7%
All Affix Classes	5131	85.1 %

Because prefixes cover so much of the data, searching for the lemmatisation class could be drastically improved by using the prefix as the primary search index, followed by the suffix.

The correlation measure of 68.1% of the affix length to word length shows that the length of a word can be used in deciding on the right class for a word. This, therefore, shows that the {prefix, suffix, word length} set of features will give the best class discrimination when finding the right class for a word.

CHAPTER 5: ISIXHOSA GRAPHICAL LEMMATISER

5.1 Introduction

An overview of the isiXhosa Graphical Lemmatiser (XGL) is provided, followed by an explanation of its different components and how they function. The XGL was designed and implemented as part of this study. Details on the model are covered in the conclusion and a summary of the chapter is provided.

The isiXhosa Graphical Lemmatiser (XGL) is a machine learning (Mitchell, 1997:2) lemmatiser. It is inspired by the Ripple Down Rules (Compton et al. 1991) algorithm. The literature review identified the Ripple Down Rules (RDR) algorithms, a form of decision tree learning, to be the algorithm producing the best results for morphologically complex languages like isiXhosa. The RDR algorithm is explained in section 2.3.2: *Data Driven Lemmatisation Studies*.

5.2 XGL Model

At the centre of the XGL is the lemmatisation model that is used in the lemmatisation process. This section explains the model used by the XGL in lemmatisation. This model is generated by the XGLTrainClassTree component based on the output of XGLTrainClassesSplit.

The model consists of three parts: a lexicon of word-lemma pairs that the XGL encountered during training, the hierarchy of transformation classes, and a class confidence threshold level. This section starts with the lexicon.

5.2.1 XGL Model's Lexicon

Chapter 3 has shown that a number of isiXhosa word classes are a closed set and lemmatise to full word-form. The easiest way to handle these words is by refereeing to a look-up list, a lexicon. In addition the lexicon of word-lemma pairs ensures that words that have been encountered in training are lemmatised to a very high accuracy. All the lemmas for each word encountered during training are kept, with a count of how many times a particular lemma was encountered. A sample of a lexicon is listed below:

```
aliphelise': {   'phelisa': 1},
aliqingqelwa': {   'qingqa': 1},
aliyi': {   'ya': 1},
alo': {   'lo': 2},
```

Depending on how the training data was generated, it can sometimes not be 100% accurate. There is a possibility that more than one lemma can be allocated to the same word in the training data. To store that information, a prevalence count for each lemma is kept as shown below:.

```
aliphelise': {   'phelisa': 1, 'phela':7},
```

In the above example, the lemma with the highest prevalence would be used in deciding on the lemma.

5.2.2 Model's Hierarchy of Transformation Classes

The hierarchy of transformation classes is used by the lemmatiser to generate lemmas for words that the XGL has not encountered before i.e. Out of Vocabulary (OoV) words. XGL does this by finding the most appropriate transformation class for a word. As transformation classes were explained in section *4.3.1.1: Transformation Class*, this section explains how they are generated, how they are structured into a hierarchy, and how they are used.

5.2.2.1 Generating Transformation Classes

To generate a transformation class between a word and a lemma, one first finds the longest common string (LCS) between the word and its lemma. The differences between the LCS and the word are the affixes to be removed. The differences between the LCS and the lemma are the affixes to be inserted to create the lemma. In lemmatising *ekuqinisekiseni* to *qina*, the LCS is *qin.* The affixes to be removed are *eku* at the beginning of the word and *isekiseni* at the end of the word. These should be replaced with nothing at the beginning and "*a*" at the end, hence the class *Leku>Risekiseni>a*.

The LCS algorithm for XGL was based on a Python implementation that is freely available at Wikipedia[4]. However, this implementation was biased towards suffixing languages. This became apparent when the algorithm gave "*a*" as the longest common string for between *asiyi* and *ya,* when it actually is *y* for isiXhosa. This algorithm was modified to be biased towards prefixes instead of suffixes.

An example of a conversion class is shown below:

Ekuqinisekiseni => qina. : *Leku>Risekiseni>a*

[4] https://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Longest_common_substring#Python2

5.2.2.2 Structure and Information Related to Transformation Classes

To be able to decide whether a transformation class is the right one for lemmatising a word, some word features need to be used. The section below details the choice of features and the information to be captured with a class. This is followed by how the classes are indexed.

5.2.2.2.1 Choosing Features

Based on the work done in the data analysis part of the study, the features that characterise a word-lemma pair were chosen to be the prefix, suffix and word-length of the word. This information is used to find the most appropriate transformation class for the word.

The training data provides a number of possible prefixes, suffixes and circumfixes that could be identified for a word. For the purposes of the study, the prefix and suffix were generalised to a circumfix. Using circumfixes showed immense promise in improving the spellchecker for isiZulu, a language very similar to isiXhosa (Prinsloo & de Schryver 2004). A prefix was represented as a circumfix with an empty suffix, and a suffix was represented as a circumfix with an empty prefix. For each circumfix, therefore, the XGL captures the transformation classes, and for each class, statistics of the word-length encountered for that class are stored. It is possible for one circumfix to be present in more than one transformation class. Because of this, the model can store more than one class per circumfix. The statistics stored are the number of encounters, the mean word-length, and the standard deviation of the word-length for the words encountered for that class. These statistics are used to discriminate between classes. An example of a leaf of the model is shown below:

```
('be', 'isa'):

     {'CLASSES':

          {'Lbe>Risa>a': {'Count': 1, 'Stats': {'Mean':
          12.0,'Std': 0.0}},

          'Lbe>Risa>o': {'Count': 1, 'Stats': {   'Mean':
          11.0, 'Std': 0.0}

     }

},
```

The statistics are used to model the probability of the word belonging to the class, p*(class|word).* The simple Gaussian distribution on word length is used as an estimate of the probability of the word belonging to the class.

5.2.2.2.2 Class Tree Hierarchy

IsiXhosa affixes are made of concatenated morphemes (Pahl, 1982:2). This is also shown in chapter 3, specifically in negation and derivation of words from other word categories. The regularity of this concatenation allows for affixes to be structured into a hierarchy. For prefixes, the concatenation is left to right and for suffixes the concatenation is right to left (Meinhof, 1932). An example of a hierarchy of circumfixes is shown in the example below:

```
('o','')
     ('oku', 'na')
          ('oku','ana')
               ('oku', 'isana')
     ('oku', 'nyelwana')
```

The hierarchy is determined as follows: a circumfix C1 is a child of another circumfix C2 if the prefix of C1 starts with the prefix of C2 and the suffix of C1 ends with the suffix of C2, and the circumfix C1 is not the same as C2.

Arranging possible affixes in a hierarchy allows us to restrict the search for the appropriate class to a subset of the class tree. The process of finding the most appropriate transformation class will be discussed in the section dealing with how XGL uses transformation classes to generate lemmas for unknown words.

5.2.3 XGL Class Confidence Threshold

For the XGL to use a class in transforming a word, it must be confident enough that it is using the most appropriate class for the word, so the threshold is the minimum confidence that XGL will accept for a class to be used. In addition, not all isiXhosa words are transformed, e.g. *ngoba* [because]. *Ngoba* is a conjunction and conjunctions are lemmatised to full word forms. The XGL therefore needs to decide at what point it should transform a word. This is where the confidence threshold comes in. If the confidence that a word belongs to a class is high enough, that class is used to transform the word, otherwise the word is not transformed. This is to ensure that the XGL does not use a wrong lemmatisation strategy to lemmatise a word.

To choose a particular lemma or transformation class, the XGL calculates the probability that a word belongs to a transformation class. As mentioned earlier the class model keeps statistics on the length of words encountered during training for each class. The simple Gaussian distribution

on word length is used as an estimate of the probability of the word belonging to the class *p(class|word).*

The class confidence threshold is the minimum acceptable probability that a word belongs to a transformation class.

5.3 How does the XGL work?

This section of the chapter delves into the inner workings of the XGL. An overview of the XGL is provided followed by an explanation of its different components and how they function.

5.3.1 Overview of XGL

XGL generates a lemmatisation model from word-lemma pairs. This model is then used to lemmatise other words.

The workflow in XGL is shown below:

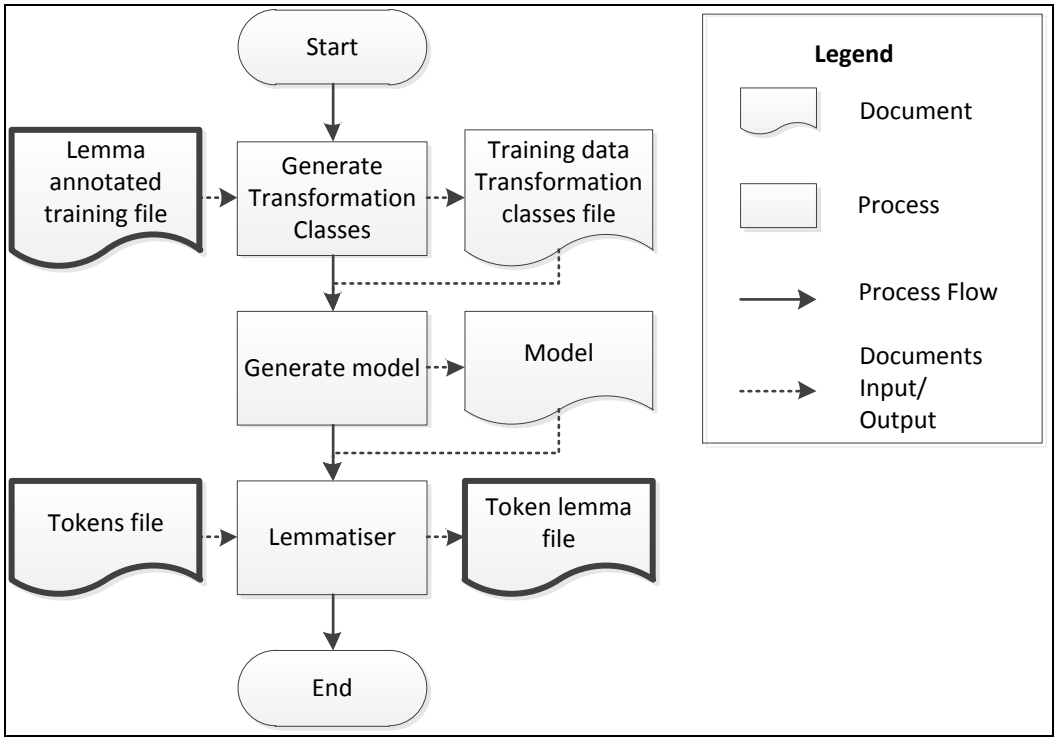

Figure 11: XGL Workflow

Each process step in the workflow corresponds to an XGL component.

The XGL is made up of three components that work in tandem. The first two components are used in training the XGL, thereby generating the lemmatisation model, and the third component

is used for lemmatisation using the generated model. The components are: XGLTrainClassesSplit, XGLTrainClassTree and XGLLemmatise.

The XGLTrainClassesSplit component is used to generate lemmatisation transformation information from the word-lemma pairs in the training data. The details of this lemmatisation transformation information are described in section *5.2.2.2.1: Choosing Features*. This lemmatisation transformation information should be saved to a file for input into the next component of the XGL.

The XGLTrainClassTree component generates the lemmatisation model from the lemmatisation transformation information file. This lemmatisation model should also be stored in a file to be used as input in the lemmatisation of other isiXhosa words.

The XGLLemmatise component uses the lemmatisation model generated earlier to lemmatise words.

The next section will go into how each component of the XGL works.

5.3.2 How does the XGLTrainClassesSplit work?

The purpose of the XGLTrainClassesSplit is to generate the transformation classes for each word-lemma pair in the training file. This component ignores empty lines, and lines with a blank lemma.

For each word-lemma pair, XGLTrainClassesSplit returns the word, the lemma, and the affixes used in the transformation class and the transformation class separates them using commas.

5.3.3 How does XGLTrainClassTree work?

The XGLTrainClassTree component of the XGL is the heart of this lemmatiser and generates the model, as stated earlier.

This component captures encountered word-lemma pairs into the lexicon with the prevalence counts. It then calculates the statistics for each class and structures the class tree. The lexicon, the transformation class tree structure, and the threshold of 0.975 are then compiled into the model and outputted into a file or standard output.

The threshold was decided during validation tuning. Threshold tuning was done by executing the lemmatiser at different thresholds in the range 0.0 to 1.0. The results are shown in Table 45 below

Table 45: XGL Performance vs Threshold

	Total Acc	OoV Acc	Known Acc	FMeasure	Precision	Recall
0.00000	0.633211	0.070175	0.992255	0.650100	0.884516	0.633211
0.00001	0.638991	0.085020	0.992255	0.657961	0.884088	0.638991
0.00010	0.641093	0.090418	0.992255	0.661123	0.882927	0.641093
0.00100	0.644246	0.098516	0.992255	0.665615	0.884062	0.644246
0.01000	0.653705	0.122807	0.992255	0.678835	0.884959	0.653705
0.10000	0.672622	0.171390	0.992255	0.702836	0.888880	0.672622
0.20000	0.684708	0.202429	0.992255	0.717159	0.889794	0.684708
0.30000	0.698371	0.237517	0.992255	0.733469	0.892429	0.698371
0.40000	0.706779	0.259109	0.992255	0.743298	0.892853	0.706779
0.50000	0.717814	0.287449	0.992255	0.755407	0.896092	0.717814
0.75000	0.759327	0.394062	0.992255	0.797968	0.904397	0.759327
0.85000	0.780347	0.448043	0.992255	0.816394	0.904182	0.780347
0.90000	0.801892	0.503374	0.992255	0.834146	0.903163	0.801892
0.95000	0.828692	0.572200	0.992255	0.857219	0.909574	0.828692
0.97500	0.842880	0.608637	0.992255	0.869208	0.914709	0.842880
0.98000	0.856017	0.642375	0.992255	0.880274	0.919597	0.856017
0.98500	0.859695	0.651822	0.992255	0.884190	0.922682	0.859695
0.99000	0.869154	0.677463	0.991394	0.891001	0.924630	0.869154
0.99900	0.894377	0.742240	0.991394	0.910963	0.933195	0.894377
0.99990	0.897530	0.750337	0.991394	0.913887	0.935049	0.897530
1.00000	0.959012	0.908232	0.991394	0.944452	0.933622	0.959012

The graph below shows the trend of the Performance vs. Threshold in graphical form.

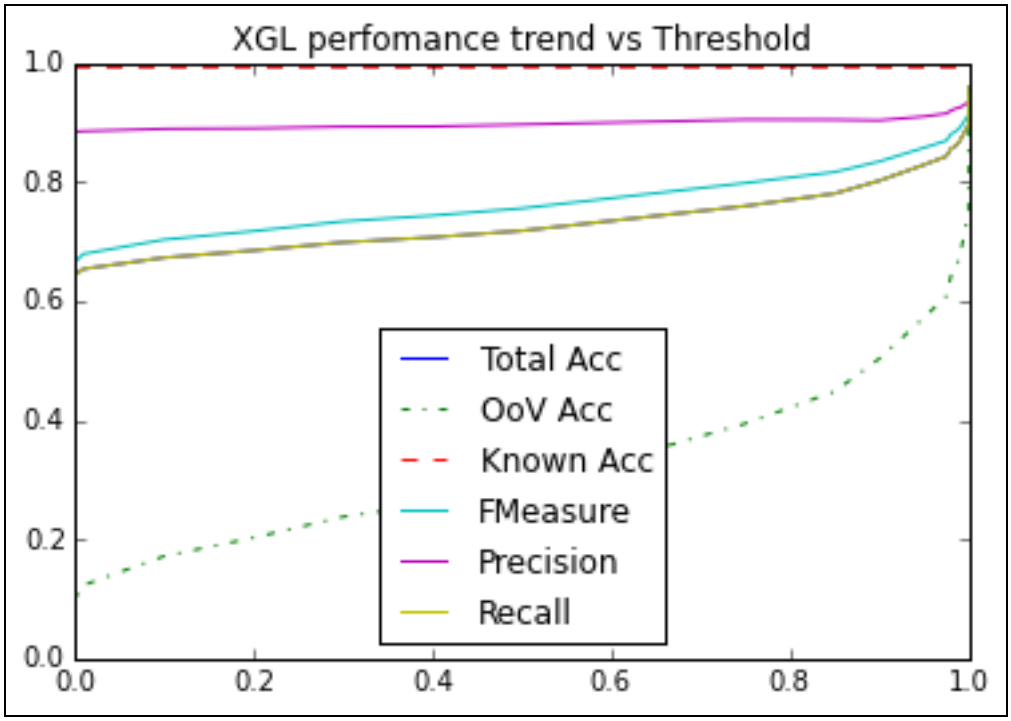

Figure 12: XGL validation Performance vs Threshold

From the graph above one can see that the performance of XGL follows a trend but jumps at the threshold of 1.0. This jump is even more visible in the accuracy of the XGL on Out of Vocabulary (OoV) words between the thresholds of 0.9999 and 1,0, The accuracy on OoV words at a threshold of 0.9999 is 75.03%. However the lemmatiser jumps to 90.82% at 1.0. We

could not explain this anomaly, and to be conservative, we chose a threshold of 0.975 to avoid overtraining. Overtraining in machine learning is when an artificial intelligence system fits the training data very well, but is unable to generalise to unknown data. A threshold of 0.975 implies that the lemmatiser is 97.5% confident that the word belongs to the class being evaluated.

5.3.4 How does the XGLLemmatise work?

For each word in the tokens file, the process shown in Figure 13 is followed.

First, the lexicon is searched. If the word is found in the lexicon, the most prevalent lemma is returned. If the word is not found in the lexicon, the most appropriate class is then used to transform the word to a lemma.

As mentioned in section *5.2.2.2.2: Class Tree Hierarchy* , the class tree is structured according to the circumfixes identified in the training data. From the root of the class tree, matching circumfixes are found together with matching sub circumfixes and compiled into a candidates list. A circumfix matches a word if:

(1) The word starts with the prefix of the circumfix, if the circumfix has a prefix,

(2) The word ends with the suffix of the circumfix, if the circumfix has a suffix, and

(3) The combined length of the circumfix components is less than the word length.

If there are no candidate classes, the word is returned un-lemmatised, otherwise the probability that the word belongs to the candidate classes is calculated. A simple Gaussian distribution on word length was chosen to model the probability that a word belongs to a transformation class. The statistics that were captured by XGLTrainClassesTree for each class, with the word-length, are used in the probability calculation.

The transformation class with the highest probability is chosen as the transformation class to use in reducing the word to its lemma.

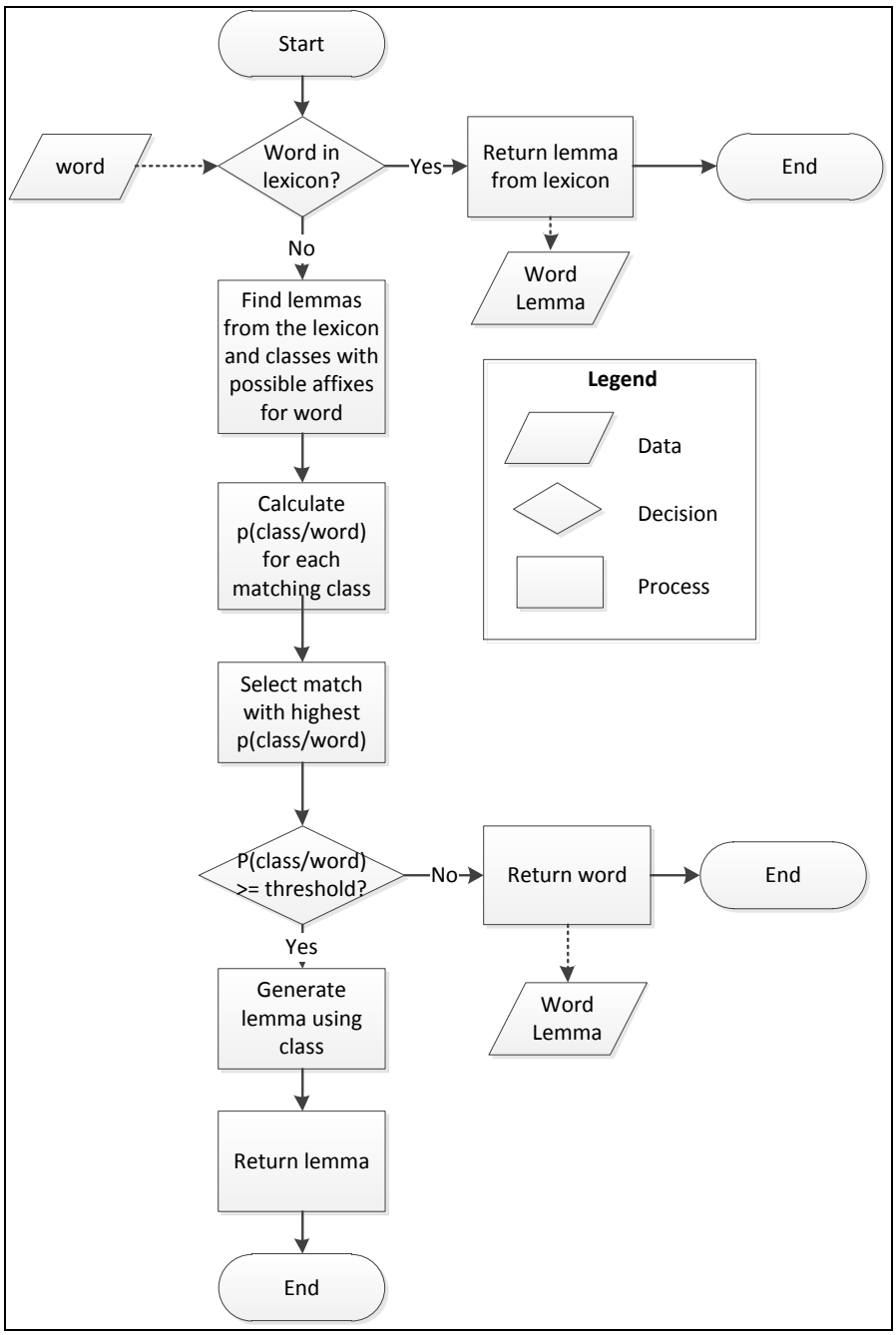

Figure 13: Word Lemmatisation Workflow

If the probability of the preferred class is less than the threshold, the word is not transformed using the class but returned as it is.

5.4 Using the XGL

This section details how the XGL should be used. For each of the components, the section details how the component is used, what the input should be, and what the output is.

The XGL was written in Python (van Rossum 2007) version 2.7. Therefore, the components are Python files i.e. XGLTrainClassesSplit.py, XGLTrainClassTree.py, and XGLLemmatise.py.

The data structures of XGL were not optimised for performance but were structured for accuracy.

5.4.1 How to use XGLTrainClassesSplit

XGLTrainClassesSplit is executed as follows:

```
>python XGLTrainClassesSplit.py lemma_Annotated_file
[output_file]
```

The lemma annotated file is the training file with word-lemma pairs. The requirement is for each line to contain a single pair of a word and a lemma separated by a tab character. The lemma annotated files is a tab delimited file. Lines containing empty lemmas or blank lines are ignored by the XGL. The lemma transformations generated are saved into the output file if specified, otherwise this information is written to the standard output, e.g. the computer screen. If the output file specified already exists, it is overwritten, otherwise it is created.

5.4.2 How to use XGLTrainClassTree

XGLTrainClassTree is used for creating the lemmatisation model. The XGLTrainClassTree is executed as follows:

```
>python XGLTrainClassTree.py lemma_transformation_training_file
[output_file]
```

The lemma transformations training file is the output from the previous state, the XGLTrainClassesSplit component. The model generated by this component is written into the output file, if one is specified. If the output file is not specified, the component uses the standard output to display the model. If the output file specified already exists, it is overwritten, otherwise it is created.

5.4.3 How to use the XGLLemmatise

The XGLLemmatise is used to lemmatise words to lemmas based on the already trained model. The XGLLemmatise is executed as follows:

```
>python XGLLemmatise.py class-tree-file tokens-file [output-
file] [verbose]
```

The class-tree-file, the model generated by XGLTrainClassTree, is input to this component. The tokens file is a file containing the words to be lemmatised. The tokens' file must contain a single word per line. The lemmas of the words in the token file are written to the output file, if specified, otherwise to the standard output. If the output file already exists, it is overwritten, otherwise it is

created. The output is the word followed by the lemma in the same line. The output is tab delimited. If the *verbose* parameter is specified and is "True", the lemmatiser writes as output all the possible word lemmas in order of preference, in the same line. This is mainly for interrogating the lemmatiser's performance.

5.5 Conclusions

This chapter explains the design of the XGL and how the XGL works. It starts by giving details of the model used by the XGL, including the motivation behind structuring the class model into a circumfix tree. The chapter delves a bit into the feature selection, the motivation behind the selected features, and the selection of the right word to lemma transformation.

An overview of the XGL is given together with how each component works and the motivation for the design.

The chapter ends by detailing how to use the three components of the XGL, i.e. the XGLTrainClassesSplit, XGLTrainClassTree, and XGLLemmatise.

CHAPTER 6: EVALUATION

6.1 Introduction

This section details how the experiment was conducted and presents results for the experimental work.

6.2 Experimental Design

The section starts by giving a summary of the data used in the experiments. It then considers how the data was separated and sampled to minimise bias in the setup of the lemmatisers used in the experiment. This is followed by an explanation of how the control lemmatisers were chosen. The section finishes by giving a description of the experiment and a summary of the section.

As data and data sampling are at the centre of the experiments, this section starts by discussing the data.

6.2.1 Data Source

The data used in this study was sourced from the South African Language Resource Management Agency website[5], and was described in the chapter on data analysis. It is the lemmatisation corpus of the IsiXhosa NCHLT Annotated Text Corpora. The development of this corpus is described by Eiselen and Puttkammer (2014).

The corpus consists of two lemma-annotated files, a 50000 word corpus of word form-lemma pairs and a 5000 word corpus of word-lemma pairs for testing purposes. From this point onwards, the 50000 word corpus will be referred to as the general corpus, and the 5000 word corpus will be referred to as the testing corpus. According to Eiselen and Puttkammer (2014), the general corpus was developed using a finite state lemmatiser (Bosch et al. 2006). The testing corpus was initially generated in the same way, but was then quality assured by linguistic experts for use as a gold standard. The accuracy of the general corpus in relation to the testing corpus was measured at 79.82% by Eiselen and Puttkammer (2014).

Of importance, is the sequence of tokens within the corpora. The sequence of tokens was kept the same as in the original source text i.e. keeping the sequence of sentences and words in a

[5] Language Resource Management Agency website is http://www.rma.nwu.ac.za.

sentence. This is important because different parts of speech are distributed throughout the text, as they would be in real use. The corpora are made up of sentences. Each word in the sentence occupies a single line in the file. Each punctuation mark also occupies its own line and an empty line separates the sentences. Each word or punctuation has its lemma written in the same line and separated by a comma. Because punctuations and empty lines lemmatise to nothing, this study excludes them. This is reflected in the number of tokens for the datasets discussed below.

6.2.2 Data Setup

For development and experimental purposes, the data was divided into three sets, i.e. development data, validation data, and evaluation data. Each of these three had training data and testing data. Training data is used to train the lemmatiser, and testing data is used to measure the performance of the lemmatisers.

In dividing the data, one wanted to ensure that none of the work involved created any bias to the ultimate experimental results. The objective was to evaluate the lemmatisers objectively.

To ensure that the character of the data was not lost, only contiguous chunks of particular sizes were extracted during the experimental work.

6.2.2.1 Development Data

Development data is data used during the development of a tool. This data is used while one is setting up the lemmatisers or checking that the lemmatisers work as intended. The quality of this data is not critical, as it is not used in the evaluation. However, it must have the same structure as the evaluation data.

Two development sets of different sizes were extracted from the data: a 2000 line set and a 1000 line set. This was mainly to check that the system performed differently for different training set sizes; a characteristic of machine learning tools (Mitchell 1997: 2).

For the 2000 line set, the first 2000 lines from the general corpus were extracted as training data; the second contiguous 2000 lines of data were extracted from the general corpus as testing data.

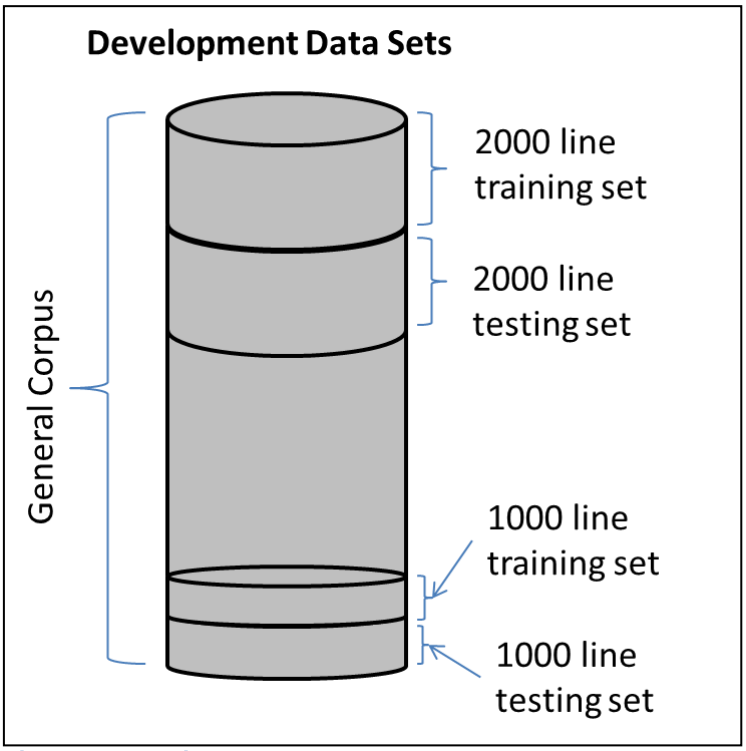

Figure 14: Development Data Sets

For the 1000 line set, the second last 1000 lines from the general corpus were extracted as training data and the last contiguous 1000 lines of data were extracted from the general corpus as testing data.

After cleaning up empty lines and punctuation lines, the sizes of the sets were as follows:

Table 46: Development Data Set Sizes

Data Set	Size (lines)
2000 Token development training set	1600
2000 Token development testing set	1597
1000 Token development training set	859
1000 Token development testing set	839

6.2.2.2 Validation and Evaluation Data

Whatever system has been implemented, it needs to be tuned for the best performance; this is where validation comes in. Validation data is used to tune the system without biasing the ultimate results of the experiment. Evaluation is the ultimate run of the experiment.

6.2.2.2.1 Training Set and 10-Fold Sampling

Because of the small size of the data, it was decided that the same data would be used for the training purposes in validation and evaluation, but that different test sets should be used for the two. Thus, the general corpus was used for training the lemmatisers for both validation and evaluation.

On analysis of the general corpus, it was found that it consisted of 44608 tokens. The tokens included punctuations and empty lines to separate sentences. Punctuations and empty lines had "Null" lemmas.

After removing the punctuation lines and blank lines, the training set was reduced to 36535 lines.

6.2.2.2.1.1 10-Fold Sampling

Strict k-fold sampling splits the data set into k blocks. Of the k blocks, k-1 blocks are used for training and the extra block is used for testing. This is done k times with each block becoming a testing block once. Having k results from the same data allows for statistical evaluation of the results. The training set sizes are therefore (k-1)/k of the data size, and the test set sizes a k^{th} of the data size. Strict k-fold sampling is ideal for where there is only one set of data to use for both training and evaluation of the performance of the tool.

Where a separate testing data set exists, a different approach needs to be taken. A form of "leave-one-out-sampling" is performed on the training data and the testing set is used in its entirety.

A 10-fold sampler was developed for this experiment. The 10-fold sampler was developed in Python (Van Rossum, 2007) and works as follows: it divides the training set into 10 contiguous blocks. Using the contiguous blocks, it maintains the structure of the sentences inherent in the data, keeping the data as it would be seen in real life. The sampler also caters for sampling different training set sizes.

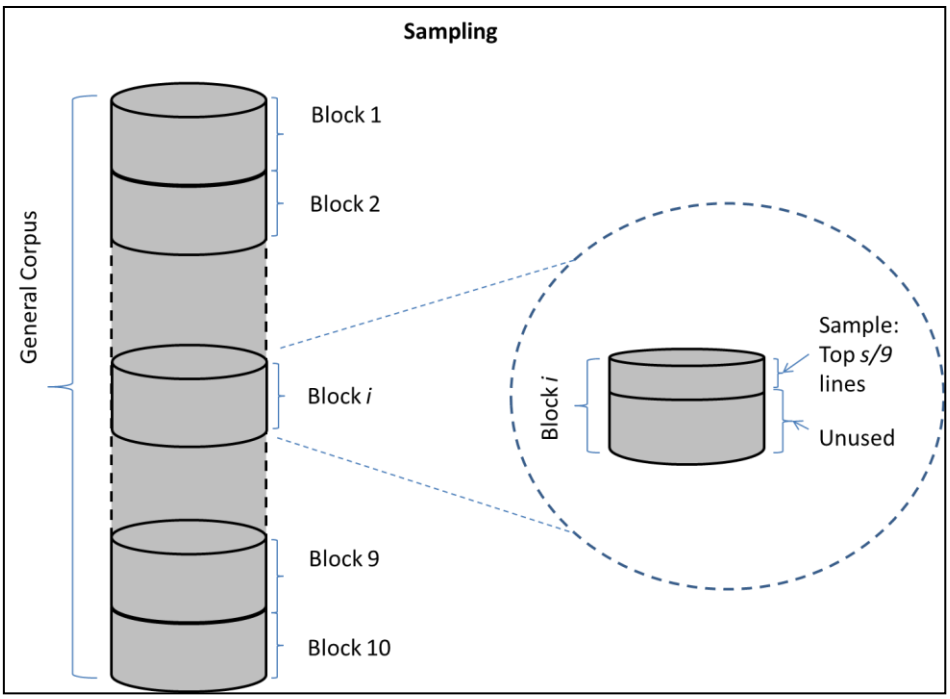

Figure 15: Sampling for 10 Fold Validation

To sample a size ***s*** training set**,** the sampler extracts the first ***s/9*** lines from each of the 10 blocks. The rest of the lines are not used. Nine of those extracts are saved to a training file and one is saved to a testing file. This is done 10 times with each block being used for testing only once.

6.2.2.2.2 Testing Sets

For testing the lemmatisers, the testing corpus was used. To ensure that the validation experiments did not affect the evaluation work, the testing corpus was split into two, a 2368 line validation testing set and a 2368 line evaluation testing set was used.

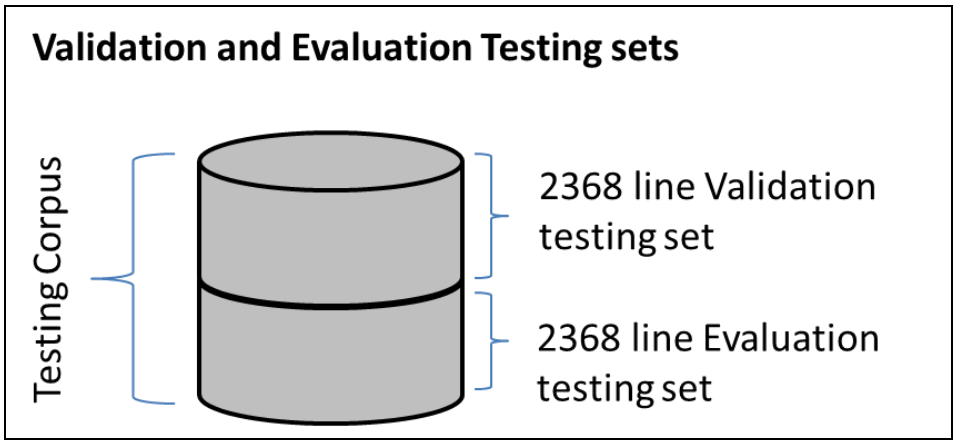

Figure 16: Validation and Evaluation Testing Test

To ensure that the word and sentence sequence structure was maintained, the first contiguous half of the testing corpus was made the validation testing set, and the contiguous bottom half of the testing corpus became the evaluation testing set. Figure 16 shows the splitting of the testing corpus into test sets.

The test sets also contained empty lines and blank lemmas; these were removed, resulting in the test set sizes shows in Table 47.

Table 47: Validation and Evaluation Testing Set Sizes

Data Set	Size (lines)
Validation testing	1903
Evaluation Testing Set	1890

6.2.3 Choice of Lemmatiser

The study considered the implementations of machine learning lemmatisers for isiXhosa. Two publicly available lemmatisers were trained, evaluated, and compared to one built specifically for this study i.e. the isiXhosa Graphical Lemmatiser (XGL).

The freely available lemmatisers chosen were the CST lemmatiser (Jongejan & Haltrup 2013) and the LemmaGen lemmatiser (Juršič et al. 2010). These lemmatisers were chosen because they implement the Ripple Down Rules (RDR) algorithm which seemed to be suited to highly synthetic languages. The CST lemmatiser was also chosen because it was used in many studies (Saraswathi & Geetha, 2007; Agic et al., 2013; Ingason et al., 2008) and was therefore a good benchmark. The LemmaGen lemmatiser showed superior performance against other Ripple Down Rules lemmatisers, including the CST lemmatiser (Juršič et al., 2010), and was also a good benchmark. However, the LemmaGen had seen limited use.

6.2.3.1 CST Lemmatiser

The CST[6] lemmatiser implementation was written in Borland C++ 5 and was compiled using Microsoft Visual C++. This lemmatiser uses a hierarchy of rules. Each rule is represented by the form: `affix0*affix1*…*affixK->insert0*insert1*…*insert.` The hierarchy is similar to the Ripple-Down Rules in that for a child rule, the parent rule should hold true for

[6] The CST lemmatiser is open sourced and freely available at http://cst.dk/download/cstlemma/current.

candidate classes. Conflicts in lemmatisation were not handled, and the first lemma generated is accepted as the output. The implementation was compared to the suffix rules of 12 European languages.

6.2.3.2 LemmaGen Lemmatiser

LemmaGen[7] is an enhanced Ripple-Down Rules (Plisson et al., 2004) lemmatiser tested on 12 languages. The lemmatisation of a new word is done in the same way as the most similar word form in the lexicon. The system also used the suffix feature, as it was meant for European languages as well. Ambiguation was done by choosing the most prevalent/frequent class. Where there was equal prevalence, the second most similar class was used. The original RDRs form a tree structure and are ordered, implying that the first rule to fire is accepted. To improve the efficiency and readability of rules, the LemmaGen implementation extends the RDR structure by imposing a similarity condition for an exception list, meaning that all the suffixes share the same k-1 characters where k can be chosen.

The LemmaGen lemmatiser is written in C++ and compiled in Microsoft Visual C++.

6.2.4 Overview of the Experiment

Four stages were followed in setting up the experiment i.e., development, experimental setup, validation and evaluation.

6.2.4.1 Development

The development stage ensures that the lemmatiser runs and generates results. This technical stage confirms that the lemmatiser can be trained and tested. At this stage, the technical constraints arise and the experiment needs to be adjusted accordingly.

The major technical constraint encountered at this stage was that the CST lemmatiser did not handle null/blank lemmas. As null lemmas are for punctuations, it necessitated the removal of all punctuations from the data; hence the reduction in the data size, as will be seen later in the chapter.

It was also noted that the commas separating the lemma from the full-word in a line of a training caused problems, so the data set were converted to tab delimited format.

[7] LemmaGen is open source software available at http://lemmatise.ijs.si/Software.

In the development stage, the lemmatisers were installed, configured and tested to see if they worked as intended. The systems were trained and tested on development data sets and results showing a difference between the two development sets were obtained, thus revealing that the lemmatisers work.

Having confirmed that the lemmatisers work, a proper experimental setup was constructed.

6.2.4.2 Experimental Setup

This section describes the experiment setup. The experiments were run on the IPython Notebook (Pérez & Granger, 2007) system.

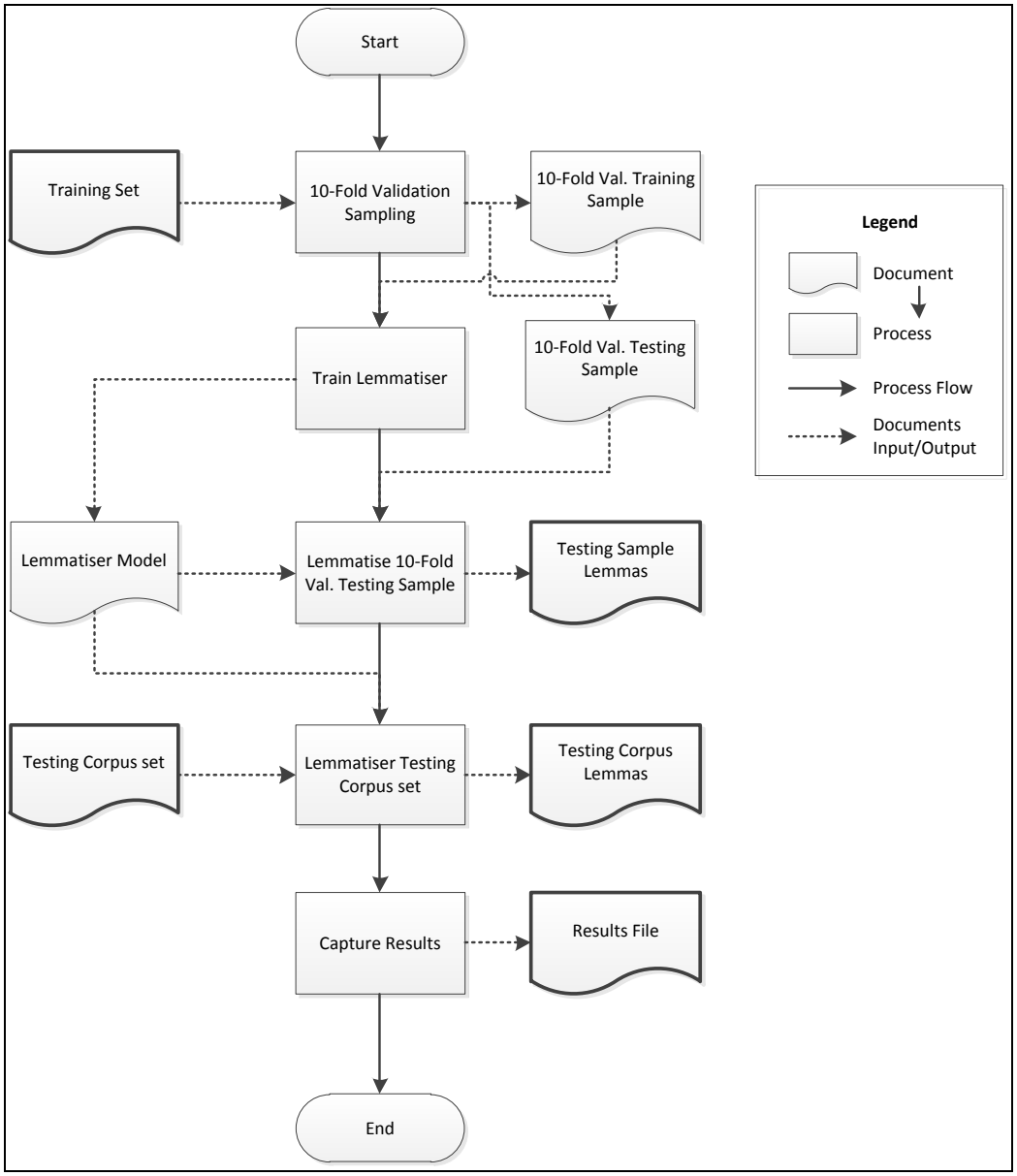

Figure 17: Experiment Workflow

The experiment starts by doing a 10-fold sampling from the training set file. The sampler generates the training file and the testing file. The 10-fold sampler was written in Python (Van Rossum, 2007).

The lemmatiser is then trained using the training sample file. The output of the training processes is stored in the model files.

The model file is used by the lemmatiser to lemmatise the words in the testing files and the results are captured. For strict 10-fold validation, the 10-fold validation testing sample is lemmatised. Because there is also a testing corpus, the testing corpus is also lemmatised.

The results are captured to a results file as follows:

- Accuracy, which is the proportion of words lemmatised correctly
- The F1-Score
- Number of training words
- Memory storage used during training process
- Memory storage used during lemmatisation,
- Training execution time and
- Lemmatisation duration.

6.2.4.3 Validation and Evaluation

The validation experiment is a setup of the above experiment where one tunes the systems involved for better results. The experiment was set up as above, but the validation testing set was lemmatised. The results of the validation experiment are not reported in this study but the evaluation of the experiment is reported.

6.2.5 Conclusions on Experimental Design

The chapter shows that the data was sourced via the South African Language Resource Management Agency (RMA). The data used is the IsiXhosa NCHLT Annotated Text Corpora (South African Department of Arts and Culture & Centre for Text Technology (CTexT) North-West University South Africa 2013) described by Eiselen and Puttkammer (2014).

An explanation is given on the choice of the lemmatisers and how the data was split-up for the experimental work. This split ensured that there was no bias incorporated into the results of the experiment.

Details of the experiment highlighting the 10-Fold validation sampling method are given. The experiment was setup to ensure the validity of the experiment and reliability of the results.

6.3 Results

6.3.1 Introduction

This section details the results of the study. The section starts by doing an accuracy comparison of the three systems, followed by an analysis of the computer resources performance and the significance of the results.

The results are presented for two forms of evaluation, i.e. general corpus evaluation and test corpus evaluation. The general corpus tests are strictly tenfold sampling tests where the training and testing sets come from the general corpus. The second evaluation is on using the testing corpus where the general corpus is used for training, but the testing is done using the testing corpus. In all the tests, measures taken on the reliance of the results are presented as p-values of the Wilcoxon test (Wilcoxon, 1945).

A summary of the section is provided. The data analysis software, Pandas (McKinney, 2010), was used for the statistical analysis.

6.3.2 Linguistic Performance

In this study, linguistic performance is measured on accuracy and the F1 Score. The accuracy measure gives an overall score while the F1 score measures the technical performance of the lemmatisers.

Accuracy is measured as a ratio of the number of words that were lemmatised correctly to the total number of words submitted for lemmatisation.

The F1-Score is the weighted average of precision and recall, and is defined in section *2.2.5.3: F-Measure*.

6.3.2.1 Accuracy

This section of the document presents the results obtained in measuring the accuracy of the lemmatisers. This section starts by stating the overall accuracy results, continues to present the

lemmatisation accuracy on Out of Vocabulary (OoV) words, and ends by presenting the accuracy on known words.

6.3.2.1.1 Overall Accuracy

For accuracy measurements measured on the general set, the CST lemmatiser outperformed the XGL and LemmaGen lemmatiser, as shown in Figure 18. The CST lemmatiser achieved an average overall accuracy rate of 71.8% compared to XGL's 66.88% and LemmaGen's 63.89%.

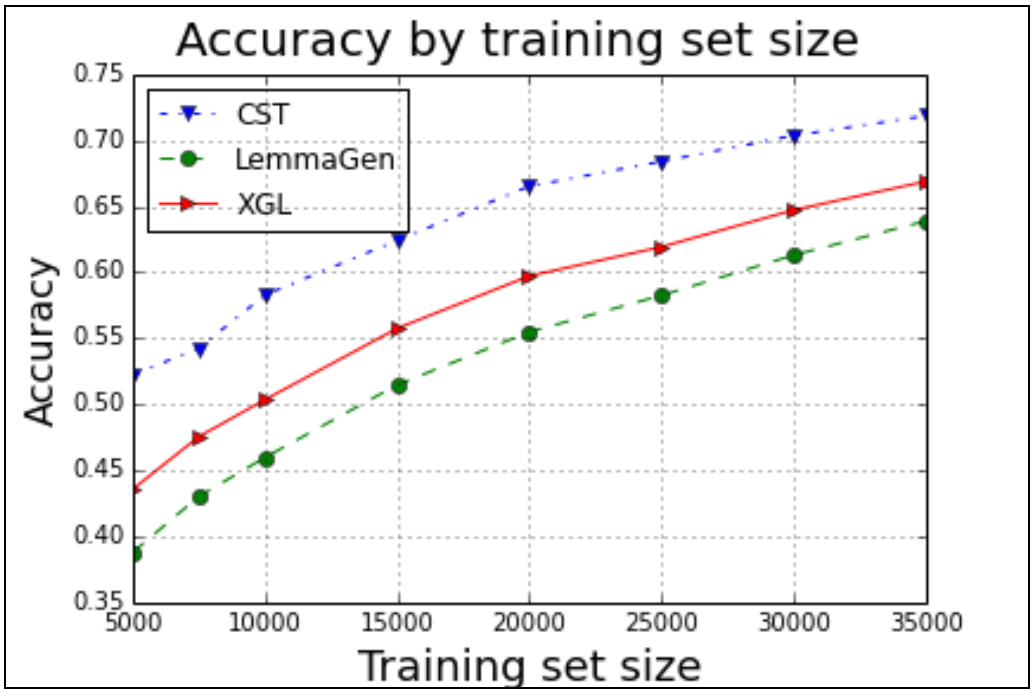

Figure 18: Lemmatisation Accuracy on General Corpus by Training Set Size

This shows that the CST lemmatiser created a better model from the training data than the other two lemmatisers. The Wilcoxon (1945) confidence tests were done between the XGL and the CST lemmatiser, and the XGL and LemmaGen, to test the confidence levels on the results. The maximum p-value was 0.007, which is below the threshold of 0.05. Therefore confidence can be placed on the results.

The resulting p-values mean that there is a maximum 0.7% chance that the results were caused by statistical errors or noise.

When the lemmatisers were evaluated against the testing corpus, a different picture emerged. All the lemmatisers performed better, but in this setting, the XGL lemmatiser performed far better than the other two, as shown in Figure 19.

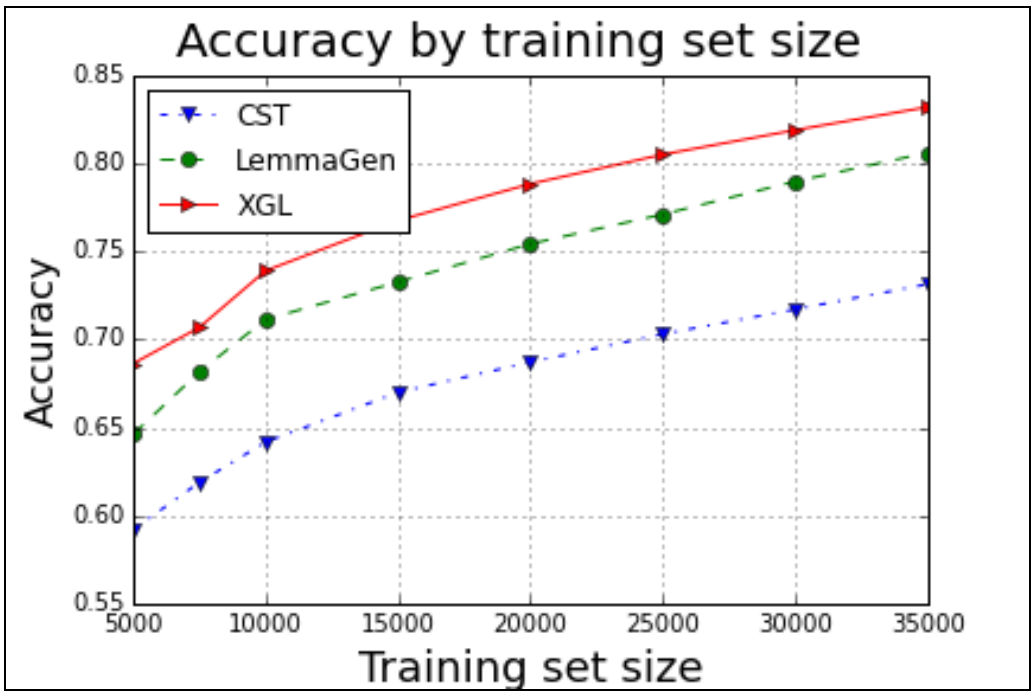

Figure 19: Lemmatisation Accuracy on Testing Corpus by Training Set Size

The XGL's peak accuracy was measured at 83.19%, followed by the LemmaGen lemmatiser at 80.6%, and the CST lemmatiser's accuracy measured at 73.14%.

To get a measure of confidence on the above results, a Pair-wise Wilcoxon test was also done. The resultant p-values were all below the threshold of 5%, showing that the accuracy results are statistically significant.

The increase in the overall accuracy rates when the lemmatisers are tested against the testing corpus is curious. It could be because the testing corpus is a gold standard and consequently has less noise than the general corpus. The accuracy results also suggest that the CST Lemmatiser generates a lemmatisation model that reflects the training data better than the other lemmatisers. This is evident in the CST lemmatiser's superior performance when measured on the general set compared to the tested set. On the other hand, the XGL lemmatiser generalises to the isiXhosa language better than the other two. This is evident in its superior performance when tested against the isiXhosa gold standard, the testing corpus. This makes sense if one considers that the XGL lemmatiser was designed with isiXhosa in mind, whilst the other lemmatisers were not designed specifically for isiXhosa.

6.3.2.1.2 Accuracy on Known Words

One of the performance measures is the accuracy of the lemmatisers on words that it encountered during the training process, i.e. known words. Figure 20 shows the accuracy of the lemmatisers on known words:

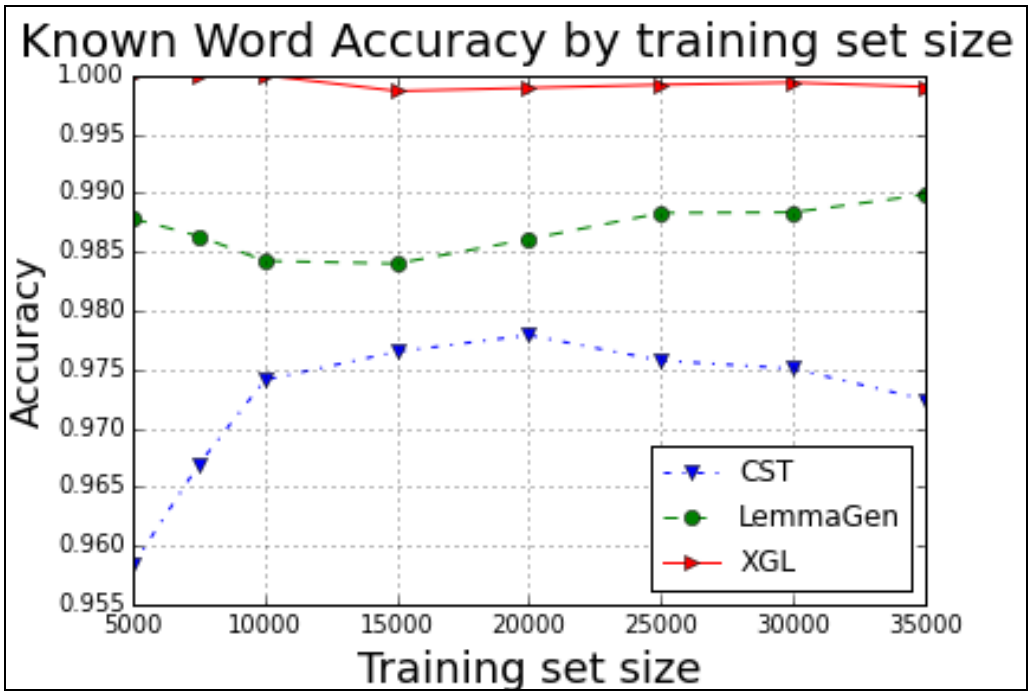

Figure 20: Average Accuracy for Known Words Tested on General Corpus by Training Set Size

As can be seen in Figure 20, the XGL lemmatiser provides very high accuracy on known words when trained and evaluated on the general set i.e. close to 100%. The CST lemmatiser provided the lowest accuracy of 97.24% at a training set size of 35000 pairs. Table 48 shows the pair-wise Wilcoxon p-values for these results at different training set sizes:

Table 48: Pair-Wise Wilcoxon p-values for Known Word Lemmatisation on General Corpus by Training set size

	XGL-LemmaGen	LemmaGen-CST	XGL-CST
5000	0.011719	0.012515	0.005062
7500	0.011719	0.015156	0.005062
10000	0.005062	0.020879	0.005062
15000	0.005062	0.092601	0.005062
20000	0.005062	0.092601	0.005062
25000	0.005062	0.016605	0.005062
30000	0.005062	0.012515	0.005062
35000	0.005062	0.005062	0.005062

The Wilcoxon p-values showed some statistical overlap in the 15000 and 20000 training sample sizes between LemmaGen and the CST Lemmatiser at 0.0926. These were higher than 5%, and are highlighted in Table 48. For the rest of the comparisons the p-values were all below 0.05, confirming the results to be dependable. This means that for the training set sizes of 15000 and 20000 word lemma pairs, the difference that shows in the graph cannot be relied on statistically.

The picture is not that different when the lemmatisers are evaluated against the testing set, except for an increase in the accuracy of all three lemmatisers.

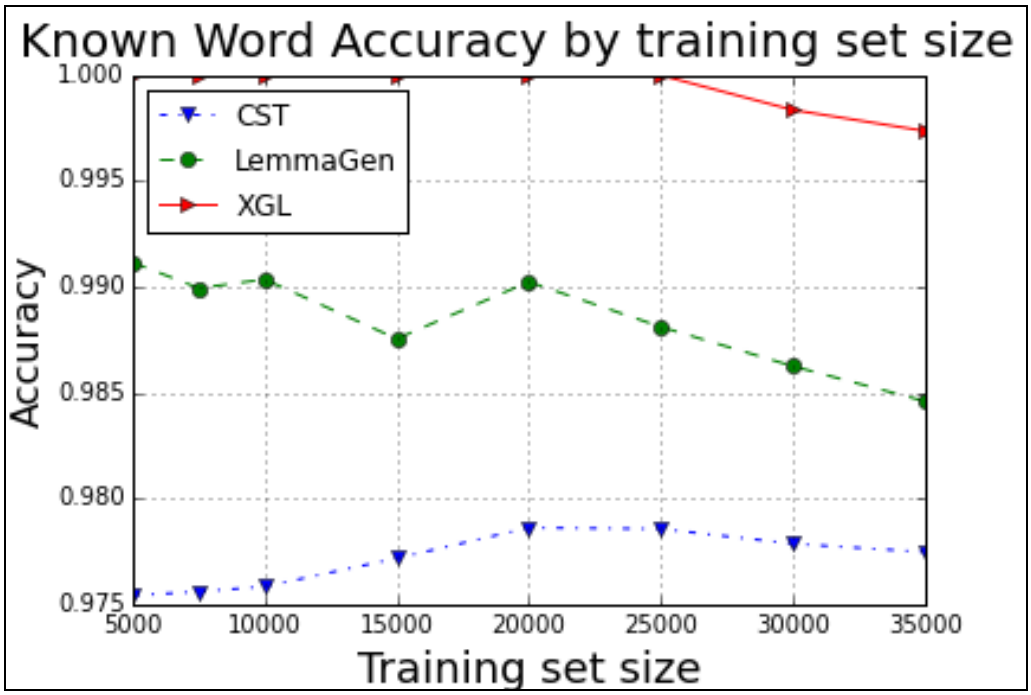

Figure 21: Average Accuracy on Known Words Evaluated on Testing Corpus by Training Set Size

As can be seen in Figure 21, again the XGL lemmatiser provides for very high accuracy on known words when evaluated on the testing corpus again – slightly less than 100%, with CST providing the lowest accuracy again, but at of 97.75% at the maximum training set size of 35000 pairs. The pair-wise Wilcoxon p-values for these results were all 0.005.

All the result comparisons gave statistically significant p-values that are an order of magnitude below 5%, and may therefore be relied upon.

The high results stated in this section should not come as a surprise because all the lemmatisers are lexicalised.

6.3.2.1.3 Accuracy on Out of Vocabulary Words (OoV)

An even more important measure of accuracy is the performance of the lemmatiser on Out of Vocabulary (OoV) words. OoV words are words that the lemmatiser did not encounter during training. This measure of the performance of a lemmatiser shows how well the lemmatiser generalises from the training data.

Figure 22 shows the average accuracy rates on OoV word when the lemmatisers are tested against the general corpus:

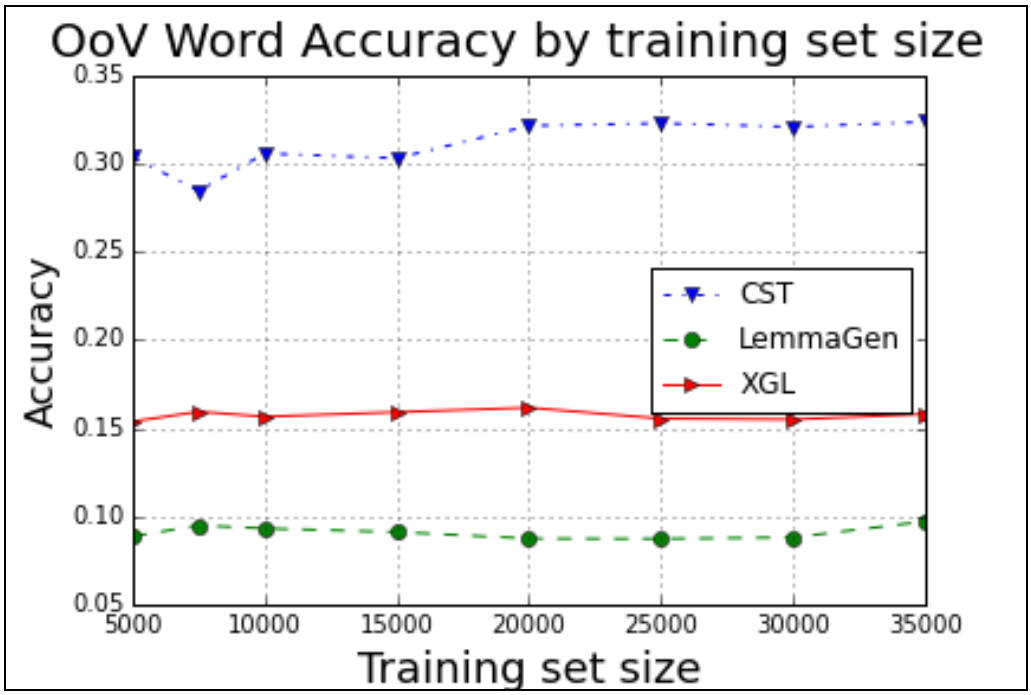

Figure 22: Average Accuracy on OoV Words Validated on General Corpus by Training Set Size

As can be seen, the CST lemmatiser provided for better accuracy on OoV words when trained and evaluated on the general corpus, peaking at 32.36% for 35000 word-lemma pairs. XGL was the next best lemmatiser, with an accuracy hovering around 16% throughout the range. LemmaGen provided the lowest accuracy of 9.7% at the maximum training set size range of 35000 word-lemma pairs.

The pair-wise Wilcoxon p-values for the OoV accuracy rate were all 0.005, implying statistical significance of the measures between all the lemmatisers, as they were all below the threshold of 5%. This means that the chance that the results are caused by statistical noise is around 0.5%.

The picture again changes when the lemmatisers are evaluated against the testing corpus for OoV words. The increase in accuracy is curious for all the lemmatisers. It is also important to note that the accuracy for all three lemmatisers also increases with the size of the training set.

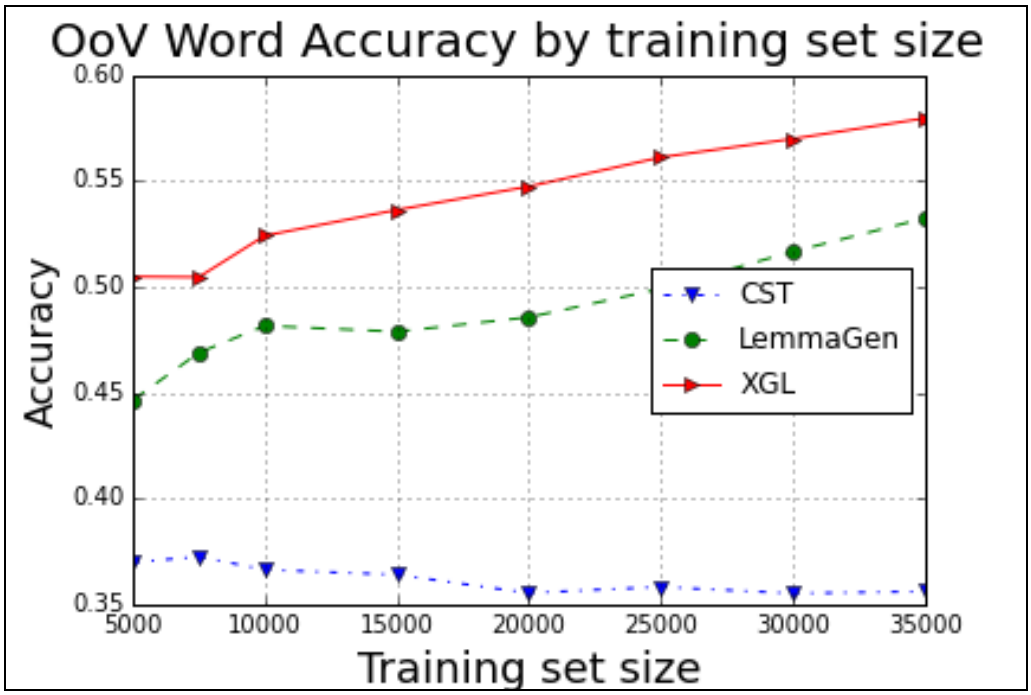

Figure 23: Average Accuracy on OoV Words Evaluated on Testing Corpus by Training Set Size

As can be seen in Figure 23, the XGL lemmatiser outperformed the other two lemmatisers on the accuracy for OoV words when evaluated on the testing corpus, peaking at 57.97% for 35000 words-lemma pairs, with LemmaGen following at 53.25% and the CST lemmatiser providing the lowest peak accuracy of 35.62%. Of interest, was that the CST lemmatiser's average performance slightly degraded as the training set size increased when evaluated against the testing set.

For this evaluation, the resultant p-values were also an order of magnitude below the threshold of 0.05, meaning that one could rely on the results.

The results presented in this section show that the CST lemmatiser generalises from the training data better than the other lemmatisers, as is evident from its OoV accuracy rate on the general corpus being higher than the other lemmatisers. However, when the lemmatisers are tested against a gold standard, the XGL lemmatiser shows better accuracy than the other lemmatisers. This suggests that the XGL lemmatiser generalises to the isiXhosa language better than the CST lemmatiser. Again, this is expected, as the lemmatiser was designed with isiXhosa in mind, whilst the other lemmatisers were not. Of interest though, is that the CST lemmatiser performs very well when tested against the general corpus, which is also the training set, but slightly degrades when tested against the separate testing corpus. This again shows that the CST lemmatiser models the training data very well, but not necessarily the language isiXhosa. This is of course expected, as the CST lemmatiser was not designed for isiXhosa but instead it is a general lemmatiser.

6.3.2.2 F1-Score results

The F1-score gives an indication of how well a classifier performed internally with respect to balancing precision and recall. The F1-Score gives a balance average between the two.

When tested against the general corpus, CST Lemmatiser outperformed the XGL and LemmaGen with a maximum F1-Score of 0.7 at 35000 word-pairs. The XGL lemmatiser followed with the maximum F1-Score of 0.64. The lowest peak F1-Score was attributed to the LemmaGen lemmatiser's 0.62.

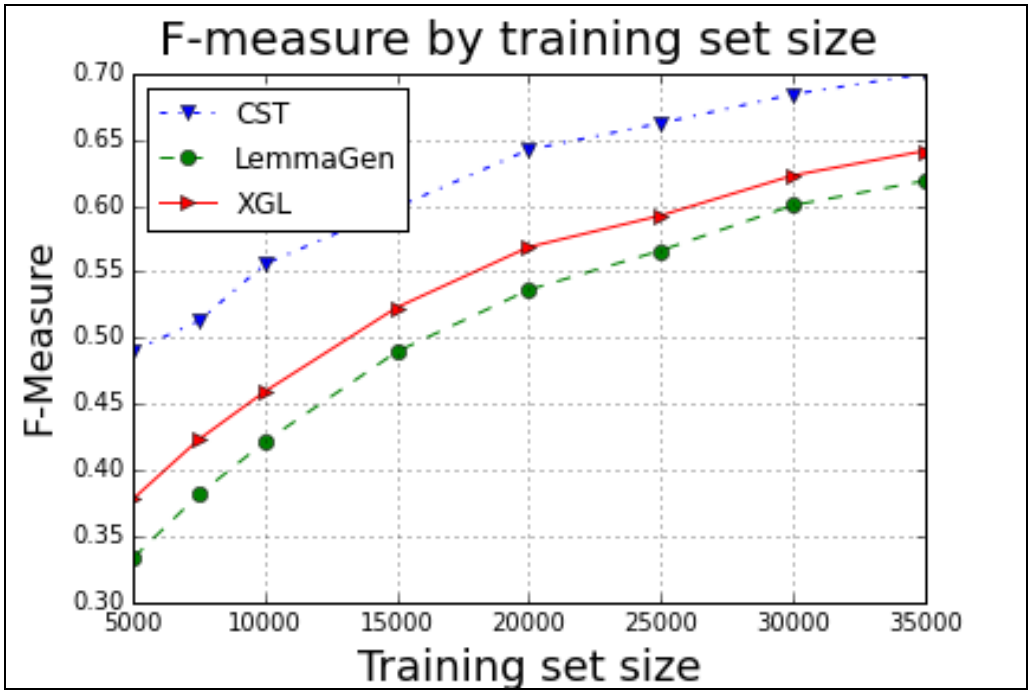

Figure 24: F1-Score for Evaluation on General Corpus by Training Set Size

The Wilcoxon test done on the F1-Score showed that the results could be relied upon, as the p-values were all 0.005, which is an order of magnitude below the 0.05 threshold.

Evaluating the lemmatiser on the testing corpus test gave the set a different set of results. As expected from the average accuracy rates, the F1-Score increased for all the lemmatisers. The XGL lemmatiser had the best F1-scores, at 0.86, followed by LemmaGen at 0.85. The CST lemmatiser had the lowest maximum average F1-score at 0.77.

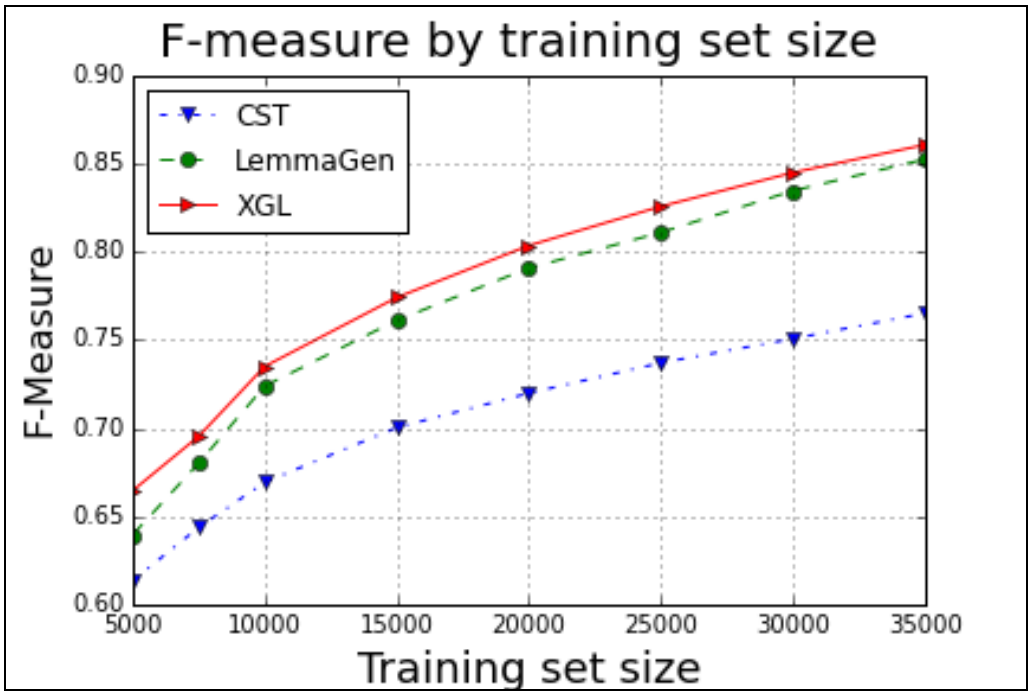

Figure 25: F1-Score Evaluated on Testing Corpus by Training Set Size

Testing for the statistical significance of the results using the Wilcoxon test showed p-values of 0.005, proving that the results are dependable.

These results indicate that the CST lemmatiser does a better model for the training data, but that the XGL creates a better model for the language.

6.3.3 Computing Resources Performance

In this section, we investigate the computing performance of the lemmatisers, specifically memory usage and execution times.

It is important to note that the CST Lemmatiser and LemmaGen were written in C++ and compiled to machine code. The XGL, however, was written in Python 2.7 and ran as an interpreted script. This difference is expected to show in the performance of the XGL regarding the use of computing resources. The section starts by looking at execution times results.

6.3.3.1 Execution Duration

There are two stages to consider when looking at execution: training execution time and lemmatisation execution time. This section starts with the training execution duration.

6.3.3.1.1 Training Execution Duration

The graph below (Figure 26) shows the average training times per word-lemma pair of the three lemmatisers in relation to training set size:

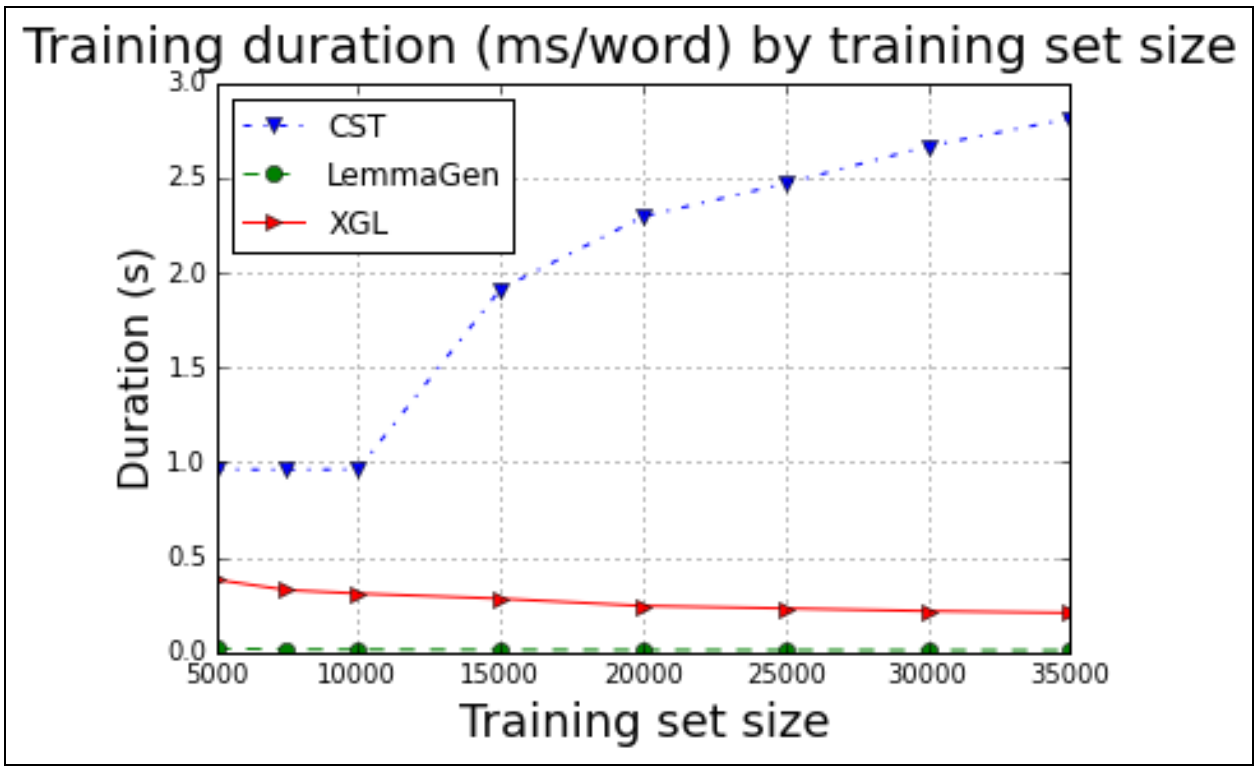

Figure 26: Training Duration (mS/word) by Training Set Size

At the maximum training set size of 35000 word-lemma pairs, the CST Lemmatiser was the slowest of the lemmatisers at 2.81 ms/word, and the LemmaGen was the fastest lemmatiser overall at 10 *µ*s/word. The slowness in the CST lemmatiser is attributed to its rules pruning process. The CST lemmatiser does a number of iterations using entropy maximisation in its tree pruning process. The duration of this pruning increased with the size of the training set. The XGL lemmatiser, at 207 *µ*s/word, was more than an order of magnitude slower than the LemmaGen lemmatiser.

Testing for the statistical significance of the results using the Wilcoxon test showed significant statistical differences between the lemmatiser, with p-values of 0.005, which is an order of magnitude below the 0.05 required for statistical significance.

6.3.3.1.2 Lemmatisation Duration

Lemmatisation duration indicates how fast the lemmatiser generates the lemma for a word. In this scenario, it was the XGL lemmatiser that was slow. It was orders of magnitude slower that the other lemmatisers.

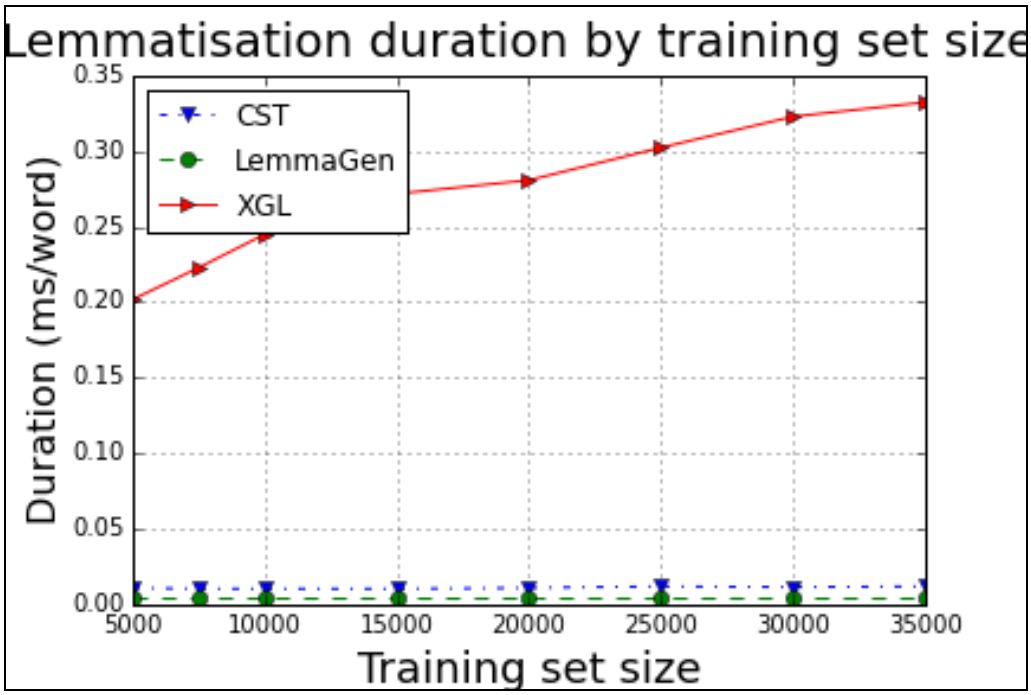

Figure 27: Average Lemmatisation Duration (mS/word) by Training Set Size

The fastest lemmatiser was the LemmaGen lemmatiser at 5 *µ*s/word, followed by the CST lemmatiser at 12 *µ*s/word. The XGL trailed at 332 *µ*s/word.

Testing for the statistical significance of the results using the Wilcoxon test showed that the lemmatisation results were statistically significant as all the p-values were 0.005; orders of magnitude were below the threshold of 0.05.

6.3.3.2 Conclusions on Execution Durations

The XGL lemmatiser is a slow lemmatiser because it is a Python script and has not been compiled to machine code. However, it is not as slow as the CST lemmatiser for the training stage.

LemmaGen could be considered a fast lemmatiser for isiXhosa, as it had the smallest execution times for both training and lemmatisation.

6.3.3.3 Memory Consumption

Memory consumption is one of the key measures in the performance of a lemmatiser. Again, results are presented for training memory consumption and lemmatisation memory consumption.

6.3.3.3.1 Training Memory Consumption

The graph below (Figure 28) shows the average memory usage per word-lemma pair for the three lemmatisers in relation to training set size.

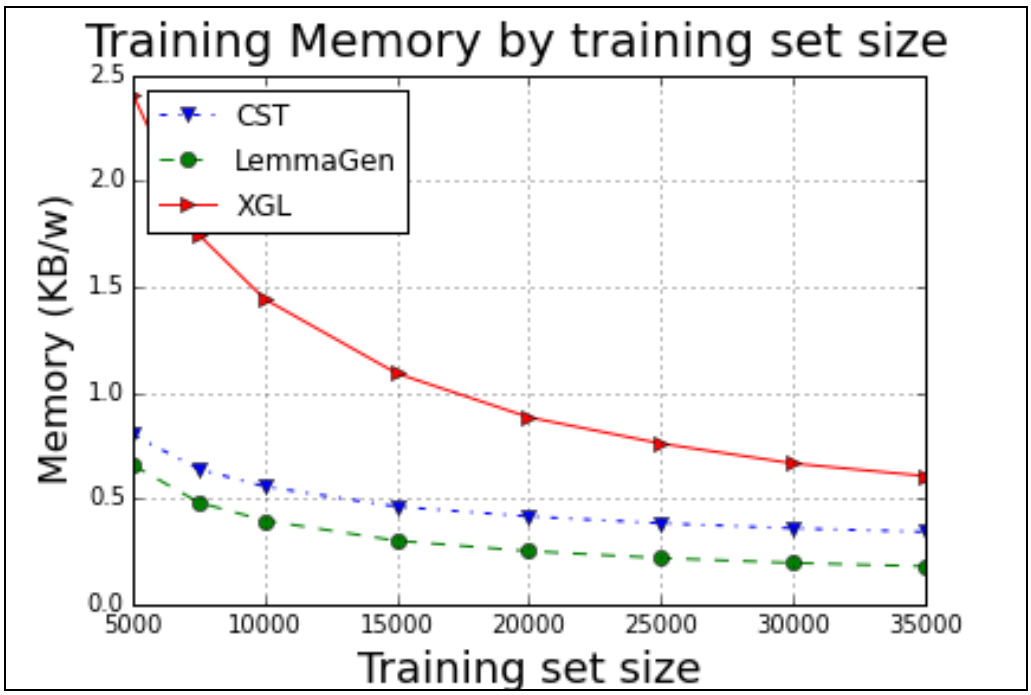

Figure 28: Average Training Memory (KB/word) by Training Set Size

The XGL Lemmatiser used the most memory at 2.4 KB/word at 5000 word-lemma pairs, and LemmaGen was the most efficient with memory usage starting at 659 bytes/word as compared to the CST Lemmatiser's 804 bytes/word. As the size of the training set increased, the differences reduced dramatically, but they were still visible. For the maximum training set size of 35000 word lemma pairs, LemmaGen still used the least memory per word at 181 bytes/word, followed by the CST at 344 bytes/word. The XGL was still the least economical with memory at 608 bytes/word.

Testing for the statistical significance of the results using the Wilcoxon test showed significant statistical differences between the lemmatisers.

6.3.3.3.2 Lemmatisation Memory Consumption

Lemmatisation memory usage gives an indication of how efficient the lemmatiser is with memory during the lemmatisation exercise. As is evident below, the XGL lemmatiser was order of magnitude higher in memory consumption than the other lemmatisers.

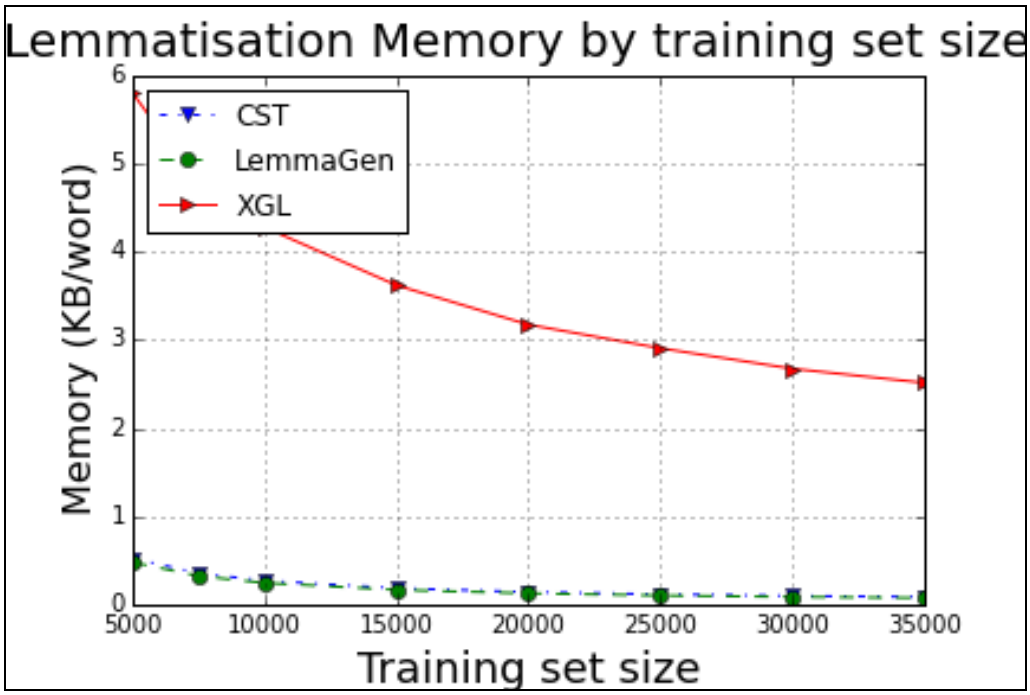

Figure 29: Lemmatisation Memory Usage (KB/word) by Training Set Size

The most efficient lemmatiser was the LemmaGen lemmatiser at 76 bytes/word at 35000 word-pairs, followed closely by the CST lemmatiser at 86 bytes/word at the same training set size. The XGL lemmatiser was the least memory efficient at 2.5KB/word at the same training set size.

6.3.3.4 Conclusions on Memory Usage

For both training and lemmatisation, the XGL lemmatiser proved the least efficient with memory usage. The LemmaGen was the most efficient, followed by the CST Lemmatiser.

6.4 Summary

The XGL lemmatiser showed superior accuracy to both the CST and the LemmaGen lemmatisers when measured against the testing corpus. However, when evaluated against the general corpus, the CST lemmatiser was the most accurate followed by the XGL. This showed that the XGL lemmatiser generalised to the language of isiXhosa better than the other lemmatisers. The CST lemmatiser generalised to the training data better, as seen from its superior accuracy against the general corpus. These differences could respectively be attributed to the fact that the XGL lemmatiser was designed with isiXhosa in mind, and the CST lemmatiser's tree pruning step during training was good at refining the model from the data. These could be the reasons for the XGL's better performance on the gold standard corpus, and the CST's better performance on the general corpus, which was used for training. The results trend on accuracy was also reflected on the F1-Score.

The XGL's usage of computing resources was orders of magnitude higher than the other two lemmatisers on resource usage, except against the CST lemmatiser during training. This was expected, as the CST lemmatiser did a number of iterations of tree pruning to maximise the entropy of its model, and the XGL was implemented as a Python script.

CHAPTER 7: CONCLUDING REMARKS

7.1 Summary of the work

IsiXhosa is one of the resource-scarce languages of South Africa when it comes to Human Language Technologies (HLT). As it is related to other Nguni languages i.e. isiZulu, Siswati, isiNdebele, any work done in isiXhosa could form the basis for work in other Nguni languages. A lemmatiser is one of the base components required in an HLT stack. Machine learning allows for the quick development of systems, as long as there is existing data from which that system could learn. In this case, the Language Resource Management Agency had lemmatisation data for isiXhosa. The study therefore was the investigation, implementation and performance evaluation of a machine learning lemmatiser for isiXhosa.

To structure this work, the following objectives were set and followed:

(1) To define the characteristics of a successful lemmatiser;

(2) To define the appropriate lemmas for isiXhosa in a Natural Language Processing (NLP) environment;

(3) To determine the best lemmatisation data features for isiXhosa;

(4) To design and implement a model for an isiXhosa machine learning lemmatiser, and

(5) To compare the implemented isiXhosa lemmatiser to existing machine learning lemmatisers.

The benchmark linguistic performance for an isiXhosa lemmatiser was the work conducted by Eiselen and Puttkammer (2014) in the generation of the data used in this study. That work was measured to have a lemmatisation accuracy of 79.82% when measured against a gold standard, the same standard that was used for this study.

To define the characteristics of a successful lemmatiser, a literature review was conducted and is detailed in chapter two. The review investigated general prevalent HLT techniques to gain an understanding of the field of HLT and the generic tools used in the field. Because the study is on automated lemmatisation, the chapter continued to look at lemmatisation studies worldwide, taking into consideration the nature of isiXhosa. This work addressed the first objectives of the study, which are to define features of a successful lemmatiser.

To design a lemmatiser with isiXhosa in mind, it is important to understand the language. Chapter three, therefore, dealt with the meaning of lemma for isiXhosa in a natural language processing (NLP) context. Because isiXhosa is a morphologically complex language with a number of defined word categories, the study had to adopt an approach to word categorisation so that the study could consider the nuances for each isiXhosa word category. This chapter addressed the second objective: defining the appropriate lemma for each isiXhosa word category in the NLP context.

Armed with this linguistic knowledge on the NLP lemmas in isiXhosa, the data used to train the lemmatisers was explored in chapter four using the Lemmatisation corpus of the IsiXhosa NCHLT Annotated Text Corpora. The development of these corpora is described by Eiselen and Puttkammer (2014). The data was generated from the South African government isiXhosa documents, which restricted the study data to a subset of the isiXhosa test. This corpus consists of a general corpus of 50000 words and a testing corpus of 5000 words. The general corpus was earmarked for training the lemmatisers and the testing corpus was earmarked for evaluating the lemmatisers. To maintain the validity of the study, the analysis work was done on the training corpus only, i.e. the general corpus. The work consisted primarily of the statistical analysis of the affixes identifiable in the data, and how much of the data they covered. The study also looked at the length of the word and compared it with the size of the affixes. This chapter addressed the third study objective: identifying the best data features for an isiXhosa lemmatiser.

Having identified the features that would work best to identify isiXhosa lemma transformation classes, the nature of an isiXhosa lemma, and the characteristics needed for a lemmatiser, an isiXhosa machine learning lemmatiser could be designed, modelled and implemented from scratch. This work is presented in chapter five: isiXhosa graphical lemmatiser. Chapter five addresses the fourth objective of the study, actually designing, modelling and implementing an isiXhosa lemmatiser.

With a working isiXhosa lemmatiser, evaluation against the two control lemmatisers began. The experimental design and the results are presented in chapter six. This chapter addresses the last objective of the study: an evaluation of the performance of a machine learning lemmatiser for isiXhosa.

7.2 Main Findings

A lexicalised probabilistic graphical lemmatiser for isiXhosa, the XGL, was investigated, designed, implemented, and evaluated.

Chapter two of the thesis provides answers to the first question of the study: **What characteristics do the most successful lemmatisers have?** It became apparent from the study that the best lemmatisers approach lemmatisation as a classification problem. The prevalent class type was that of a word to the lemma transformation class. It was also clear that successful machine learning lemmatisers use a lexicon, and that a machine learning lemmatiser that is inspired by linguistic rules, provides for better results. The use of word category information also was shown to improve results. In addition, this study assisted in identifying control lemmatisers that the isiXhosa lemmatiser would need to be benchmarked against. As the benchmark lemmatisers had to be freely available, the CST lemmatiser and LemmaGen lemmatiser were chosen. The CST lemmatiser was chosen because it has been used extensively and the LemmaGen lemmatiser was chosen because it showed good performance and had been tested thoroughly, especially for the use of computing resources.

In chapter three, the answers to the question about **the appropriate lemma for isiXhosa in an NLP context** were provided. The study first confirmed isiXhosa to be a prefixal language with the concordial agreement constituting the major part of the language structure and permeating almost all the isiXhosa word categories. The study adopted the word categories of Pahl (1982) and Louw et al. (1984). The lemmas for many of the isiXhosa word categories were identified as word stems; the words that are not inflected from stem, but are free morphemes, were identified to lemmatise to the full word form. There are also cases where the words in a category are a closed set. In this case, and because the question is on NLP lemmas, a case was made for lemmatising those to their full word form as well. It was also noted that certain derivations concatenate morphemes to form or extend the function of an affix.

The question regarding **good data features for an isiXhosa lemmatiser** was answered in chapter four. Because, in this study, lemmatisation is modelled as finding transformation classes, it was clear that the affixes would play a pivotal part in the search for the right class. The prefix proved to have the greatest coverage of the text, as confirmed by the study on the nature of the isiXhosa lemma. Circumfixes also showed the next most prevalent, with suffixes playing a small part when they are on their own. The correlation between word length and affix length was high enough to suggest that word length should also be used in finding the right lemmatisation transformation class for a word.

Therefore, the features to finding a lemmatisation transformation class were clear. They are prefixes, suffixes and word length in order of priority.

The question of **how the features should be structured in a model** was answered in chapter five. The design of the isiXhosa Graphical Lemmatiser (XGL) showed that the transformation

classes should be indexed in a hierarchical manner by circumfix, where a prefix is modelled as a circumfix with a blank suffix component and a suffix as a circumfix with a blank prefix component. The hierarchical structure was motivated by the fact that affixes are made up of concatenated morphemes, and the concatenation is left to right for prefixes and right to left for suffixes. The XGL was lexicalised because in the chapter establishing the lemma for isiXhosa it was found that some categories of words lemmatise to full wordform.

The word length statistics were linked to the transformation class as the mean and standard deviation of the lengths of the word belonging to the class. Prevalence of a class was also kept.

The question of the **best way to model an isiXhosa lemmatiser** with the implementation of an isiXhosa Graphical Lemmatiser was also answered in chapter five. The lemmatiser was modelled as a lexicalised probabilistic graphical classifier. The XGL lemmatiser was implemented as three components; two are used to create the lemmatisation model from training data, and the third component is used to lemmatise isiXhosa words. The isiXhosa lemmatiser is written in Python programming and has not been compiled or optimised for computing resource performance. The model is, however, designed for linguistic performance.

In chapter six, the question **on how the performance of the implemented isiXhosa lemmatiser compares to existing similar lemmatisers on the lemmatisation of isiXhosa**, is answered. To ensure the validity of the study, the experimental design incorporated splitting the data into three: development data, validation data and evaluation data. To ensure that the development work did not affect validating and evaluating the lemmatisers, all the development data came from the general corpus only. The training data for validation and evaluation came from the general corpus and the testing data for validation and evaluation came from the testing corpus. This meant that the testing corpus had to be split in two to prevent the evaluation tests from being tainted by the validation process. When tested against the corpus used in the training, i.e. the general corpus, the accuracy rate of the XGL lemmatiser was below that of the CST, but not as bad as the LemmaGen lemmatiser.

At the maximum of 35000 training pairs, the XGL lemmatiser achieved an average overall accuracy rate of 66.88%, the CST lemmatiser achieved 71.8% and the LemmaGen achieved 63.89%. However, when evaluated on a gold standard, the XGL lemmatiser performed better that the other lemmatisers. All the lemmatisers had improved results when evaluated against the gold standard. The XGL achieved 83.19%, the LemmaGen followed at 80.6%, and the CST Lemmatiser trailed at 73.13%. The F1-Score followed a similar trend with the CST Lemmatiser outperforming the XGL lemmatisers when evaluated against the general corpus, but the XGL

outperformed the other two lemmatisers when evaluated against the gold standard. The results were 0.86 for the XGL, 0.85 for the LemmaGen and 0.77 for the CST lemmatiser.

There was a significant difference in computer resource usage. The LemmaGen lemmatiser was fast and efficient with memory usage during both the training and lemmatization phases. The CST Lemmatiser was orders of magnitude slower than the other lemmatisers during training, but slightly slower than the LemmaGen during lemmatization. The CST's memory usage was slightly worse than that of the LemmaGen, and the XGL was an order of magnitude slower that LemmaGen during training. The XGL was the slowest during lemmatization.

From the results, one can surmise that the XGL lemmatiser generalized better to the language of isiXhosa than the other two lemmatisers, hence the better performance when evaluated against the gold standard. This is not a surprise as the lemmatiser was designed with isiXhosa in mind. However, the CST lemmatiser did a better model of the training data than the other lemmatisers did, hence its better performance when evaluated on the general corpus.

The XGL lemmatiser was not implemented for optimal computing resource utilisation, thus its dismal performance on computing resources.

7.3 Evaluation of the Hypothesis

The research hypothesis was that a machine learning lemmatiser specifically designed for isiXhosa will perform significantly better linguistically than existing lemmatisers in the lemmatisation of isiXhosa.

Therefore, the null hypothesis was that a machine learning lemmatiser specifically designed for isiXhosa will not perform significantly better linguistically than existing lemmatisers in the lemmatisation of isiXhosa.

The linguistic performance metrics used in the study are the accuracy and the F1-Score. The linguistic evaluation sample was the testing corpus. The isiXhosa data used in the study was extracted from government documents. Although this is not a complete representation of isiXhosa text, it suffices for the study.

The XGL lemmatiser did indeed perform linguistically better than the other lemmatisers. The XGL's accuracy was 83.19% compared to LemmaGen's accuracy rate of 80.6% and the CST Lemmatiser's rate of 73.13%. On the F1-Score, again the XGL performed better than the other lemmatisers with an F1-Score of 0.86 compared with 0.85 for LemmaGen and 0.77 for the CST lemmatiser.

The study used the Wilcoxon signed rank test to test for significance. The comparative results gave p-values of 0.005 in all the comparisons. Because this is below the threshold of 5%, it confirmed that the results were significant.

In conclusion, the null hypothesis is rejected because the XGL lemmatiser performed better than the other lemmatisers and the results passed the 95% threshold for significance.

7.4 Future Work

This work was restricted to one language (isiXhosa) using a small training set made up primarily of government data. Future work could include testing the XGL on a more balanced and considerably larger data set, as well as testing XGL's performance in other closely related languages.

The data source used in the study does contain parts-of-speech, but this was not used in this study. This could be an enhancement to the XGL as incorporating parts of speech tags to lemmatisation has been proven to improve lemmatisation accuracy.

The XGL Lemmatiser's implementation was not optimised for computer resource performance. Implementing the lemmatiser in a faster programming language using optimal data structures could improve the performance of the lemmatiser.

An enhancement to the XGL could be the automatic selection of the confidence threshold based on the data.

The XGL lemmatiser showed an anomaly at a threshold of 1.0, where the accuracy jumped from 89.8% at a threshold of 0.9999 to 95.9% a threshold of 1.0. This anomaly needs investigation, preferably with a more balanced data set.

The XGL lemmatiser could be improved with tree pruning, as is used by the CST Lemmatiser, to improve the model.

The XGL lemmatiser is biased toward prefixing languages; however, it could be bootstrapped to suffixing languages, or even infixing languages.

7.5 Conclusions

The development of the XGL lemmatiser has proved that it is possible to develop HLT resources for resource scarce languages.

This work is considered another brick towards building resources for resource scarce languages. By using the linguistic knowledge of a language in the design and development of a machine learning HLT resource, we showed that it is possible to develop tools for resource scarce languages that better match language characteristics.

BIBLIOGRAPHY

Aduris, I., Aldezabal, I., Alegria, I., Artola, X., Ezeiza, N., and Urizar, R., 1996. EUSLEM: A lemmatiser/tagger for Basque. In *Procedings of EURALEX '96*. pp. 17–26.

Agic, Z., Ljubesic, N., and Danijela Merkler, 2013. Lemmatisation and Morphosyntactic Tagging of Croatian and Serbian. In *Proceedings of the 4th Biennial International Workshop on Balto-Slavic Natural Language Processing*. Sofia, Bulgaria, pp. 48–57.

Assembly, C., 1996. *Constitution of the Republic of South Africa*, South Africa.

Bates, M., 1995. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, 92(22), pp.9977–9982.

Belkin, M. and Goldsmith, J., 2002. Using eigenvectors of the bigram graph to infer morpheme identity. *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, 6(July), pp.41–47.

Bender, E.M., 2013. *Linguistic Fundamentals for Natural Language Processing: 100 Essentials from Morphology and Syntax*, Morgan & Claypool.

Berger, A., Pietra, V., and Pietra, S., 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1), pp.39–71.

Booij, G., 2012. *The grammar of words: an introduction to linguistic morphology* Third Edit., Oxford University Press.

van den Bosch, A. and Daelemans, W., 2009. Memory-based Morphological Analysis. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*. Maryland: ACL, pp. 285–292.

Bosch, S., Jones, J., Pretorius, L., and Winston Anderson, 2006. Resource Development for South African Bantu Languages: Computational Morphological Analysers and Machine - Readable Lexicons. In *Proceedings on the Workshop on Networking the Development of Language Resources for African Languages. 5th International Conference on Language Resources and Evaluation*. pp. 38–43.

Bosch, S., Pretorius, L., and Fleisch, A., 2008. Experimental Bootstrapping of Morphological Analysers for Nguni Languages. *Nordic Journal of African Studies*, 17(2), pp.66–88.

Botha, G., Zimu, V., and Barnard, E., 2007. Text-based language identification for South African languages. *SAIEE Africa Research Journal*, 98(4), pp.141–146.

Boyce, W.B., 1844. *A Grammar of the Kaffir Language* Second Edi., London: Wesleyan Missionary Society.

Brahmi, A., Ech-Cherif, A., and Benyettou, A., 2013. An Arabic Lemma-based Stemmer for Latent Topic Modeling. *The International Arab Journal of Information Technology*, 10(2), pp.160–168.

Brits, K., Pretorius, R., and van Huyssteen, G.B., 2005. Automatic lemmatization in Setswana: towards a prototype. *South African Journal of African Languages*, 1, pp.37–48.

Chan, E., 2008. *Structures and Distribution in morphology learning*. University of Pennsylvania.

Chrupala, G., 2006. Simple data-driven context-sensitive lemmatization. *Procesamiento del Lenguaje Natural*, 37, pp.121–127.

Collins, M., 2003. Head-Driven Statistical Models for Natural Language Parsing. *Computational*

Linguistics, 29(4), pp.589–637.

Collobert, R., Weston, J., Karlen, M., Kavukcuoglu, K., and Kuksa, P., 2011. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12, pp.2493–2537.

Compton, P., Edwards, G., Kang, B., Lazarus, L., Malor, R., Menzies, T., Srinivasan, P., A, P., and Sammut, S., 1991. Ripple down rules : possibilities and limitations. In *Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop*. Calgary, Canada, University of Calgary, pp. 6–1.

Creutz, M. and Lagus, K., 2005. Inducing the morphological lexicon of a natural language from unannotated text. *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*, pp.106–113.

Creutz, M. and Lagus, K., 2002. Unsupervised Discovery of Morphemes. In *Procedings of the 6th Workshop of the ACL Special Interest Group in Computational Phonology*. Philadelphia, pp. 21–30.

Daelemans, W., Groenewald, H.J., and van Huyssteen, G.B., 2009. Prototype-based Active Learning for Lemmatization. *Proceedings of the RANLP'2009 International Conference Recent Advances in Natural Language Processing*, pp.65–70.

Demšar, J., 2006. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7, pp.1–30.

Eiselen, R. and Puttkammer, M.J., 2014. Developing Text Resources for Ten South African Languages. In N. C. (Conference Chair) et al., eds. *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*. Reykjavik, Iceland: European Language Resources Association (ELRA), pp. 3698–3703.

Erjavec, T. and Dzeroski, S., 2004. Machine learning of morphosyntactic structure: Lemmatizing unknown Slovene words. *Applied Artificial Intelligence*, 18(1), pp.17–41.

van Eynde, F., Zavrel, J., and Daelemans, W., 2000. Part of Speech Tagging and Lemmatisation for the Spoken Dutch Corpus. *Proceedings of LREC*, pp.1427–1433.

Gesmundo, A. and Samardžić, T., 2012. Lemmatisation as a tagging task. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers*. Association for Computational Linguistics, pp. 368–372.

Gildea, D. and Jurafsky, D., 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3), pp.245–288.

Gouws, R.H. and Prinsloo, D.J., 2005. *Principles and practice of South African lexicography.*, Stellenbosch: African Sun Media.

Groenewald, H.J., 2006. *Automatic Lemmatisation for Afrikaans*. North-West University.

Groenewald, H.J., 2007. Educating Lia: The development of a linguistically accurate memory-based lemmatiser for Afrikaans. In Z. Shi, K. Shimohara, & D. Feng, eds. *Intelligent Information Processing III*. Boston: Springer US, pp. 431–440.

Groenewald, H.J., 2009. Using Technology Transfer to Advance Automatic Lemmatisation for Setswana. In *Proceedings of the EACL 2009 Workshop on Language Technologies for African Languages – AfLaT 2009, pages 32–37, Athens, Greece, 31 March 2009.* pp. 32–37.

Grünwald, P., 2005. Introducing the Minimum Description Length Principle. In P. D. Grünwald, I. J. Myung, & M. A. Pitt, eds. *Advances in Minimum Description Length: Theory and Application*. Cambridge, MA,USA: MIT Press, pp. 3–21.

Hammarstrom, H., Forkel, R., Haspelmath, M., and Nordhoff, S., 2014. *Glottolog 2.3*, Leipzig:

Max Planck Institute for Evolutionary Anthropology. Available at: http://glottolog.org.

Have, C.T., 2009. Stochastic Definite Clause Grammars. *Recent Advances in Natural Language Processing 2009*, pp.139–144.

Hrnčič, D., Mernik, M., Bryant, B.R., and Javed, F., 2012. A memetic grammar inference algorithm for language learning. *Applied Soft Computing*, 12(3), pp.1006–1020.

van Huyssteen, G.B. and Snyman, D.P., 2012. Cross-Lingual Genre Classification for Closely Related Languages. In *Proceedings of the 22nd Annual Symposium of the Pattern Recognition Association of South Africa*. pp. 132–137. Available at: rma.nwu.ac.za.

Ingason, A.K., Helgadóttir, S., Loftsson, H., and Rögnvaldsson, E., 2008. A mixed method lemmatization algorithm using a hierarchy of linguistic identities (HOLI). In A. Ranta & B. Nordström, eds. *Advances in Natural Language Processing, 6th International Conference on NLP, GoTAL 2008, Proceedings*. Gothenburg, Sweden., pp. 205–216.

Jones, J., Podile, K., and Puttkammer, M., 2005. Challenges relating to standardisation in the development of an isiXhosa spelling checker. *South African Journal of African Languages*, 25(1), pp.1–10.

Jongejan, B. and Dalianis, H., 2009. Automatic training of lemmatization rules that handle morphological changes in pre-, in- and suffixes alike. In *Proc. 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP. Suntec*. Suntec, Singapore, pp. 145–153.

Jongejan, B. and Haltrup, D., 2013. *the CST Lemmatiser*, Denmark. Available at: http://www.cst.dk.

Jurafsky, D. and Martin, J.H., 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.*, Upper Saddle River, New Jersey: Pearson Prentice Hall.

Juršič, M., Mozetič, I., Erjavec, T., and Lavrač, N., 2010. LemmaGen: Multilingual Lemmatisation with Induced Ripple-Down Rules. *Journal of Universal Computer Science*, 16(9), pp.1190–1241.

Karttunen, L., 1993. *Finite-state lexicon compiler. Technical Report ISTL-NLTT-1993-04-02*, Palo Alto, CA.

Kosch, I.M., 2006. *Topics in Morphology in the African Language Context*, Pretoria: Unisa Press.

Koskenniemi, K., 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th international conference on Computational linguistics* -. pp. 178–181.

Lewis, M.P., Simons, G.F., and Fennig, C.D.,Eds., 2014. *Ethnologue: Languages of the World* Seventeent., Dallas Texas: SIL International. Available at: http://www.ethnologue.com.

Louw, J.A., Finlayson, R., and Satyo, S.C., 1984. *Xhosa Guide 3 for XHA100-F*, Pretoria: University of South Africa.

Maho, J., 2009. *NUGL Online*, Available at: http://goto.glocalnet.net/mahopapers/nuglonline.pdf [Accessed April 21, 2014].

Manning, C.D., Raghavan, P., and Schutze, H., 2009. *An Introduction to Information Retrieval* Online edi., Cambridge, England: Cambridge University Press.

Manning, C.D. and Schütze, H., 1999. *Foundations of Statistical Natural Language Processing*, MIT Press.

McKinney, W., 2010. Data Structures for Statistical Computing in Python. In *Proceedings of the 9th Python in Science Conference*. pp. 51–56.

McLaren, J., 1948. *A Xhosa Grammar* Third. G. . Welsh, ed., London; Cape Town: Longmans, Green.

Meinhof, C., 1932. *Introduction to the Phonology of the Bantu languages.*, Berlin: Dietrich Reimer/Erns Vohsen.

Mini, B.M. and Tshabe, S.L.,Eds., 2003. *The Greater Dictionary of isiXhosa: Volume 2 (K to P)*, Alice: University of Fort Hare.

Mitchell, T.M., 1997. *Machine Learning*, WCB/McGraw-Hill.

Mncube, F.S.M., *Xhosa manual*, Johannesburg: Juta & Company Ltd.

Nguyen, D.T., Nguyen, K. Van, and Pham, T.T., 2013. Implementing a Subcategorised Probabilistic Definite Clause Grammar for Vietnamese Sentence Parsing. *International Journal on Natural Language Computing*, 2(4), pp.1–19.

Orosz, G. and Novák, A., 2013. PurePos 2.0: a hybrid tool for morphological disambiguation. *RANLP*, (September), pp.539–545.

Pahl, H.W., 1982. *IsiXhosa* D. M. Ntusi & S. M. Burns-Ncamashe, eds., King Williams Town: Educum Publishers.

Pahl, H.W., Pienaar, A.M., and Ndungane, T.A.,Eds., 1989. *The Greater Dictionary of Xhosa: Volume 3 (Q to Z)*, Alice: University of Fort Hare.

Pasha, A., Al-badrashiny, M., Diab, M., Kholy, A. El, Eskander, R., Habash, N., Pooleery, M., Rambow, O., and Roth, R.M., 2014. MADAMIRA : A Fast , Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proc. LREC. 2014*. pp. 1094–1101.

de Pauw, G. and de Schryver, G., 2009. African Language Technology: The Data-Driven Perspective. In *Second Colloquium on Lesser Used Languages and Computer Linguistics*. European Academy, pp. 79–96.

de Pauw, G. and de Schryver, G., 2008. Improving the Computational Morphological Analysis of a Swahili Corpus for Lexicographic Purposes. *Lexikos*, 18(July), pp.303–318.

Perera, P. and Witte, R., 2005. A Self-Learning Context-Aware Lemmatizer for German. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*. Vancouver, Canada, pp. 636–643.

Pérez, F. and Granger, B.E., 2007. IPython: A System for Interactive Scientific Computing. *Computing in Science and Engineering*, 9(3), pp.21–29.

Plisson, J., Lavrac, N., and Mladenic, D., 2004. A Rule based Approach to Word Lemmatization. In *Proceidings of the 7th International Multi-Conference Information Society IS 2004*. pp. 83–86.

Pretorius, L. and Bosch, S.E., 2005. Finite-State Computational Morphology: An Analyzer Prototype For Zulu. *Machine Translation*, 18(3), pp.195–216.

Prinsloo, D.J., 2011. A Critical Analysis of the Lemmatisation of Nouns and Verbs in isiZulu. *Lexikos*, 21, pp.169–193.

Prinsloo, D.J. and de Schryver, G.-M., 2004. Spellcheckers for the South African languages , Part 2 : The utilisation of clusters of circumfixes. *South African Journal of African Languages*, 1, pp.83–94.

Rasinger, S.M., 2013. *Quantitative Research in Linguistics: An Introduction* 2nd Editio., London: Bloomsbury Academic.

Ratnaparkhi, A., 1998. *Maximum entropy models for natural language ambiguity resolution*.

University of Pennsylvania.

Rissanen, J., 1978. Modelling by the shortest data description. *Automatica*, 14, pp.465–471.

van Rossum, G., 2007. Python Programming Language. *USENIX Annual Technical Conference*, 41.

Russell, S. and Norvig, P., 2014. *Artificial Intelligence: A Modern Approach, Third edition* Prentice H., London: Prentice Hall Press.

Saraswathi, S. and Geetha, T. V., 2007. Comparison of performance of enhanced morpheme-based language model with different word-based language models for improving the performance of Tamil speech recognition system. *ACM Transactions on Asian Language Information Processing*, 6(3), p.Article number 9.

Schmid, H., 1995. Improvements in part-of-speech tagging with an application to German. *In Proceedings of the ACL SIGDAT-Workshop*.

Shalonova, K. and Golenia, B., 2010. Weakly Supervised Morphology Learning for Agglutinating Languages Using Small Training Sets. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling)*. pp. 976–983.

Sharma Grover, A., van Huyssteen, G.B., and Pretorius, M., 2010. South African human language technologies audit. In *7th International Conference on Language Resources and Evaluation*. Valetta, Malta, pp. 2847–2850.

Shen, L., Satta, G., and Joshi, A.K., 2007. Guided learning for bidirectional sequence classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, pp. 760–767.

South African Department of Arts and Culture and Centre for Text Technology (CTexT) North-West University South Africa, 2013. isiXhosa NCHLT Annotated Text Corpora. Available at: http://rma.nwu.ac.za/index.php/isixhosa-nchlt-annotated-text-corpora.html.

Spiegler, S., van der Spuy, A., and Flach, P.A., 2010. Ukwabelana - An open-source morphological Zulu corpus. *1020 Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, (August), pp.1020–1028.

Spiegler, S.R., 2011. *Machine Learning for the analysis of morphologically complex languages*. PhD Thesis, University of Bristol.

Statistics South Africa., 2012. *Census 2011: Census in brief*, Available at: http://www.statssa.gov.za/Census2011/Products/Census_2011_Census_in_brief.pdf.

Suhartono, D., Christiandy, D., and Rolando, R., 2014. Lemmatization Technique in Bahasa: Indonesian Language. *Journal of Software*, 9(5), pp.1202–1210.

Tamburini, F., 2011. The AnIta Lemmatiser. *Working Notes of EVALITA*.

Theron, P. and Cloete, I., 1997. Automatic acquisition of two-level morphological rules. In *Proceedings of the fifth conference on Applied Natural Langauge Processing*. Washington, DC: Morgan Kaufmann Publishers, San Francisco, CA, pp. 103–110.

Tshabe, S.L. and Shoba, F..,Eds., 2006. *The Greater Dictionary of isiXhosa: Volume 1 (A to J)*, Alice: University of Fort Hare.

Welman, C., Kruger, F., and Mitchell, B., 2005. *Research Methodology* 3rd Editio., Cape Town: Oxford University Press, Southern Africa.

Wilcoxon, F., 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, 1(6), pp.80–83.

Zipf, G., 1945. The meaning-frequency relationship of words. *Journal of General Psychology*,

33, pp.251–256.