NORTH-WEST UNIVERSITY
YUNIBESITI YA BOKONE-BOPHIRIMA
NOORDWES-UNIVERSITEIT

The effects of part-of-speech tagging on text-to-speech synthesis for resource-scarce languages

G. I. Schlünz
22105034

Dissertation submitted in partial fulfilment of the requirements for the degree Master of Science in Engineering Sciences at the Potchefstroom campus of the North-West University.

Supervisor:	Prof E. Barnard
Co-supervisor:	Prof G. B. van Huyssteen
Assistant Supervisor:	Mr D. R. van Niekerk

November 2010

Acknowledgements

- I lift my hands to our Redeemer Jesus Christ, whose *"divine power has given to us all things that pertain to life and godliness, through the knowledge of Him who called us by glory and virtue"* (2 Peter 1:3 NKJV). He has not only been my strength during this Master's study, but He is teaching me how to become a master at living life!
- I thank Prof Gerhard van Huyssteen for supervising my work—for relating in such a real way, understanding my circumstances and keeping me on track!
- I am indebted to Prof Etienne Barnard for his valuable insights at critical times.
- A big thank you also goes to Daniel van Niekerk, who answered a lot of questions and helped with a lot of practicalities.
- I am grateful for the incredible work environment at the Human Language Technologies Research Group of the Meraka Institute, CSIR—the relaxed and free atmosphere is such a privilege!

Abstract

In the world of human language technology, resource-scarce languages (RSLs) suffer from the problem of little available electronic data and linguistic expertise. The Lwazi project in South Africa is a large-scale endeavour to collect and apply such resources for all eleven of the official South African languages. One of the deliverables of the project is more natural text-to-speech (TTS) voices. Naturalness is primarily determined by prosody and it is shown that many aspects of prosodic modelling is, in turn, dependent on part-of-speech (POS) information. Solving the POS problem is, therefore, a prudent first step towards meeting the goal of natural TTS voices.

In a resource-scarce environment, obtaining and applying the POS information are not trivial. Firstly, an automatic tagger is required to tag the text to be synthesised with POS categories, but state-of-the-art POS taggers are data-driven and thus require large amounts of labelled training data. Secondly, the subsequent processes in TTS that are used to apply the POS information towards prosodic modelling are resource-intensive themselves: some require non-trivial linguistic knowledge; others require labelled data as well.

The first problem asks the question of which available POS tagging algorithm will be the most accurate on little training data. This research sets out to answer the question by reviewing the most popular supervised data-driven algorithms. Since literature to date consists mostly of isolated papers discussing one algorithm, the aim of the review is to consolidate the research into a single point of reference. A subsequent experimental investigation compares the tagging algorithms on small training data sets of English and Afrikaans, and it is shown that the hidden Markov model (HMM) tagger outperforms the rest when using both a comprehensive and a reduced POS tagset.

Regarding the second problem, the question arises whether it is perhaps possible to circumvent the traditional approaches to prosodic modelling by learning the latter directly from the speech data using POS information. In other words, does the addition of POS features to the HTS context labels improve the naturalness of a TTS voice? Towards answering this question, HTS voices are trained from English and Afrikaans prosodically rich speech. The voices are compared with and without POS features incorporated into the HTS context labels, analytically and perceptually. For the analytical experiments, measures of prosody to quantify the comparisons are explored. It is then also noted whether the results of the perceptual experiments correlate with their analytical counterparts. It is found that, when a minimal feature set is used for the HTS context labels, the addition of POS tags does improve the naturalness of the voice. However, the same effect can be accomplished by including segmental counting and positional information instead of the POS tags.

Key words: part-of-speech tagging, text-to-speech synthesis, resource-scarce language, naturalness, prosody, HTS context labels.

Contents

Abstract **ii**

1 Introduction **1**

1.1 Background . . . 1

1.1.1 Natural Language Processing . . . 1

1.1.2 Text-to-Speech Synthesis . . . 5

1.2 Contextualisation . . . 8

1.2.1 Part-of-Speech Tagging . . . 8

1.2.2 Text-to-Speech Synthesis . . . 9

1.3 Problem Statement . . . 10

1.4 Research Questions . . . 11

1.5 Aims . . . 11

1.6 Hypotheses . . . 11

1.7 Contributions . . . 11

1.8 Research Methodology . . . 11

1.9 Overview . . . 12

2 A Literature Review of Part-of-Speech Tagging Algorithms **14**

2.1 Introduction . . . 14

2.2 Learning Paradigms . . . 15

2.2.1 Supervised Learning . . . 15

2.2.2 Unsupervised Learning . . . 15

2.2.3 Semisupervised Learning . . . 16

2.3 Supervised Tagging Algorithms . . . 17

2.3.1 Hidden Markov Models [11, 17, 83] . . . 17

2.3.2 Transformation-Based Learning [12, 13] . . . 19

2.3.3 Tree Tagging [65] . . . 20

2.3.4 Maximum Entropy [44, 59] . . . 22

2.3.5 Memory-Based Learning [22] . . . 24

2.3.6 Sparse Network of Winnows [63] . . . 25

2.3.7 Conditional Random Fields [48] . . . 26

2.3.8 Cyclic Dependency Networks [89] . . . 27

2.3.9 Support Vector Machines [6, 32] . . . 29

2.4 Conclusion . . . 30

3 Part-of-Speech Tagging in a Resource-Scarce Environment 33
3.1 Introduction . . . 33
3.2 Experiment 1: Comparing Supervised Taggers . . . 33
3.2.1 Setup . . . 34
3.2.2 Results . . . 38
3.3 Experiment 2: Reducing the Tagsets . . . 44
3.3.1 Setup . . . 44
3.3.2 Results . . . 45
3.4 Conclusion . . . 46

4 Part-of-Speech Effects on Text-to-Speech Synthesis 51
4.1 Introduction . . . 51
4.2 Common Setup . . . 51
4.2.1 Tagger . . . 51
4.2.2 Voices . . . 52
4.2.3 Analytical Test . . . 54
4.2.4 Perceptual Test . . . 55
4.3 Experiment 1: POS Effects Using Maximum Features . . . 56
4.4 Experiment 2: POS Effects Using Minimum Features . . . 57
4.5 Experiment 3: POS Effects Using a Less Accurate Tagger . . . 57
4.6 Experiment 4: Comparing Minimum and Maximum Features . . . 59
4.7 Conclusion . . . 59

5 Conclusion 61
5.1 Summary and Conclusions . . . 61
5.1.1 An Optimal Part-of-Speech Tagging Algorithm . . . 61
5.1.2 The Effects of Part-of-Speech Features in the HTS Labels . . . 63
5.2 Future Work . . . 64

Bibliography 65

A Part-of-Speech Tagsets 73

B Tables of Significance 78

List of Figures

1.1 The syntactic tree for the sentence `The cat chases a mouse` 2
1.2 Overlap in the fields of NLP, ASR and TTS 4
1.3 State-of-the-art speech synthesis techniques 7

2.1 TBL process diagram 19
2.2 A tree tagger decision tree 21
2.3 Left-to-right Conditional Markov Model 28
2.4 Bidirectional dependency network 28

3.1 Supervised tagging: overall accuracies using the full tagsets 41
3.2 Supervised tagging: overall accuracies using the reduced tagsets 47

List of Tables

2.1 Best scores for supervised POS tagging on large English training corpora 31
2.2 Best scores for supervised POS tagging on small training corpora 32

3.1 Statistics of the test data using the full tagsets 34
3.2 Supervised tagging: overall accuracies using the full tagsets 39
3.3 Supervised tagging: statistical significance using the full tagsets 40
3.4 Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **eng.wsj1** (using the full tagset) 42
3.5 Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **afr.nwu1** (using the full tagset) 43
3.6 Statistics of the test data using the reduced tagsets 44
3.7 Supervised tagging: overall accuracies using the reduced tagsets 45
3.8 Supervised tagging: statistical significance using the reduced tagsets 46
3.9 Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **eng.wsj2** (using the reduced tagset) 48
3.10 Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **afr.nwu2** (using the reduced tagset) 49

4.1 Features to be used in the HTS context labels 53
4.2 Contingency table for McNemar's test 55
4.3 Naturalness results when using maximum features 57
4.4 Naturalness results when using minimum features 58
4.5 Naturalness results when using a less accurate tagger 58
4.6 Results of the comparison between minimum and maximum features 59

A.1 English POS tagsets 73
A.2 Afrikaans POS tagsets 74

B.1 *t*-test table 78
B.2 Chi-square table 79

List of Abbreviations

AI	Artificial Intelligence
ASR	Automatic Speech Recognition
CE	Contrastive Estimation
CL	Computational Linguistics
CMM	Conditional Markov Model
CRF	Conditional Random Field
DSP	Digital Signal Processing
EM	Expectation Maximisation
G2P	Grapheme-to-Phoneme
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
HTS	HMM-Based Speech Synthesis System
LDA	Latent Dirichlet Allocation
MAP	Maximum a Posteriori
MaxEnt	Maximum Entropy
MBL	Memory-Based Learning
MEMM	Maximum Entropy Markov Model
MFCC	Mel-Frequency Cepstral Coefficients
MLE	Maximum Likelihood Estimation
MRF	Markov Random Field
MSE	Mean Squared Error
NLP	Natural Language Processing
POS	Part-of-Speech
RSL	Resource-Scarce Language
SLF	Self-Learned Features
SNoW	Sparse Network of Winnows
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TBL	Transformation-Based Learning
TTS	Text-to-Speech
WPDV	Weighted Probability Distribution Voting
WSJ	Wall Street Journal

Chapter 1

Introduction

1.1 Background

1.1.1 Natural Language Processing

Natural Language Processing (NLP) is a multi-disciplinary field that borrows from computer science, linguistics and cognitive psychology. It combines their theory with computation to process natural (human) language text. In other words, NLP entails the computational representation and analysis—that is understanding and generation—of the text [49].

Another closely related and overlapping field is Computational Linguistics (CL), which differs subtly from NLP in that it uses computation to understand linguistics better. Hence, computational representation and analysis are in this case the means and not the end as in NLP. Both fields have their roots in Artificial Intelligence (AI), which is the study of computational models of human cognition [90].

NLP processes text on different levels of linguistic analysis [49]:

Phonology This is the study of how speech sounds function and are organised in a particular natural language. Conversely, *phonetics* analyses the physical production of speech, independent of language [73]. Some important terminology are the following: A *phoneme* is the smallest theoretically contrastive unit (able to distinguish words) in the sound system of a language. A *phone* is the smallest physically identifiable unit (yet not able to distinguish words) in speech [71]. A phoneme is realised as one or more phones in different phonemic contexts or environments—these phones are termed *allophones* [68]. For example, the aspirated `[pH]` in `pin` and the unaspirated `[p]` in `spin` are allophones of the phoneme `/p/` [25].

Morphology The smallest meaningful unit in the grammar of a language is called a *morpheme* [70]. This level then performs morphological decomposition of words into *roots* and *affixes* to infer their internal structure [72]. Consider the example word `misjudged`. A root carries the principal part of meaning in the word, namely `judge`. An affix augments the meaning of the principal part. It can be a *prefix* that is prepended to the word, namely `mis-` meaning "wrong", or a *suffix* that is appended to the word, that is `-ed` indicating the past tense.

Lexicology Lexical analysis determines the underlying meaning or sense of individual words, typically by lookup in a dictionary called a *lexicon* [69]. If a word has multiple senses, it is

disambiguated at the semantic level.

Syntax This level infers the grammatical structure of the sentence, that is the structural dependencies among the constituent words. It includes, inter alia, the tagging of the words with Part-of-Speech (POS) categories, for example *noun*, *verb* and *preposition*, and the parsing of phrases such as *noun phrases*, *verb phrases* and *prepositional phrases*. The structure is most intuitively represented as a tree, of which an example can be seen in Figure 1.1. The (probable) grammatical order required of the parts of speech within these structures helps to eliminate the ambiguity of multiple such categories for a single word.

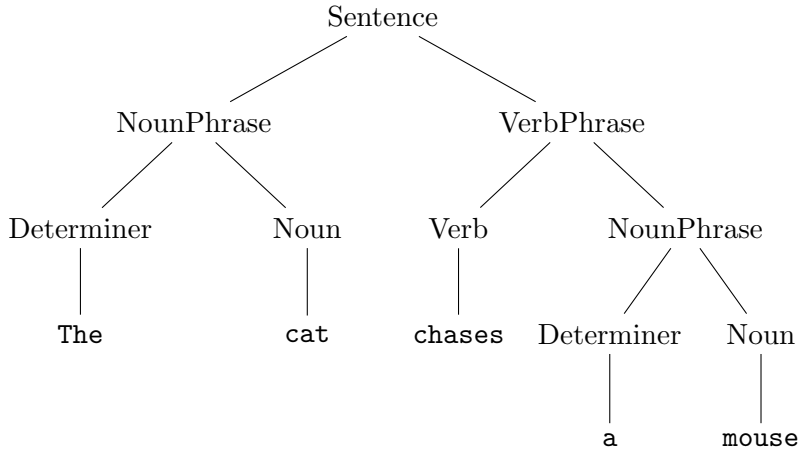


Figure 1.1: The syntactic tree for the sentence `The cat chases a mouse`

Semantics In general, this is the study of meaning of linguistic expressions. More narrowly defined, it is the study of word sense on the sentence level, not yet considering discourse and pragmatic factors (explanations to follow) [74]. At this level, the meaning of the remaining ambiguous words from the lexical stage are resolved by considering the interactions among the individual word senses in the sentence. This is called *word-sense disambiguation*. The representation of the meaning of words may be done using *predicate logic*. This decomposes a word into its basic properties or semantic primitives. When these primitives are shared among words in the lexicon, meaning can be unified across and inferences drawn from the words [49]. The sentence

$$\texttt{John is the father of Michael.} \tag{1.1}$$

could be represented with predicate logic as follows:

$$\begin{aligned}&\textsc{Relation}(\textsc{Object}(\textsc{Type}(\texttt{father}), \textsc{Agent}(\texttt{John})),\\ &\qquad\textsc{Object}(\textsc{Type}(\texttt{son}), \textsc{Agent}(\texttt{Michael})))\end{aligned} \tag{1.2}$$

The expression Agent is called a *predicate*, which assigns a property to its *argument* `John`.

Discourse Whereas syntax and semantics are, therefore, sentence-level analyses, this level of analysis functions on the whole document or discourse, connecting meaning (for example POS,

number agreement, gender, et cetera) across sentences. *Anaphora resolution* resolves pronoun references such as `She` in the sentences

```
The boy likes the girl.  She is pretty.
```
(1.3)

by using the gender attribute to assign it to the noun phrase `the girl`.

Pragmatics This is the study of meaning in context over and above that which can be captured by the text, for example the intent, plan and/or goal of the speaker, the status of the parties involved and other *world knowledge*. Pragmatics is in this way an explanation of how humans are able to overcome the inherent ambiguity in natural language sentences. Consider the following example:

```
The boy hit his little brother.  He cried.
```
(1.4)

The anaphora resolution of `He` cannot take place on the discourse level because the pronoun agrees in number and gender with both the subject and the object in the first sentence. Pragmatic knowledge that pain inflicted on a young child will normally lead to tears is required to associate `He` with the object `his little brother`.

It is important to note that the above process is not strictly sequential; many times the levels must interact "out of order" to extract meaning out of linguistic expressions [49]. Hence, a "higher" level will often be required to assist in the analysis of a "lower" one. In the sentence

```
The man entered the court.
```
(1.5)

pragmatic knowledge that the setting is a tennis tournament will help to disambiguate `court` to COURT-TENNIS and not COURT-LAW during semantic analysis.

The scope of NLP technically allows the natural language text to be spoken or written, but spoken language processing has split off into separate fields: Automatic Speech Recognition (ASR) for understanding and Text-to-Speech (TTS) for generation. These fields may be related back to their parent by considering them to *use* NLP, particularly in the stages where the speech is in text form—in the case of ASR this is in the second half of processing after recognition, and in the case of TTS it is in the first half before synthesis. The stages of acoustic recognition and synthesis borrow techniques from engineering to perform the digital signal processing (DSP) of the speech waveform.

Figure 1.2 shows a simplified block diagram of NLP to illustrate roughly where ASR and TTS overlap with their parent. It has been adapted from [94]. The blocks are the available NLP levels; arrows coming into a block indicates its input and arrows coming out indicates its output. The *dashed* arrows are the ASR flow (starting at the acoustic signal and ending at the words); the *solid* arrows are the TTS flow (starting at the words and ending at the acoustic signal) and the *dotted* arrows are the unused NLP flow (by its children fields). A digression on ASR is beyond the scope of this dissertation; TTS will be introduced in the next subsection.

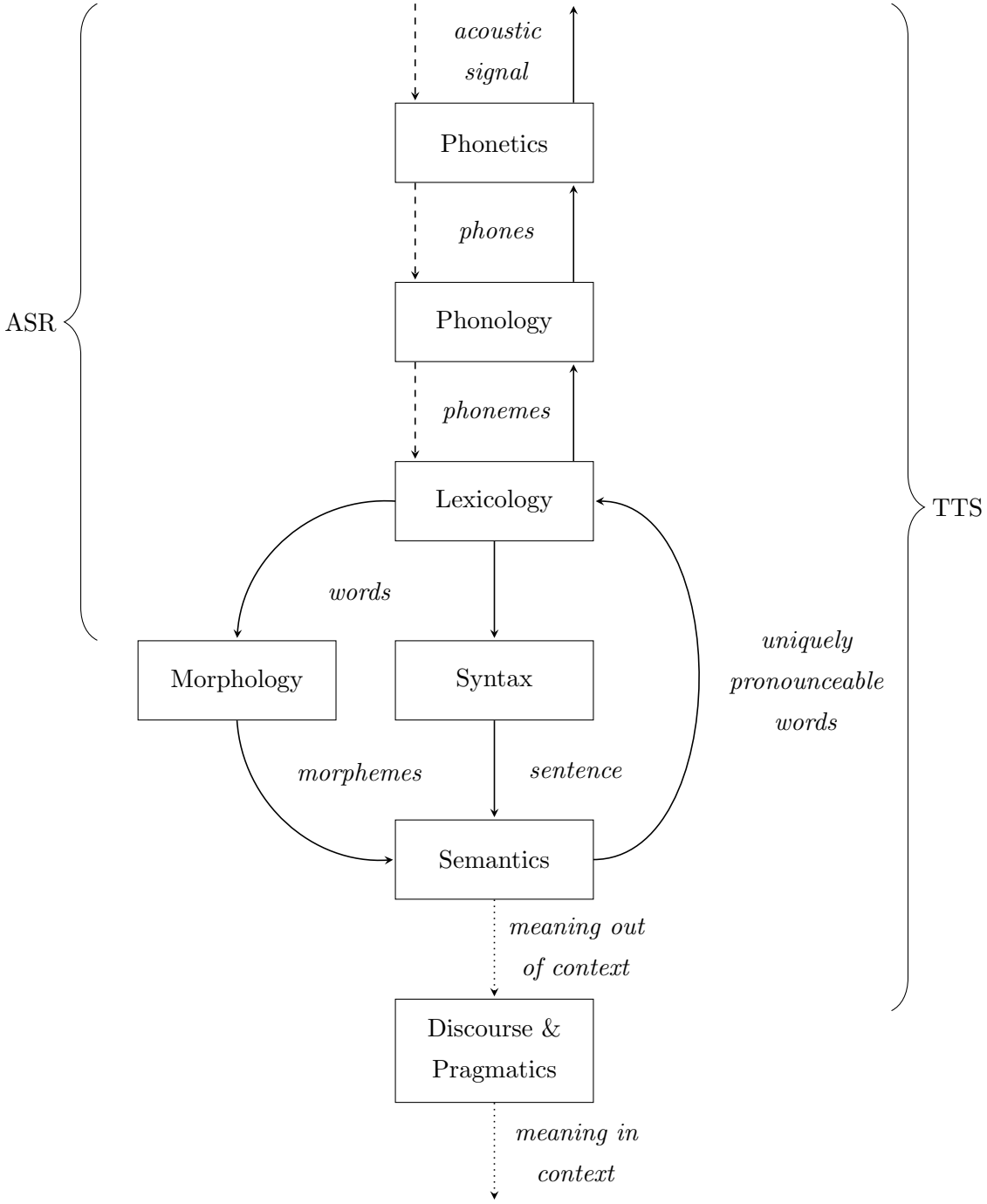


Figure 1.2: Overlap in the fields of NLP, ASR and TTS

1.1.2 Text-to-Speech Synthesis

TTS is the generation of speech from text. It comprises the following stages (adapted from [83]):

Text segmentation The first stage splits the character stream of the text into initial manageable units, namely sentences and tokens. *Sentencisation* is important to limit the scope of subsequent processing, since sentences mostly do not influence the pronunciation of one another. Sentence boundaries are usually marked by punctuation. Conversely, words and their positions in the sentence do influence its pronunciation, therefore *tokenisation* segments a sentence into its constituent tokens, the written forms of the unique words yet to be discovered. Whitespace is the delimiter for many languages.

Text decoding The second stage decodes each token into one or more uniquely pronounceable words. Non-standard word tokens such as numbers, dates and abbreviations are classified and expanded into their standard word natural language counterparts in a process called *normalisation*. Examples of expansions are:

$$\texttt{101} \rightarrow \texttt{one hundred and one} \tag{1.6}$$

$$\texttt{2010/11/19} \rightarrow \texttt{nineteen november twenty ten} \tag{1.7}$$

$$\texttt{etc.} \rightarrow \texttt{et cetera} \tag{1.8}$$

A special case of *homograph disambiguation* then disambiguates homographs[1] among the token expansions that are not homophones[2]. Consider the following:

$$\texttt{bear} \rightarrow \text{BEAR-ANIMAL or BEAR-BURDEN?} \tag{1.9}$$

$$\texttt{bass} \rightarrow \text{BASS-FISH or BASS-MUSIC?} \tag{1.10}$$

`bear` in (1.9) does not need to be disambiguated, but `bass` in (1.10) does. The classification techniques employed in the normalisation and disambiguation processes range from simple regular expression rules to more elaborate context-sensitive rewrite rules, decision lists, decision trees and Naive Bayes classifiers [80, 83, 95].

Text parsing The third stage infers additional lexical, syntactic and morphological structures from the words that are useful for the pronunciation and prosodic modelling stages to follow. The tasks include *POS tagging* (assignment and disambiguation of POS categories to words), *chunking* (parsing of non-overlapping phrases) and *morphological analysis* (identification of stems and affixes in words).

Pronunciation modelling The fourth stage models the pronunciation of individual words. It maps the words to their constituent phonemes, either by looking up known words in a *lexicon* or by applying *grapheme-to-phoneme (G2P) rules* to unknown words. *Syllabification* divides the words into syllables. *Word-level stress* (an inherent property of isolated words: it is stress on certain syllables) or *tone*, depending on the language type, is then assigned to the syllables.

[1] Words with different meanings but the same written form.

[2] Words with the same pronunciation irrespective of their meaning or written form.

Prosodic modelling The fifth stage predicts the prosody of the whole sentence, namely the *phrasing* (pauses between phrases), *sentence-level stress* (a phenomenon of connected speech: certain words in a phrase are stressed according to their word-level stress, at the expense of reducing the word-level stress of the other words; typically content words[3] are stressed and function words[4] are reduced) and *intonation* (the melody or tune of an entire sentence).

Speech synthesis The sixth stage encodes the above information into speech. The various synthesis techniques will be discussed at the end of the section.

In TTS the first five stages of text analysis are collectively referred to as the *NLP frontend*. The sixth stage of speech synthesis is also called the *DSP backend*.

To explain the difference between NLP for TTS and standard NLP, it is necessary to refine what was stated in Section 1.1.1: standard NLP may be viewed as a collection of text processing *modules*, functioning on the different linguistic levels. The modules employ various statistical or rule-based *techniques* to perform their functions. The statistical techniques, in turn, use characteristics or properties extracted from data, called *features*.

NLP for TTS then borrows from the standard NLP collection only those modules, techniques and features that are necessary to affect acceptable pronunciation of words and phrases in the text. For example, discourse and pragmatics are not necessary for TTS. From this perspective, NLP for TTS is a *subset* of standard NLP with *shallower* analyses. However, NLP for TTS also consists of modules, techniques and features that are not part of the standard NLP collection, for example pronunciation and prosodic modelling, because the text is meant for a spoken context. In this way, NLP for TTS is also an *extension* of standard NLP.

The DSP backends of state-of-the-art TTS systems can synthesise the speech according to the major techniques categorised in Figure 1.3 (wherein "gen" stands for the generation technology).

These synthesis techniques may be described briefly as follows [83]:

Articulatory synthesis This parametric method uses an articulatory model of the human vocal tract[5] to simulate the *physical* process of speech production. The control parameters of the model are (inter alia) sub-glottal pressure, vocal fold tension and the relative positions of the different articulatory organs [82].

Formant synthesis The vocal tract has certain major resonant frequencies[6] that change as the configuration of the vocal tract changes. The spectral peaks of the resonances are called *formants* and are the distinguishing frequency components of speech [79]. A formant synthesiser thus aims to simulate the *acoustic* process of speech production in a *source-filter* paradigm—the source models the glottal waveform (a pulse train for voiced sounds and ran-

[3] Words that carry information such as nouns, verbs, adjectives and adverbs.

[4] Words such as pronouns, prepositions, articles and conjunctions.

[5] The vocal tract is the cavity through which a sound wave travels—from the glottis (the space between the two vocal folds in the larynx) through the pharynx (the throat) to the lips and nose [50].

[6] Resonance is the phenomenon of an acoustic system to vibrate at a larger amplitude than normal when driven by a signal which frequency approximates the natural frequency of vibration of the system. Multiple resonant frequencies may be present at the harmonics (frequencies that are an integer multiple) of the natural frequency [3].

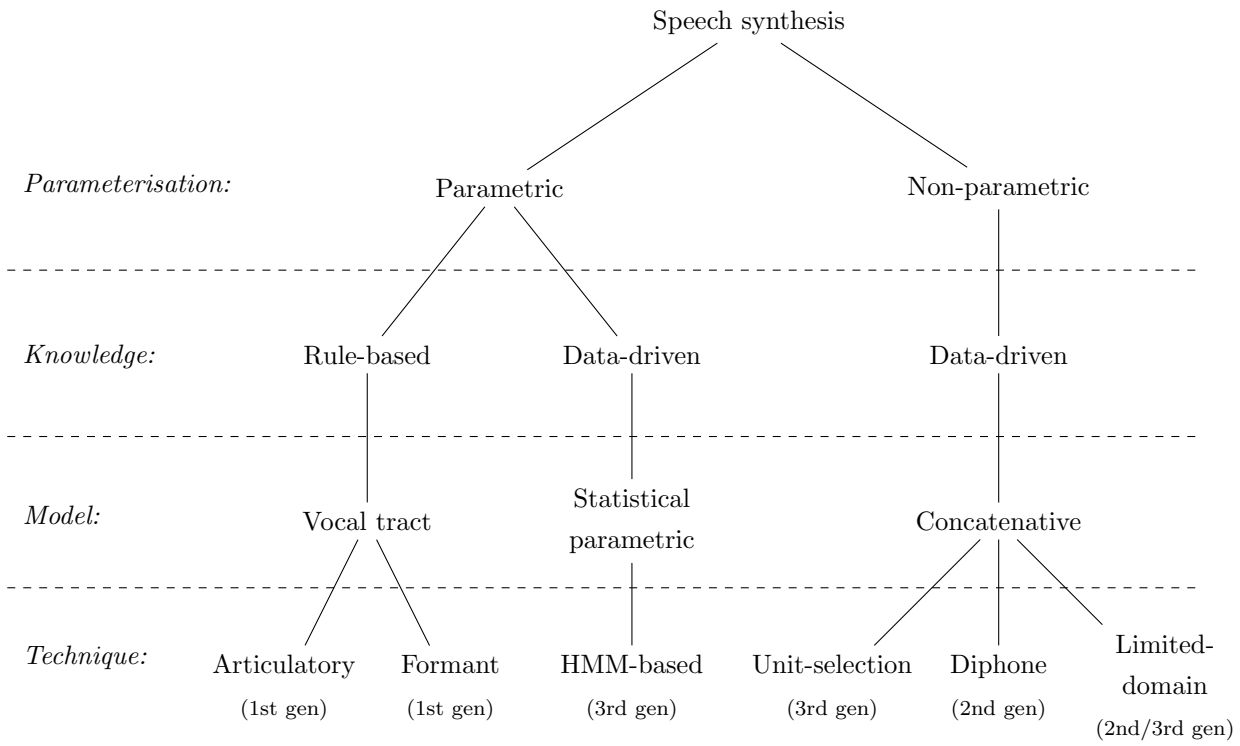


Figure 1.3: State-of-the-art speech synthesis techniques

dom noise for unvoiced sounds) and the filter models the formant resonances of the vocal tract[7] [76].

Diphone synthesis This is a concatenative synthesis technique that moves away from the explicit rule-based models of speech production towards a data-driven approach for generating speech content. An inventory of segmented units of recorded speech is compiled, one instance for each unique type, and concatenated at runtime to synthesise new speech [83]. Intuitively, phoneme waveforms would make sense as the units, but their concatenation poses problems due to coarticulation effects. *Diphone* waveforms, which capture the transition from the middle of one phoneme to the middle of the next, provide a workaround since there is minimal coarticulation at their boundaries [82]. The original diphone units available to the synthesiser will not have the required prosody of the target utterance, so DSP techniques are used to modify the neutral pitch and timing of these diphones to match those of the specification [83].

Unit-selection synthesis Unit-selection is like diphone synthesis but uses a much larger inventory of multiple units per unique type, recorded in different prosodic contexts (not only pitch and timing, but also stress, phrasing, et cetera). At runtime, the synthesiser selects the most appropriate sequences of units that fit the specified prosodic criteria, according to a *target cost*. In this way the prosody is modelled implicitly by the data, meaning that the quality of the synthesis is heavily dependent on the quality and coverage of the unit database. The DSP stage mostly only has to join the units. However, the joining is not that trivial anymore, since

[7] A filter is a system that alters the signal passing through it. This is exactly what the vocal tract does to the sound wave originating at the glottis [27].

the variability in units necessarily results in variability at the unit edges—a consideration that is taken into account in the *concatenation cost* [7, 83].

Limited-domain synthesis For some applications the range of utterances to be synthesised is limited such that it becomes feasible simply to concatenate whole words or phrases from an inventory. When the vocabulary is out-of-range the synthesiser will then fall back on diphone or unit-selection databases. The task is, therefore, to maximise the quality of the most common utterances and have it degrade gracefully to the less common ones [29].

Hidden Markov Model-based synthesis This is an example of statistical parametric synthesis that borrows concepts from both parametric formant and data-driven concatenative synthesis. It uses the source-filter paradigm to model the speech acoustics, but this time the parameters are estimated from the recorded speech instead of being hand-crafted. During training of the system, both *excitation* (inter alia fundamental frequency or *F0*[8]) and *spectrum* (inter alia mel-frequency cepstral coefficients (MFCCs)[9]) parameters are extracted from the data and modelled by context-dependent HMMs. The contexts considered are phonetic, linguistic and prosodic. Furthermore, each HMM has state *duration* probability densities to model the temporal structure of speech. During synthesis, the duration, excitation and spectrum parameters are generated from the concatenated HMMs. The latter two sets of parameters are used in the excitation generation and synthesis filter module to synthesise the speech waveform [7].

TTS voice quality is deemed acceptable according to two performance criteria: *intelligibility* and *naturalness*. Intelligibility measures how understandable the speech is to a listener, that is to which degree the listener will be able to recount the original words in the text. Typical methods employed to evaluate intelligibility include comprehension and transcription tests. Naturalness measures how much the TTS voice sounds like the voice of a human. Methods of evaluation include perceptual tests where a listener rates a single utterance or compares two utterances relatively. The two criteria are in most cases inversely correlated—naturalness is often added to a voice at the expense of its intelligibility—so a balance has to be maintained [83].

1.2 Contextualisation

1.2.1 Part-of-Speech Tagging

A part of speech is a linguistic category assigned to a word in a sentence based upon its morphological and syntactic—or morphosyntactic—behaviour. Words are grouped into POS categories according to the affixes they take (morphological properties) and/or according to their relationship with neighbouring words (syntactic properties) [41]. Example POS categories common to many languages are *noun*, *verb*, *adjective* and *adverb*.

[8] Every periodic signal, such as that which speech approximates, has a fundamental frequency given by the inverse of its period [77]. When considering resonance in speech, the fundamental frequency can be likened to the natural frequency of vibration. Fundamental frequency is a characteristic description of prosody in speech synthesis.

[9] Simplistically, MFCCs are derived in a process that involves the Mel-cepstrum (the spectrum of a non-linearly mapped spectrum) of a signal—they are a good approximation to the response of the human auditory system [28].

Words are often ambiguous in their POS categories, for example `record` can be a noun or a verb. The ambiguity is normally resolved by looking at the context of the word in the sentence, for example in

```
The athlete broke the record for the 100m sprint.
```
(1.11)

`record` can only be a noun.

POS tagging is the automatic assignment and disambiguation of POS categories to words in electronic text. It is a prominent topic in NLP that has been well investigated, since it is a fundamental first step to subsequent syntactic, semantic and other NLP procedures, in applications such as TTS, information retrieval and grammar checking.

Approaches to automatic tagging include rule-based and statistical ones. The former use either hand-crafted rules [31, 35, 43], which require intricate linguistic knowledge, or rules learned from data [12, 40]. The latter are data-driven and use statistical methods, such as Markov models [11] or maximum entropy models [59], to determine the lexical probability (for example, without context, `address` is more likely to be a noun than a verb) and contextual probability (for example, after `to`, `address` is more likely to be a verb). Both approaches to POS tagging are, therefore, very resource-intensive tasks (either in terms of human resources or data resources), and it is a prominent engineering problem in NLP to optimise the use of such resources.

1.2.2 Text-to-Speech Synthesis

The development of TTS voices for resource-scarce languages (RSLs) remains a challenge today. RSLs suffer from the problem of little available electronic data, such as texts and recorded speech, and linguistic expertise, such as phonological and morphosyntactic knowledge. The Lwazi project in South Africa [85] is a large-scale endeavour to gather and apply such human language technology resources for all eleven of the official South African languages. On the TTS front, a multilingual TTS system, called Speect [52], has been developed.

In the first phase of the project, Speect incorporated the following modules in its NLP frontend:

- Whitespace-based tokenisation
- G2P rules
- Syllabification
- Punctuation-based phrase break insertion

The DSP backend was a unit-selection synthesiser. A small speech corpus was recorded with neutral prosody for each language. The neutral prosody compensated for the few examples that would be present per unique type in the unit-selection database. Using just these few resources, baseline intelligible voices for all the languages could be synthesised [47].

One of the TTS goals for phase two of the Lwazi project is to produce more natural voices. Towards this end, Speect is now exploring the HTS engine [86, 97] as HMM-based synthesiser for more and, hopefully, better control over the voices (the parameterisation allows for manipulation).

Furthermore, the speech corpora are going to be larger (albeit still very small compared to those of majority languages) and prosodically richer.

The naturalness of a TTS voice is primarily determined by prosody [26, 62]. Prosody includes phrase breaks, sentence-level stress and intonation [83], and possibly word-level stress or tone as well (see Section 1.1.2). *Central to the modelling of most of the above effects stands POS tagging.* To elaborate:

- Word-level stress is dependent on the POS of the word, for example, in English, nouns often carry stress on different syllables than verbs [61]. This is true for word-level tone as well (which, in addition, requires a morphological analysis for finer grained information, such as tense, on top of the basic POS category [98]).

- Sentence-level stress requires a syntactic structure [83] of which POS information is a building block. Even a simple content-function word rule requires the POS of a word to categorise it.

- Phrase breaks can either be predicted from chunking [83], which in turn requires POS tagging, or directly from the POS tags themselves in an HMM approach to modelling the junctures [84].

- Aspects of intonation, such as the sentence-final pitch of questions, may benefit from identifying, for example, "WH"-words in English through POS tagging.

Solving the POS problem is, therefore, a prudent first step towards meeting the goal of natural TTS voices.

1.3 Problem Statement

The improvement of the naturalness of synthesised speech by means of POS information faces the following challenges in a resource-scarce environment:

1. An automatic tagger is required to obtain the POS information from the text to be synthesised. State-of-the-art taggers are data-driven and require large amounts of labelled data for training. The trend of POS tagging research over the years has been to improve accuracy on these large data sets using different machine learning algorithms and/or features. From an engineering perspective, however, it is more pragmatic to optimise and test the algorithms on less data to cater for RSLs. Recent studies, such as Agrawal and Mani [1] and Hasan *et al.* [38], have started to address this problem, but a comprehensive investigation that compares *all* the tagging algorithms is still outstanding.

2. The pronunciation and prosodic modelling stages require expensive resources in addition to the POS information: stress and tone rules necessitate non-trivial linguistic knowledge. Phrasing based on chunking requires annotated chunk data and the implementation based directly on POS sequences and junctures needs data labelled with the junctures. Once again an engineering solution is required to minimise the resource usage so that a TTS voice for an RSL can still be built efficiently, yet with more naturalness.

1.4 Research Questions

Following from the above statements, these questions may be asked:

1. Which available POS tagging algorithm will be the most accurate given little training data?
2. Is it possible to circumvent the traditional approaches to pronunciation and prosodic modelling by learning the latter directly from the speech data using POS information? In other words, does the addition of POS features to the HTS context labels improve the naturalness of a TTS voice?

1.5 Aims

The research questions lead to the following aims:

1. To find a POS tagging algorithm that exploits little training data the best; and
2. To determine whether POS information has an effect on the naturalness of a TTS voice.

1.6 Hypotheses

The following hypotheses are made in answer to the research questions:

1. The POS tagging algorithm that achieves the highest accuracy in the literature on much data will fare the best on little data as well; and
2. Since the speech data are prosodically rich, and pronunciation and prosody are dependent on POS, the addition of POS features will indeed improve the naturalness of a synthesised voice.

1.7 Contributions

The study hopes to make the following contributions to the natural language engineering community:

1. A "one-stop" review of POS tagging algorithms and their performance on limited data will provide a reference framework for the RSL researcher-developer to perform POS tagging tasks effectively and efficiently; and
2. An alternative way to build more natural synthesised voices more cheaply will stimulate rapid development of TTS systems with limited resources.

1.8 Research Methodology

The modi operandi to reach the aims of this research are:

1. A comparative study of past and state-of-the-art POS tagging algorithms will be done regarding their functioning, resource requirements and accuracy. The algorithms may be categorised according to their resource requirements as *supervised*, *unsupervised* and *semisupervised*. The review will only cover supervised algorithms though. The performance figures of the algorithms, as recorded in the literature, will be discussed. Experiments will be conducted on the supervised taggers using English text from the Penn Treebank WSJ corpus [54] and Afrikaans text from a balanced corpus developed in [56]. Increments of 5,000 labelled tokens up to a maximum of 40,000 tokens will be used as different training data sets. The test data set consists of 10,000 tokens separate from the training data. The original POS tagsets, as well as reduced versions, will be used. The performance measure for the taggers is accuracy, calculated as the percentage correctly tagged tokens out of the total amount of tokens in the test set. A t-test will examine the statistical significance of the tagger accuracies.

2. HTS voices will be trained from English and Afrikaans prosodically rich speech. The speech corpora consist of approximately 1,000 utterances each—for English, the CMU_ARCTIC "US bdl" speaker data are used; for Afrikaans, in-house recordings are used. 100 random utterances from the data are kept aside as a test data set. The voices will then be compared with and without POS features incorporated into the HTS context labels in four experiments. The first experiment tests the POS effects when using a maximal feature set for the context labels. The second experiment repeats the first, but uses a minimal feature set. The third experiment compares voices trained with different quality POS taggers. The final experiment matches the voices using minimal features with POS information against the voices using maximal features without POS information. The analytical measures of absolute difference in duration and mean squared error in pitch and intensity will be used to calculate the closeness of a synthesised utterance to its original natural speech counterpart. One synthesised voice is then considered more natural than another if it is closer to the natural speech. The experiments will include perceptual tests in an attempt to validate the analytical results. 10 mother-tongue speakers from each language will each listen to 20 pairs of synthesised utterances to determine which voice is more natural in a particular experiment. McNemar's test will be used to test the significance of the analytical and perceptual results.

1.9 Overview

Chapter 1 introduced the background content and objectives of the dissertation. The problem statement, research questions, aims, hypotheses, expected contributions and research methodology were presented.

Chapter 2 undertakes the literature study on POS tagging algorithms. The chapter briefly discusses the supervised, unsupervised and semisupervised learning paradigms, and then reviews the supervised algorithms. Each section on a particular algorithm starts off by explaining the theory behind the algorithm and ends off by listing the accuracies obtained in the literature, as well as the resources required to do so.

An experimental investigation of the supervised POS taggers follows in Chapter 3 to test part

one of the research hypothesis. The sections elaborate on the setup—the taggers used, their parameters, the datasets and the tagsets—and tabulate and discuss the results.

Chapter 4 relates the analytical and perceptual experiments on the different TTS voices to assess part two of the hypothesis. The first section explains the setup common to all the subsequent experiments: details of the POS tagger used, the speech datasets, how the voices are built, the HTS context label feature sets, the analytical measures and the perceptual test. The next sections describe each experiment and list and discuss its results.

Finally, the dissertation concludes in Chapter 5. The chapter starts with a summary by briefly restating the problems addressed by the research, how the aims were accomplished by the previous chapters and how the outcomes reflect on the initial hypotheses. The chapter ends with a section on possible extensions to the research that can be addressed by future work.

Chapter 2

A Literature Review of Part-of-Speech Tagging Algorithms

2.1 Introduction

An automatic tagger is required by the TTS system to tag the words with their POS information at synthesis runtime. The task of POS tagging can be accomplished with hand-written rules or statistical approaches, which use data. Over the past two decades, as more data became available, different machine learning algorithms have been applied to POS tagging and have proven to be very successful.

This chapter provides a review of these data-driven algorithms. Literature to date consists mainly of disparate journal articles and conference proceedings, each discussing a single algorithm. Performance figures across the papers are difficult to compare, since different experimental conditions (amount of data, tagsets, language, et cetera) have been used. Furthermore, the focus has mainly been to improve tagging accuracy on large data sets. Only recently has performance on small data sets been investigated.

The purpose of this review is to consolidate the research on the most popular algorithms and present them from a resource-based perspective, which is important for an RSL. Data-driven POS tagging algorithms may be categorised according to the learning paradigm they employ to model the data. They can be *supervised*, *unsupervised* or *semisupervised*, where the term "supervision" refers to the human intervention required for training. The type of learning thus directly determines the type and amount of resources the algorithms require.

Section 2.2 explains these learning paradigms. Supervised learning is briefly introduced before the algorithms in the class are discussed at length in Section 2.3. An in-depth exposition on unsupervised and semisupervised learning is beyond the scope of this dissertation, so these paradigms are only touched upon with literature references. The chapter concludes in Section 2.4 with an interpretation of the supervised results in light of the first hypothesis stated in Section 1.6, namely that the tagging algorithm that fares the best on large data sets will also outperform other algorithms on small data sets.

2.2 Learning Paradigms

2.2.1 Supervised Learning

Supervised algorithms operate on labelled training data; that is each example in the data consists of an input object (for example, an observation) and an output value (for example, a categorisation). In POS tagging, the input is a word from the language and the output its corresponding POS category. Typically, a large collection of continuous text, called a corpus, is gathered and labelled (or annotated) by hand—a very laborous and, therefore, expensive task.

The study of each algorithm in Section 2.3 is based mainly on one representative paper so that the review can cover the topic sufficiently in breadth, rather than focusing in depth, which is beyond the scope of the dissertation. Accuracy figures for English are used throughout as a performance benchmark on large corpora, while figures for selected RSLs are used (where available) to note the effect of little data on a particular algorithm.

It is worth noting at this point that classifiers based on the different algorithms can be combined for an overall (slightly) better performance than what any of the individual classifiers can achieve. The motivation behind it is that the employed machine learning formalisms and/or captured knowledge differ enough to produce different classification errors. In other words, the classifiers have different strengths and weaknesses that can be exploited in combination [91]. The review is about the core algorithms, however, so the assumption will simply be made that classifier combination will produce slightly better results in most cases.

2.2.2 Unsupervised Learning

Unsupervised algorithms infer models from raw, unlabelled data. No supervision is required, making these algorithms a very attractive solution in a resource-scarce environment. However, their accuracy is currently far below that which can be achieved by supervised algorithms.

A popular approach to unsupervised learning of POS categories is the distributional tagging algorithm in [66], which can tag a language for which no knowledge about the categories is available beforehand. It uses the general distributional properties of the text in the training corpus to cluster syntactically similar tokens. The assumption is that the syntactic behaviour of the tokens is reflected in co-occurrence patterns. Therefore, the similarity between two tokens is measured by the degree to which they share the same left and right neighbouring tokens. This approach is explored further in [18] and combined with morphological analysis in [19] and [24].

The above approaches require that the number of clusters, or syntactic categories, be specified. [4] describes a graph-clustering method that infers the kind and number of categories automatically. Two partitions of word cluster graphs are calculated and then merged: one based on distributional similarity of high frequency words and another on log-likelihood similarity of neighbouring co-occurrences of low frequency words. A lexicon is constructed from the resultant word clusters (words mapped to anonymous syntactic categories) and used to train a trigram tagger. The tagger is finally augmented with an affix classifier for unknown words.

2.2.3 Semisupervised Learning

Semisupervised algorithms fall between the supervised and unsupervised paradigms. They use both labelled and unlabelled data. This partial supervision improves accuracy beyond that of unsupervised alternatives, while alleviating some of the burden imposed by manual labelling. The labelled portion of data can come in the form of *seed data*, a small subset of manually labelled data with which to bootstrap the algorithm on the unlabelled data, a tagging *dictionary* of all the words in the lexicon with their possible POS categories (also called ambiguity classes), or a *prototype list* of all the POS categories, each assigned a finite number of word examples from the lexicon.

Seed data approaches include self-training and co-training [20]. In self-training a single tagger is trained once on the seed data and then retrained iteratively on the unlabelled data. In co-training two taggers are retrained iteratively on the output of each other. The idea is that the one tagger can learn useful information from the output of the other if their descriptions of the data are sufficiently different. A variant of these two training approaches is Self-Learned Features (SLF) [57]. It retrains the base supervised model using the predictions from the unlabelled corpus as extra *features* in the model, rather than as new examples to the training corpus. These features are related to the distribution of the classes through the unlabelled corpus.

Dictionary approaches see Hidden Markov Models (HMMs) trained with the Baum-Welch algorithm, an instance of Expectation-Maximisation (EM) that iteratively aligns the observations of ambiguity classes with the states of true POS tags, until the most likely sequences are extracted [21]. A semisupervised version of Transformation-Based Learning (TBL) applies rules to the data that transform the ambiguity classes into single POS tags [14]. Contrastive Estimation (CE) [78] is an alternative training method to EM for HMMs. Parameter estimation may be viewed as pushing probability mass toward the training examples. Whereas EM only considers *to* where the mass is pushed (a positive training example), CE considers *from* where it is taken as well (a neighbourhood set of negative examples). CE moves the mass from the negative to the positive examples under the hypothesis that good models are those that discriminate an observed example from its neighbourhood. Bayesian HMM tagging [34] advocates that, instead of estimating a single set of optimal model parameters such as in EM, a distribution over the latent variables, given the observed data, may be obtained by integrating over all possible model parameter values. The integration makes the model robust in its choices for a tag sequence and allows the use of linguistically appropriate priors. In Bayesian LDA-based tagging [88], the tagging model is an extension of the Latent Dirichlet Allocation (LDA) model [8]. Like the Bayesian HMM, the model employs a sparse prior on the distribution $p(t|w)$ of a tag given a word and holds a distribution over its parameters instead of estimating a single optimal set. It differs, however, by incorporating the prior directly on the $p(t|w)$ distribution.

Given a POS tagset, prototype-driven learning [36] specifies a few word examples, or prototypes, for each tag without going the length of labelling the training corpus. It then links word tokens in the unlabelled training corpus to these prototypes according to their distributional similarity (Section 2.2.2). The prototype links are encoded as features in a log-linear generative model that is trained on the unlabelled data.

2.3 Supervised Tagging Algorithms

2.3.1 Hidden Markov Models [11, 17, 83]

POS tagging assigns the most likely sequence of POS tags $T = \{t_1, t_2, \ldots, t_n\}$ to a sequence of tokens $W = \{w_1, w_2, \ldots, w_n\}$. Probabilistically, this may be expressed as:

$$\hat{T} = \arg\max_T p(T|W) \tag{2.1}$$

Under the Hidden Markov Model (HMM) formalism, W may be viewed as the *observation sequence* and T as the underlying, hidden *state sequence* that produced the observations. Therefore, an HMM is actually a generative model that computes $p(W|T)$. Hence, $p(T|W)$ in (2.1) must be rewritten using Bayes' rule:

$$p(T|W) = \frac{p(W,T)}{p(W)} \tag{2.2}$$

$$= \frac{p(W|T)p(T)}{p(W)} \tag{2.3}$$

The joint probability $p(W,T)$ of (2.2) is the component of the posterior $p(T|W)$ that must be maximised in (2.1), for the evidence $p(W)$ remains constant over the maximisation. $p(W,T)$ may be decomposed as in (2.3) into a likelihood $p(W|T)$, which is the *observation or lexical probabilities* of the sequence of tokens given the sequence of POS tags, and a prior $p(T)$, which is the *state transition or contextual probabilities* of the sequence of POS tags, independent of seeing the tokens:

$$p(W|T) = \prod_{i=1}^{n} p(w_i|t_i) \tag{2.4}$$

$$\begin{aligned} p(T) &= p(t_1, t_2, \ldots, t_n) \\ &= p(t_n|t_{n-1}, t_{n-2}, \ldots, t_1)p(t_{n-1}|t_{n-2}, t_{n-3}, \ldots, t_1)\ldots p(t_2|t_1)p(t_1) \\ &= \prod_{i}^{n} p(t_i|t_{i-1}, t_{i-2}, \ldots, t_1) \\ &\approx \prod_{i=1}^{n} p(t_i|t_{i-1}, t_{i-2}) \end{aligned} \tag{2.5}$$

The prior should model the probability of a POS tag given *all* its predecessors. Yet, the curse of dimensionality does not allow this, so it approximates the probability by using only one to three predecessors—typically two to form a second-order Markov or trigram model [11].

$p(W,T)$ may now be stated succintly as:

$$p(W,T) = \prod_{i=1}^{n} p(w_i|t_i)p(t_i|t_{i-1}, t_{i-2}) \tag{2.6}$$

The argument of the maximisation in (2.1) is obtained through the *Viterbi algorithm*, which optimises $p(W,T)$ over the state sequence T to find the most likely state sequence (path through the HMM) $\hat{T}$ that produced the observation sequence W [58, 83].

The training of the HMM is data-driven. When annotated data is used—that is each token in the corpus is tagged with a POS, so the state sequences that produced the tokens are known—the

state transition and observation probabilities can be computed directly from frequency counts:

$$p(w_i|t_i) = \frac{F(w_i, t_i)}{F(t_i)} \tag{2.7}$$

$$p(t_i|t_{i-2}, t_{i-1}) = \frac{F(t_{i-2}, t_{i-1}, t_i)}{F(t_{i-2}, t_{i-1})} \tag{2.8}$$

where $F(x)$ is the number of times x is observed.

Since the trigrams themselves might also not be estimated reliably due to data sparsity—causing unrealistic low or zero probabilities—a smoothing algorithm can be used to distribute the probability mass to these trigrams. An example is the linear interpolation of unigrams, bigrams and trigrams:

$$p(t_i|t_{i-2}, t_{i-1}) = \lambda_1\hat{p}(t_i) + \lambda_2\hat{p}(t_i|t_{i-1}) + \lambda_3\hat{p}(t_i|t_{i-2}, t_{i-1}) \tag{2.9}$$

where $\hat{p}$ are frequency counts and $\lambda_1 + \lambda_2 + \lambda_3 = 1$ so that p again represents a probability. The values of λ_1, λ_2 and λ_3 are estimated by *deleted interpolation* [11, 33].

An HMM tagger is a statistically sound approach since it uses both a likelihood and prior model, but these models require the training data to be annotated. Furthermore, the larger the POS tagset, the more data are required to prevent sparsity problems.

Brants trained and tested his TnT tagger comprehensively on two corpora in [11]: the German NEGRA corpus of 355,000 tokens and the Penn Treebank Wall Street Journal (WSJ) corpus of approximately 1,200,000 tokens. The NEGRA corpus is tagged with the Stuttgart-Tübingen tagset consisting of 57 tags [64]. The Penn Treebank uses 36 POS tags and 12 other tags for punctuation and currency symbols [54]. Incrementally larger subsets of the corpora were evaluated of which 90% were partitioned for training data and 10% for test data each time. The training and test sets were disjoint.

The accuracy on the NEGRA corpus increased logarithmically from a minimum of 78.1% at 1000 training tokens to a maximum of 96.7% at 320,000 tokens. The accuracy on the Penn Treebank ranged from a minimum of 78.6% at 1000 training tokens to a maximum of 96.7% at 1,200,000 tokens. The learning curve is also logarithmic.

In the NLPAI-ML 2006 contest, Karthik *et al.* [46] and Agrawal and Mani [1] implemented various POS tagging algorithms on corpora of the resource-scarce Indian languages Telugu and Hindi. The Telugu corpus comprised 27,000 tokens for training and 6,000 for testing and the Hindi corpus 21,000 tokens for training and 8,000 for testing. There were 29 tags in the tagset. The HMM tagger of Karthik *et al.* achieved an accuracy of 82.47% on Telugu and the one of Agrawal and Mani 79.64% on Hindi.

Pilon trained the TnT tagger on 20,000 tokens of the Afrikaans corpus developed in [56]. The test set comprised 1,776 tokens. The tagset consists of 139 tags. She obtained an accuracy of 85.87%. In another experiment, using a reduced tagset of 13 tags, the tagger was 93.69% accurate.

Haselbach and Heid developed another Afrikaans tagset in [39]. They argue that the tagset of Pilon is too fine-grained for statistical taggers. Therefore, they have constructed a "slim, yet expressive" set that is "still morpho-syntactically sufficient". This tagset consists of 39 POS tags. They experimented with different taggers on a corpus of 16,636 tokens by using ten-fold cross

validation for training and testing. The TnT tagger achieved a median precision of 97.05%, although this is a bit misleading since they incorporated a lexicon (which eliminates many of the unknown words) into the tagger.

2.3.2 Transformation-Based Learning [12, 13]

Transformation-Based Learning (TBL), or transformation-based *error-driven* learning, works on the following basis as depicted in Figure 2.1 (taken from [13]):

1. Unannotated text, that is text that has not yet been classified, is passed through an *initial-state annotator*. The annotator can range in complexity from one assigning a random classification to one that has been hand-crafted.

2. The output of the annotator, that is the annotated text, is compared to the *truth*, as specified in a manually annotated corpus.

3. A *transformation* list is compiled iteratively from instantiations of transformation templates that are learned from the errors to the truth:

   (a) The highest scoring transformation from the candidate set of instantiations is added orderly to the list.

   (b) The list is then applied in a feedback loop to the initial-state annotator output to make it resemble the truth better until some stopping criteria is met.

4. Once the ordered list of transformations has been learned, new text can be annotated by first applying the initial-state annotator and then each of the transformations in order.

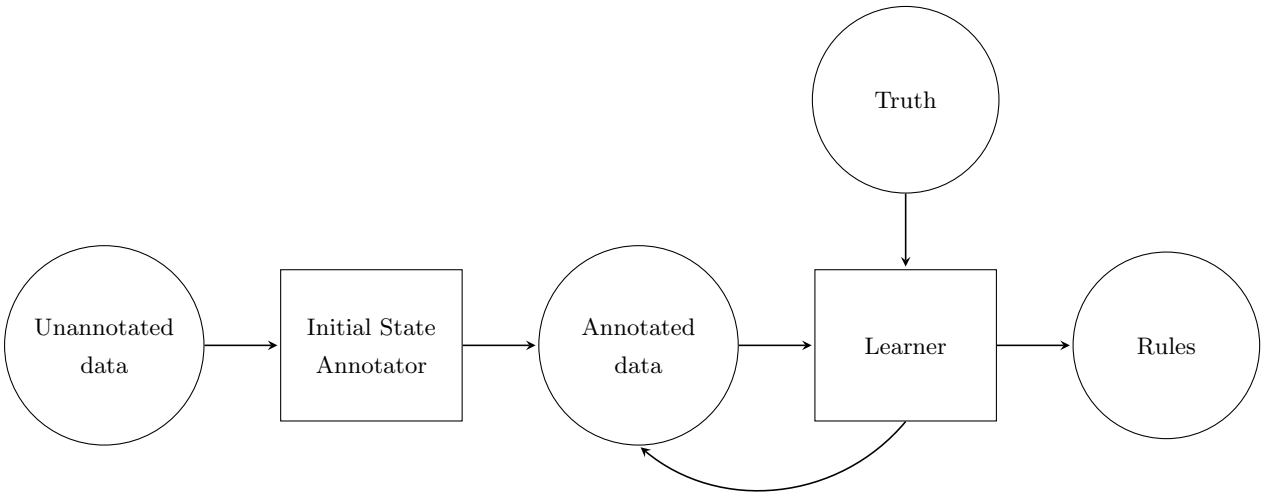


Figure 2.1: TBL process diagram

TBL is applied to POS tagging in the following way:

1. The initial-state annotator assigns each token its most likely POS tag, estimated from a tagged training corpus without regard for context. This makes it a supervised method.

2. The transformation templates are based upon local context around the token, such as the tag of the token(s), or the token(s) itself (themselves), preceding or following the current one. A transformation takes on the form

$$\textit{change tag from } A \textit{ to } B \textit{ if in context } C \tag{2.10}$$

where A and B are single POS tags.

3. The learner learns a transformation from all the permutations of possible instantiations of the templates (where the POS tags in the tagset or the words in the lexicon are inserted into the placeholders of the templates):

   (a) After each instantiation is applied to a subset of the training corpus, the number of tagging errors (as compared to the truth of that subset) is counted.

   (b) The learner adds the instantiation that results in the greatest error reduction to the transformation list, as long as the reduction is above a certain threshold.

4. An example of a learned transformation can be *"change tag from* VERB *to* NOUN *if next tag is* VERB*"*, as in the case for the tag of `running` in the sentence

```
Running is good for your health.
```
(2.11)

As with a supervised HMM tagger, the TBL tagger requires annotated training data of which the required amount scales with the size of the POS tagset (there is more ambiguity to resolve). However, where the HMM formalism can only model the immediate preceding sequential context, the TBL formalism can also model any intermittent local and/or long-distance context.

Brill applied his TBL tagger to the Brown corpus in [12]. The initial-state annotator was trained on 90% of the corpus, the patch data (the "truth") comprised 5% and the test data 5% as well. The tagset was refined to 192 tags. The tagger achieved an accuracy of 94.9% using 71 transformation patches.

Hasan *et al.* evaluated a TBL tagger on the SPSAL 2007 Hindi corpus[1] in a comparative study in [38]. 26,148 tokens made up the training data and 4,924 tokens the test data. There were 26 tags in the tagset. The accuracy was 71.5%.

2.3.3 Tree Tagging [65]

The statistical POS taggers based on Markov models, particularly second-order (trigram) models, have a large number of parameters to be estimated. The typical equation for the estimation of the transition probabilities is:

$$p(t_i|t_{i-2}, t_{i-1}) = \frac{F(t_{i-2}, t_{i-1}, t_i)}{F(t_{i-2}, t_{i-1})} \tag{2.12}$$

When the training data are sparse, the reliable estimation of small transition probabilities becomes problematic. A tree tagger avoids the data sparsity problem by employing a binary

[1] This is most likely the same corpus that was used in the NLPAI-ML 2006 contest. See Section 2.3.1.

decision tree to calculate the estimates of the transition probabilities. The tree automatically determines the appropriate size of the context used in the estimation. The context encompasses the subset of training data, for example trigrams, bigrams or unigrams, at which the tagger arrives at a particular node in the tree, as well as the questions that brought it there, such as $t_{i-2} = det$ and $t_{i-1} = noun$.

The probability of a trigram is obtained by following its decision outcome path down the tree to a leaf node that contains a distribution. Note the decision tree in Figure 2.2. The trigram $p(verb|det, noun)$ will have the probability 0.7.

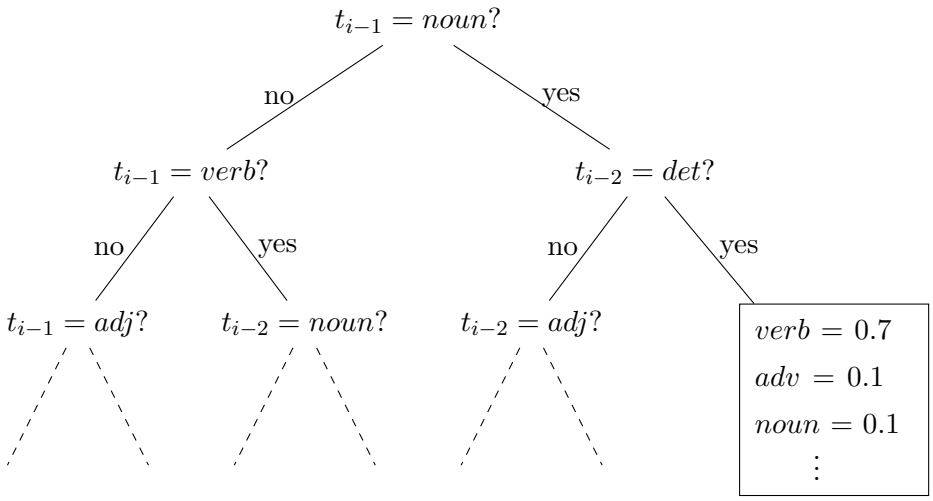


Figure 2.2: A tree tagger decision tree

The decision tree is grown recursively from a training set of trigrams extracted from an annotated corpus. In each recursion step, a test or question splits the set of trigrams at the current node into two subsets that are maximally separated according to the probability distribution of the third tag t_i. A test considers one of the two predecessors t_{i-2} and t_{i-1} and has the form:

$$t_{i-j} = t, \quad j \in \{1, 2\}, \quad t \in T \tag{2.13}$$

where T is the POS tagset.

The instantiation q of (2.13) that yields the greatest gain in information about the third tag when applied to the current node, is the one to split the node. Maximising the information gain is equivalent to minimising the average amount of information I_q that is still needed to identify the third tag after the result of test q is known:

$$I_q = -p(C_+|C) \sum_{t \in T} p(t|C_+) \log_2 p(t|C_+) - p(C_-|C) \sum_{t \in T} p(t|C_-) \log_2 p(t|C_-) \tag{2.14}$$

where C is the context of the current node, C_+ equals C plus the condition that q succeeds and C_- that q fails. $p(C_+|C)$ is the probability that q succeeds and $p(C_-|C)$ that q fails. $p(t|C_+)$ is the probability of the third tag if q succeeded and $p(t|C_-)$ if q failed.

The probabilities in (2.14) are estimated from frequency counts:

$$p(C_+|C) = \frac{F(C_+)}{F(C)} \tag{2.15}$$

$$p(C_-|C) = \frac{F(C_-)}{F(C)} \tag{2.16}$$

$$p(t|C_+) = \frac{F(t, C_+)}{F(C_+)} \tag{2.17}$$

$$p(t|C_-) = \frac{F(t, C_-)}{F(C_-)} \tag{2.18}$$

where $F(C)$ is the number of trigrams in the current context. $F(C_+)$ is the number of trigrams that pass the test; $F(C_-)$ is for those that fail. $F(t, C_+)$ is the number of trigrams that pass the test and which third tag is t; $F(t, C_-)$ is for those that fail.

The tree stops growing when the next test generates at least one subset of trigrams which size is below some threshold. Tag probabilities $p(t|C)$ for the third tag are estimated from all the trigrams in the current context and stored at the current node:

$$p(t|C) = \frac{F(t, C)}{F(C)} \tag{2.19}$$

Schmid experimented with his TreeTagger on the Penn Treebank corpus in [65]. A range from 8,000 to 2,000,000 tokens were used for training and 100,000 tokens for testing. Bigram, trigram and quatrogram contexts were evaluated. At the minimum of 8,000 training tokens, the bigram TreeTagger obtained around 82.5% accuracy and the trigram TreeTagger around 83.5%. At the maximum of 2,000,000 tokens, the bigram tagger obtained 95.78%, the trigram tagger 96.34% and the quatrogram tagger 96.36%. The accuracy increased logarithmically up to these points.

In the experiments on Afrikaans of Haselbach and Heid [39] (see Section 2.3.1 for the setup), the TreeTagger was 96.52% precise. The precision is again very high because the tagging lexicon was used.

2.3.4 Maximum Entropy [44, 59]

Maximum entropy (MaxEnt) modelling advocates the intuition that, if there is no evidence to favour a particular solution to another, both alternatives should be equally likely [44]. This requires as much information as possible about the process to be modelled, namely frequencies of events relevant to, or properties of, the process. These properties place constraints on the model and, from all the models that satisfy them, the one with the flattest distribution—that is with the highest average uncertainty or entropy—is chosen.

For POS tagging, the model defines the random variable h as the history, that is the possible word and tag contexts, of a token and t as the allowable POS tag. The joint probability of h and t is, therefore:

$$p(h, t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)} \tag{2.20}$$

where π is a normalisation constant, $\{\mu, \alpha_1, \ldots, \alpha_k\}$ are the positive model parameters and $\{f_1, \ldots, f_k\}$ are binary features $f_j(h, t) \in \{0, 1\}$.

Given the training data of n tokens $\{w_1, \ldots, w_n\}$ and their tags $\{t_1, \ldots, t_n\}$, let h_i be the history available when predicting the tag t_i for the ith token w_i in the corpus. The goal of the model learning is to maximise the entropy of a distribution, subject to certain constraints.

The entropy of $p(h, t)$ is:

$$H(p) = -\sum_{h,t} p(h, t) \log p(h, t) \tag{2.21}$$

The maximisation constraint is:

$$Ef_j = \tilde{E}f_j, \quad 1 \leq j \leq k \tag{2.22}$$

where Ef_j is the expected value of each feature and $\tilde{E}f_j$ the observed expected value of the feature in the training data ($\tilde{p}$ is an observed probability in the training data):

$$\begin{aligned} Ef_j &= \sum_{h,t} p(h, t) f_j(h, t) \\ &\approx \sum_{i=1}^{n} \tilde{p}(h_i) p(t_i|h_i) f_j(h_i, t_i) \end{aligned} \tag{2.23}$$

$$\tilde{E}f_j = \sum_{i=1}^{n} \tilde{p}(h_i, t_i) f_j(h_i, t_i) \tag{2.24}$$

The model parameters $\{\mu, \alpha_1, \ldots, \alpha_k\}$ are estimated from the above equations using *generalised iterative scaling* [23].

Each parameter α_j is a weight for its corresponding feature f_j and only contributes to $p(h, t)$ in (2.20) when the feature is active, that is when $f_j(h, t) = 1$. Given (h, t), a feature activates on any token or tag in the history, which default context is defined by:

$$h_i = \{w_i, w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}, t_{i-1}, t_{i-2}\} \tag{2.25}$$

MaxEnt tagging thus combines the flexibility of heterogenous contextual features as used by TBL tagging with the probabilistic framework as used by HMM tagging. The features are instantiations of the following default templates, where the variables X, Y and T are automatically filled in from the training data:

$$\begin{aligned} &\langle w_i = X, t_i = T\rangle, \langle w_{i-1} = X, t_i = T\rangle, \langle w_{i-2} = X, t_i = T\rangle, \langle w_{i+1} = X, t_i = T\rangle, \\ &\langle w_{i+2} = X, t_i = T\rangle, \langle t_{i-1} = X, t_i = T\rangle, \langle t_{i-2}t_{i-1} = XY, t_i = T\rangle \end{aligned} \tag{2.26}$$

Once trained, the tagger then tags a new sentence by using a beam search [59] to calculate the N highest probability candidate tag sequences and selecting the one at the top of the list.

Ratnaparkhi applied his MaxEnt tagger on the WSJ corpus of the Penn Treebank in [59]. The corpus had been split into a training data set of 962,687 tokens, a development set of 192,826 (for debugging) and a test set of 133,805. The tagger achieved 96.63% accuracy on the test data.

In the NLPAI-ML 2006 contest, Karthik *et al.* [46] obtained 82.27% on Telugu and Agrawal and Mani [1] 78.96% on Hindi.

2.3.5 Memory-Based Learning [22]

Memory-Based Learning (MBL) is fundamentally a form of k-nearest neighbours modelling. In its application to POS tagging, a set of *cases* or patterns is kept in memory, where each case consists of a token, its left and right contexts and the corresponding POS tag for the token. A new sequence of tokens is tagged by selecting for each token and its context the tags of the most similar cases, or nearest neighbours, in memory.

More formally, MBL is a form of supervised, inductive learning from examples to build a classifier. The examples are from an annotated training data set and each is stored incrementally in memory as a vector of feature values with an associated class label. For the classification of a new feature-value test pattern, its distance to all examples in memory is calculated and the class of the nearest example is assigned to the new pattern.

The distance between two vectors X and Y with values x_i and y_i for feature f_i, respectively, is:

$$\Delta(X, Y) = \sum_{i=1}^{n} G(f_i)\delta(x_i, y_i) \tag{2.27}$$

where $\delta(x_i, y_i)$ is the distance between the two values x_i and y_i

$$\delta(x_i, y_i) = \begin{cases} 0, & x_i = y_i \\ 1, & \text{otherwise} \end{cases} \tag{2.28}$$

and each feature f_i is weighted with its *information gain* $G(f_i)$, the average amount of reduction in the training data set entropy when the value of the feature is known.

MBL is an expensive algorithm, both in space and in time. For each test pattern, all its feature values must be compared against those of all the training patterns. Therefore, it is optimised through the use of *IGTree*, a heuristic approximation to (2.27) that compresses the set of cases into decision trees to store and retrieve the tagger information efficiently at classification time [22].

Daelemans *et al.* conducted experiments with their MBL tagger on the Penn Treebank WSJ corpus in [22]. The one experiment tested the tagger on known words only and the left-context of the token to be tagged was always correctly disambiguated. It used ten-fold cross-validation on several sizes of datasets in increments of 100,000 memory items. Each set would be partitioned ten times into 90% training data and 10% test data. The average accuracy of the ten runs would be taken as the tagger accuracy on that particular set.

The learning curve over the differently sized data sets is more or less logarithmic: the tagger achieved approximately 95.4% (with a standard deviation between 94.7% and 96.0% among the ten-fold cross-validation runs) at the minimum of 100,000 memory items and 96.3% (with a standard deviation between 96.2% and 96.4%) at the maximum of 2,000,000 memory items.

The tagger was also tested under practical circumstances with known and unknown words and a left context disambiguated by the tagger at the previous time instance. Using 2,000,000 tokens as training data, the tagger tagged 96.4% of 200,000 test tokens correctly.

In the NLPAI-ML 2006 contest, Karthik *et al.* [46] achieved 75.75% on Telugu and Agrawal and Mani [1] 80.55% on Hindi.

Haselbach and Heid [39] obtained 78.94% precision with the MBL tagger in their Afrikaans experiments. This figure is much lower than those of the other taggers they employed, because no

lexicon was used.

2.3.6 Sparse Network of Winnows [63]

The Sparse Network of Winnows (linear separators) (SNoW) architecture is a network of threshold gates for classification. The input nodes in the first layer correspond to the token features and the target nodes (the correct values of the classifier) in the second layer each correspond to a distinct POS. There are links from the first to the second layer and these have weights; therefore, each target node is a linear function of the input nodes.

Each target node may be viewed as an autonomous classifying subnetwork, despite all feeding from the same input nodes. A target node does not necessarily have to be connected to all the input nodes, hence the network is sparse. An example is when the input nodes, that is features, were never active with the target node for the same token sequence, or when the link was disconnected during training because the input nodes were not active often enough.

Learning in SNoW is online: every example, or feature vector, is used only once by every target node to refine its definition in terms of the others; it is then discarded. Every labelled example is treated as positive by the target node corresponding to its label and as negative by the others. Hence, each subnetwork is devoted to a single POS tag and learns to separate its tag from the tags of the other subnetworks. The *Winnow algorithm* [51, 63] performs this learning:

- The algorithm has three parameters—a threshold θ and two update parameters, a *promotion* parameter $\alpha > 1$ and a *demotion* parameter $0 < \beta < 1$. Let $A = \{i_1, \ldots, i_m\}$ be the set of active features that are linked to a particular target node and w_i the weight of the link between the ith feature and the target node.

- The algorithm predicts 1 if and only if $\sum_{i \in A} w_i > \theta$ and updates its current hypothesis of the weights only when a mistake is made.

- If the algorithm predicts 0 and the label is 1, the update is a promotion:

$$w_i \leftarrow \alpha w_i, \quad \forall i \in A \tag{2.29}$$

- If the algorithm predicts 1 and the label is 0, the update is a demotion:

$$w_i \leftarrow \beta w_i, \quad \forall i \in A \tag{2.30}$$

To predict the POS tag of a token, the feature vector of the latter activates a subset of the input nodes and the information propagates through all the subnetworks that compete for its classification. The subnetwork that produces the highest activity is the winner and its associated tag is assigned to the token.

Roth and Zelenko tried out their SNoW tagger on the Penn Treebank WSJ corpus in [63]. The training corpus consists of 600,000 tokens and the test corpus of 150,000. The accuracy of the tagger was 97.13%. No results for RSLs could be found.

2.3.7 Conditional Random Fields [48]

Generative models such as HMMs assign a joint probability to paired observation and label sequences. This is done by enumerating over all possible observation sequences, a task for which it is not practical to represent the observations with multiple interacting features and long-distance dependencies, since they make the model inference intractable.

Conditional models, on the other hand, specify the probability of a label sequence, given an observation sequence. This can depend on arbitrary observation features without their distribution having to be modelled. Maximum Entropy Markov Models (MEMMs) are such conditional models in which each state has an exponential model taking the observation features as input and outputs a distribution over possible next states.

MEMMs suffer from the *label bias problem* though: the outgoing state transitions only compete against one another (they are normalised), instead of against all other transitions in the model. This per-state normalisation of transition scores (the probability of the next state given the current state and observation) implies a conservation of score mass whereby all mass that enters a state must be distributed among the possible next states [48]. An observation can influence which next states receive the mass but not how much total mass is passed on. *This causes a bias toward states with fewer outgoing transitions.*

Conditional Random Fields (CRFs) have all the advantages of MEMMs but also solve the label bias problem. Whereas an MEMM uses local exponential models for the conditional probabilities of the next states given the current, a CRF uses a single global exponential model for the joint probability of the entire label sequence, given the observation sequence. Specifically, *chain-structured* CRFs are most suitable for sequences.

Let $\mathbf{X}$ be a random variable over the observation sequence and $\mathbf{Y}$ a random variable over the label sequence. Given a graph $G = (V, E)$ (V being the set of vertices and E the set of edges) such that $\mathbf{Y} = (\mathbf{Y}_v)_{v \in V}$, $(\mathbf{X}, \mathbf{Y})$ is a CRF if $\mathbf{Y}_v$ conditioned on $\mathbf{X}$ satisfies the following Markov property with respect to the graph:

$$p(\mathbf{Y}_v | \mathbf{X}, \mathbf{Y}_w, w \neq v) = p(\mathbf{Y}_v | \mathbf{X}, \mathbf{Y}_w, w \sim v) \tag{2.31}$$

where $w \sim v$ means w and v are neighbours in G.

The cliques of G are the edges and vertices[2], therefore the conditional distribution of $\mathbf{Y}$ given $\mathbf{X}$ is:

$$p_\theta(\mathbf{y}|\mathbf{x}) \propto \exp\left(\sum_{e \in E, k} \lambda_k f_k(e, \mathbf{y}|_e, \mathbf{x}) + \sum_{v \in V, k} \mu_k g_k(v, \mathbf{y}|_v, \mathbf{x}) \right) \tag{2.32}$$

where f_k and g_k are feature functions and $\mathbf{y}|_S$ is the set of components of $\mathbf{y}$ associated with the vertices in subgraph S.

To estimate the model parameters $\theta = (\lambda_1, \lambda_2, \ldots; \mu_1, \mu_2, \ldots)$ from the training data $D =$

[2] A clique in an undirected graph is a subset of its vertices that are all interconnected, that is, for every two vertices in the subset, there exists an edge connecting the two.

$\{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ with empirical distribution $\tilde{p}(\mathbf{x}, \mathbf{y})$, the following log-likelihood must be maximised:

$$\begin{aligned}\mathcal{L}(\theta) &= \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) \\ &\propto \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}, \mathbf{y}) \log p_\theta(\mathbf{y}|\mathbf{x}) \end{aligned} \tag{2.33}$$

An algorithm based on the *improved iterative scaling algorithm* [55] is used for the maximisation [48]. Iterative scaling algorithms update the feature weights as:

$$\lambda_k \leftarrow \lambda_k + \delta\lambda_k \tag{2.34}$$

$$\mu_k \leftarrow \mu_k + \delta\mu_k \tag{2.35}$$

The update $\delta\lambda_k$ for an edge feature f_k is the solution of:

$$\begin{aligned}\tilde{E}[f_k] &= \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}, \mathbf{y}) \sum_{i=1}^{n+1} f_k(e_i, \mathbf{y}|_{e_i}, \mathbf{x}) \\ &= \sum_{\mathbf{x},\mathbf{y}} \tilde{p}(\mathbf{x}) p(\mathbf{y}|\mathbf{x}) \sum_{i=1}^{n+1} f_k(e_i, \mathbf{y}|_{e_i}, \mathbf{x}) e^{\delta\lambda_k T(\mathbf{x},\mathbf{y})} \end{aligned} \tag{2.36}$$

where $T(\mathbf{x}, \mathbf{y})$ is the total feature count:

$$T(\mathbf{x}, \mathbf{y}) = \sum_{i,k} f_k(e_i, \mathbf{y}|_{e_i}, \mathbf{x}) + \sum_{i,k} g_k(v_i, \mathbf{y}|_{v_i}, \mathbf{x}) \tag{2.37}$$

e_i is the edge with labels $(\mathbf{Y}_{i-1}, \mathbf{Y}_i)$ and v_i is the vertex with label $\mathbf{Y}_i$. The update $\delta\mu_k$ for a vertex feature has a similar form.

Lafferty, McCallum and Pereira applied their CRF tagger to the Penn Treebank WSJ corpus in [48]. The corpus was split 50-50 into training and test data sets of 550,000 tokens each. They achieved an accuracy of 95.73%. An HMM performed at 94.31% on the same data and an MEMM at 95.19%.

In the NLPAI-ML 2006 contest, Karthik *et al.* [46] obtained 75.11% on Telugu with their CRF and Agrawal and Mani [1] 82.67% on Hindi.

Singh and Bandyopadhyay performed CRF tagging on a 63,200-token corpus of the Indian language Manipuri in [75]. The corpus was divided into three parts: a training, development and test set of 39,121, 15,000 and 8,672 tokens each, respectively. They were annotated with 26 different POS tags. The accuracy on the test set was 72.04%.

2.3.8 Cyclic Dependency Networks [89]

When building probabilistic models for POS tag sequences under the graphical model framework, the global probability of the sequence is decomposed using a directed graphical model into the product of the local portions of the model. Most of these models, such as HMMs (Section 2.3.1) and Conditional Markov Models (CMMs) (Figure 2.3), take a unidirectional approach to conditioning inference along the sequence (usually left-to-right), even though the identity of a tag naturally depends on the identities of the tags preceding *and* following it. For example, the joint probability

of a tagged token sequence (T, W) in the CMM of Figure 2.3 is, according to the chain rule of probability:

$$p(T, W) = \prod_i p(t_i|t_{i-1}, w_i) \tag{2.38}$$

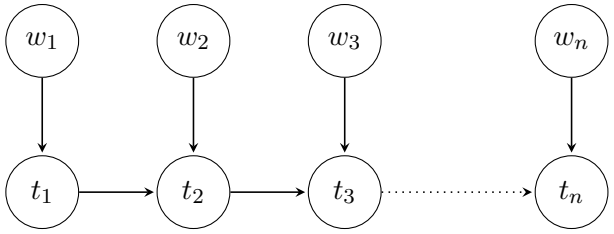


Figure 2.3: Left-to-right Conditional Markov Model

Hence, the unidirectional approach only considers one direction of influence explicitly: the prediction of the current tag t_i is based on the previous tag t_{i-1} (and the current token w_i). The interaction in the other direction between t_i and t_{i+1} is modelled implicitly when t_{i+1} is predicted.

The cyclic dependency network tagger models both directions explicitly using a bidirectional (and hence cyclic) dependency network, as in Figure 2.4. According to the graphical model framework, the local model probability is:

$$p(t_i|t_{i-1}, t_{i+1}, w_i) \tag{2.39}$$

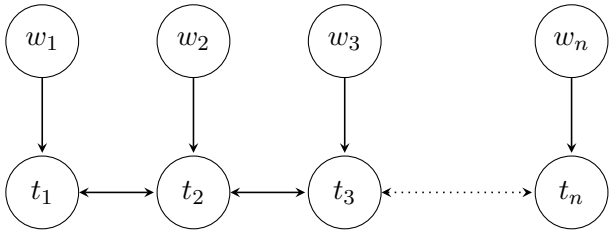


Figure 2.4: Bidirectional dependency network

The chain rule of probability does not allow the joint probability $p(T, W)$ to be reconstructed by multiplying the local conditional probabilities. Nevertheless, a score can still be calculated for the tag sequence (which would be the joint probability if the network were acyclic, as in Figure 2.3):

$$score(T, W) = \prod_i p(t_i|t_{i-1}, t_{i+1}, w_i) \tag{2.40}$$

An adaptation of the Viterbi algorithm can then be used to find the sequence that maximises the score.

The local conditional probabilities in (2.39) can be estimated from sufficient annotated training data with any maximum likelihood method, even relative frequency counts. However, for this tagger local maximum entropy models with feature templates for the conditioned variables in (2.39) are used (compare Section 2.3.4). A *conjugate-gradient* procedure [67] maximises the data likelihood.

As an extension, the maximum entropy models also incorporate lexical feature templates of the current token and surrounding tokens in addition to the bidirectional tagging feature templates.

These expressive templates lead to a large number of features that pose the danger of the models being overfitted. This is avoided by incorporating a Gaussian prior (also known as quadratic regularisation) into the models that penalises high feature weights unless they produce great score gain.

The final feature set of the tagger is the following:

$$\begin{aligned}&\langle t_i, t_{i-1}\rangle, \langle t_i, t_{i-1}, t_{i-2}\rangle, \langle t_i, t_{i-1}, t_{i+1}\rangle, \langle t_i, t_{i+1}\rangle, \langle t_i, t_{i+1}, t_{i+2}\rangle, \langle t_i, w_i\rangle,\\ &\langle t_i, w_{i-1}\rangle, \langle t_i, w_{i+1}\rangle, \langle t_i, w_i, t_{i-1}\rangle, \langle t_i, w_i, t_{i+1}\rangle, \langle t_i, w_i, w_{i-1}\rangle, \langle t_i, w_i, w_{i+1}\rangle\end{aligned} \tag{2.41}$$

The cyclic dependency network tagger was implemented by Toutanova *et al.* and tested on the Penn Treebank WSJ corpus in [89]. The training set comprises sections 0 to 18 of the corpus at 912,344 tokens. The test set comprises sections 22 to 24 at 129,654 tokens. The tagger obtained an accuracy of 97.24%. No results could be found for an RSL.

2.3.9 Support Vector Machines [6, 32]

Let $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a training set of n D-dimensional example vectors with class labels $\{t_1, \ldots, t_n\}$, where $t_i \in \{-1, +1\}$. A Support Vector Machine (SVM) is a binary classifier that learns a linear hyperplane separating the positive training examples from the negative examples with maximal margin. The margin is the distance between the hyperplane and the nearest positive and negative examples.

The separating hyperplane is characterised by a weight vector $\mathbf{w}$, with one component for each feature, and a bias b, which represents the distance of the hyperplane from the origin. The SVM classifies a new example with $sgn\,[y(\mathbf{x})]$, that is according to the sign of $y(\mathbf{x})$:

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x}) + b \tag{2.42}$$

where $\phi(\mathbf{x})$ is a mapping of $\mathbf{x}$ into a high dimensional feature space. This allows the hyperplane to be linear in the feature space, yet nonlinear in the original input space.

When the training examples are linearly separable in the feature space, the learning of the maximal margin hyperplane is a *quadratic programming* problem:

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.43}$$

subject to

$$t_i y(\mathbf{x}_i) \geq 1, \quad i = 1..n \tag{2.44}$$

However, when the training examples are not linearly separable, an exact separation will cause overfitting and not generalise well. Hence, some examples must be allowed to be misclassified, that is lie on the wrong side of the hyperplane. This is done with *slack variables* $\xi_i \geq 0, \quad i = 1..n$, one for each example in the training set. Examples for which $\xi_i = 0$ are correctly classified and lie either on the margin or on the correct side of the margin. Examples for which $0 < \xi_i \leq 1$ lie inside the margin but on the correct side of the hyperplane. Examples for which $\xi_i > 1$ lie on the wrong side of the hyperplane and are misclassified.

The quadratic optimisation now becomes:

$$\arg\min_{\mathbf{w},b,\xi,C} \left[C \sum_{i=1}^{n} \xi_i + \frac{1}{2}\|\mathbf{w}\|^2 \right] \tag{2.45}$$

subject to

$$t_i y(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1..n \tag{2.46}$$

$$C > 0 \tag{2.47}$$

$$\xi_i \geq 0, \quad i = 1..n \tag{2.48}$$

where C controls the trade-off between the slack variable penalty and the margin [6].

POS tagging is a multiclass classification problem that needs to be binarised for an SVM. A *"one-versus-rest"* scheme may be employed: an SVM is trained for every POS class in order to distinguish between its examples and those of all the rest. At classification time, the most confident tag according to the predictions of all the individual SVMs is selected.

In order to avoid the generation of excessive and irrelevant negative examples, not all training examples are considered for all the classes. A dictionary is compiled from the training corpus with the ambiguity classes (possible valid POS categories) for each token. When processing a particular token w_i and its corresponding tag t_i, that example is allocated as a positive example for class t_i and as a negative example only for the rest of the ambiguity classes of the token.

Finally, the feature set for the SVM approach is:

- Lexical values of the surrounding tokens
- Contextual values of the surrounding tags
- Token bigrams and trigrams
- Tag bigrams and trigrams

Giménez and Màrquez ran experiments on the Penn Treebank WSJ corpus in [32]. The one experiment used a closed vocabulary (all tokens were known words) and differently sized training sets. The test set consisted of 266,695 tokens. The accuracy of the SVM tagger ranged from 96.27% at the minimum of 50,000 training tokens to 97.56% at the maximum of 635,000 tokens. The learning curve is logarithmic.

In the experiment that used an open vocabulary (known and unknown words) and a training set of 635,138 and the same test set of 266,695 tokens, the accuracy was 96.9%.

Singh and Bandyopadhyay also ran an SVM tagger on the Manipuri corpus in [75] (see Section 2.3.7 for the experimental setup). The tagger was 74.38% accurate.

2.4 Conclusion

Table 2.1 summarises the results obtained by the various supervised taggers on large training corpora of English. From these figures it is evident that the selection of taggers performs well on large amounts of data. All obtain accuracies in the 96–97% range with the exception of the TBL

and CRF taggers. The slightly lower scores may be attributed to a much larger tagset (an increase in ambiguity) in the case of the TBL tagger and a smaller training data set (less examples) in the case of the CRF tagger. The top scorer is the cyclic dependency network of Toutanova *et al.* [89]

Table 2.1: Best scores for supervised POS tagging on large English training corpora

Algorithm	**Literature**	**Corpus**	**#Tags**	**#Tokens**	**Acc (%)**
HMM	Brants [11]	WSJ	48	1,200,000	96.70
TBL	Brill [12]	Brown	192	900,000	94.90
TreeTagger	Schmid [65]	WSJ	48	2,000,000	96.34
MaxEnt	Ratnaparkhi [59]	WSJ	48	962,687	96.63
MBL	Daelemans *et al.* [22]	WSJ	48	2,000,000	96.40
SNoW	Roth & Zelenko [63]	WSJ	48	600,000	97.13
CRF	Lafferty *et al.* [48]	WSJ	48	550,000	95.73
Cyclic depnet	Toutanova *et al.* [89]	WSJ	48	912,344	97.24
SVM	Giménez & Màrquez [32]	WSJ	48	635,138	96.90

An important observation about these data-driven taggers is their logarithmic behaviour on the amount of training data. Their accuracies increase proportionally much more significantly at the lower end of the data scale than at the higher end. In other words, as the training data grows, more and more additional data are required to improve the accuracies by the same percentage point (see, for example, the sections on the HMM, tree tagging and MBL algorithms). The reason is that the implicit lexicon of the training data becomes "saturated": the closed classes of words[3], which contribute greatly to accuracy, are soon sufficiently represented and modelled. What remains to improve accuracy, are the open classes of words[4], but, because of their vast membership, they require much more examples to become sufficiently represented.

Some of the taggers do climb the logarithmic curve more steeply (quickly) than others when noting what the amount of training data is for similar accuracies. The Tree and MBL taggers require 2,000,000 tokens to achieve their 96–97% accuracies, while the HMM, MaxEnt and cyclic dependency network taggers require only around 1,000,000 tokens. The SNoW and SVM taggers are the most efficient in obtaining 96–97% by only using around 600,000 tokens. It would be interesting to see what percentage gain might be achieved from using extra data up to the 2,000,000 mark.

Nevertheless, such small increases in accuracy do not justify the extra data in a resource-scarce context. Towards the tagging of RSL text, it is, therefore, not only prudent to find the most accurate tagger, but also the point on the data-accuracy graph where the data are being used optimally by the tagger. The literature study at this point can give at best an indication of what the best tagger may be. Table 2.2 summarises the accuracies of the tagging algorithms on the small training corpora of selected RSLs. The Afrikaans results of Haselbach and Heid [39] are not listed since they are precision figures, not accuracy ones. Furthermore, their use of a lexicon as an additional resource is not duplicated by and thus comparable to the other taggers.

The performance of the taggers is notably worse on the smaller data sets. The top scorer across

[3] Certain POS categories, such as prepositions and conjunctions, have a finite number of members over time.

[4] Other POS categories, such as nouns and verbs, have an infinite number of members over time.

Table 2.2: Best scores for supervised POS tagging on small training corpora

Algorithm	Literature	Corpus	#Tags	#Tokens	Acc (%)
Afrikaans					
HMM	Pilon [56]	NWU	139	20,000	85.87
Hindi					
HMM	Agrawal & Mani [1]	NLPAI-ML 2006	29	21,000	79.64
TBL	Hasan *et al.* [38]	SPSAL 2007	26	26,148	71.50
MaxEnt	Agrawal & Mani [1]	NLPAI-ML 2006	29	21,000	78.96
MBL	Agrawal & Mani [1]	NLPAI-ML 2006	29	21,000	80.55
CRF	Agrawal & Mani [1]	NLPAI-ML 2006	29	21,000	82.67
Telugu					
HMM	Karthik *et al.* [46]	NLPAI-ML 2006	29	27,000	82.47
MaxEnt	Karthik *et al.* [46]	NLPAI-ML 2006	29	27,000	82.27
MBL	Karthik *et al.* [46]	NLPAI-ML 2006	29	27,000	75.75
CRF	Karthik *et al.* [46]	NLPAI-ML 2006	29	27,000	75.11
Manipuri					
CRF	Singh & Band. [75]	Custom	26	39,121	72.04
SVM	Singh & Band. [75]	Custom	26	39,121	74.38

the languages is the HMM-based TnT tagger used by Pilon [56] on Afrikaans. The accuracy of HMM taggers on the Indian languages is significantly lower, probably because the custom implementations do not have as good unknown word handling as that for which TnT is renowned. The HMM and MaxEnt taggers are less accurate than the MBL and CRF taggers on Hindi but more accurate on Telugu. The inconsistency is difficult to explain without investigating the relation between these two Indian languages. What is known though, is that their grammar and morphology are derived from a common ancestor, Sanskrit [2]. If they are then similar, perhaps the discrepencies lie in the corpus content. Manipuri appears to be a difficult language to model, since the taggers perform worse even though the corpus size is larger.

The research hypothesis in Section 1.6 states that the POS tagging algorithm that performs the best on much data will do likewise on little data. However, no literature has been found wherein the large corpora winner, the cyclic dependency network algorithm, has been applied to small corpora. The best algorithm among those that are present, is the HMM.

The inconclusive results of the literature study necessitate actual experimentation with the different supervised POS tagging algorithms to note their performance on little data. The fact that a controlled environment can then be used for the experiments—the same language, amount of training data and tagset size—will give a clearer representation of the taggers than what the literature study could provide in Tables 2.1 and 2.2. It will also be possible to note the points of optimality in data use, since the accuracies on different subsets of the training data will be trackable. Chapter 3 relates this experimental investigation.

Chapter 3

Part-of-Speech Tagging in a Resource-Scarce Environment

3.1 Introduction

In the literature review of the previous chapter it was seen that the popular supervised POS tagging algorithms are all competitive on large English corpora, with many achieving accuracies in the 96–97% range. The performance on small corpora is much lower and less consistent at a range of 72–85%, but this picture is not an accurate representation of the algorithms. The figures of some algorithms are missing, particularly those of the cyclic dependency network tagger, which performs best on the large corpora. Furthermore, different RSLs are used for different taggers.

The first experimental investigation, therefore, revisits the research question of which available POS tagging algorithm will be the most accurate given little training data. Those algorithm implementations on the internet that are the most popular and freely available are tested. To obtain conclusive results, the experimental environment is controlled. The language, amount of data and tagset remain constant across the different taggers. English features again as a reference language, even though it is strictly not an RSL; its resources are simply restricted to emulate scarceness. In addition, another Germanic language, Afrikaans, is used as a true RSL. The taggers are tested on incremental subsets of the training data so that the optimal points of data use may be identified. The experiments employ comprehensive and reduced POS tagsets for both languages; the reduced tagsets are used to note the effect on the tagger accuracies.

Section 3.2 thus relates the experiments using the comprehensive POS tagsets and Section 3.3 the experiments using the reduced tagsets. Each section elaborates on the details of the taggers, data and tagsets in the setup and lists and discusses the results. Section 3.4 concludes the chapter by evaluating the results against the research hypothesis made on POS tagging.

3.2 Experiment 1: Comparing Supervised Taggers

The purpose of the first experiment is to find the best supervised POS algorithm among those studied in Section 2.3 by comparing implementations obtained from the internet on English and Afrikaans.

3.2.1 Setup

3.2.1.1 Data and Tagsets

The English POS tagset is the Penn Treebank set of 48 tags (46 effectively occur in the data) [54]. The Afrikaans tagset is larger at 139 effective tags, as described in [56]. These tagsets will be dubbed **eng.wsj1** and **afr.nwu1**, respectively, and may be viewed in Appendix A.

For both languages 49,900-token annotated data sets are available. The English set is the first portion of the WSJ corpus and the Afrikaans set is the balanced corpus developed in [56]. The upper 9,900 are used as a completely separate test data set. The lower 40,000 are divided into 5,000-token incremental training data subsets. The upper 5,000 of the 40,000-token training subset double as a development set for algorithm parameter optimisation.

Table 3.1 lists the number of known, ambiguous and unknown words in the 9,900-token test set, for each training subset. Known words are words in the test set that also occur in the training set. The ambiguous words are those *known* words that have two or more possible POS tags. The unknown words are words in the test set that do not occur in the training set; they are naturally all ambiguous.

Table 3.1: Statistics of the test data using the full tagsets

Training data	**Test data**		
	Known	**Ambiguous**	**Unknown**
eng.wsj1			
5,000	6,531	544	3,369
10,000	7,210	1,302	2,690
15,000	7,520	2,061	2,380
20,000	7,741	2,189	2,159
25,000	7,875	2,325	2,025
30,000	7,998	2,451	1,902
35,000	8,072	2,563	1,828
40,000	8,264	2,616	1,636
afr.nwu1			
5,000	6,798	2,679	3,102
10,000	7,361	3,127	2,539
15,000	7,664	3,294	2,236
20,000	7,849	3,462	2,051
25,000	8,495	3,981	1,405
30,000	8,611	4,441	1,289
35,000	8,686	4,525	1,214
40,000	8,829	4,562	1,071

3.2.1.2 Taggers

The POS taggers used in this experiment are listed below. They are linked to the case studies presented in Section 2.3 by noting the publications in which they were used. In this way their performance can also be looked up in Tables 2.1 and 2.2.

The configurations are chosen such that tagging accuracy is the primary objective and speed a secondary objective. The reason for the inclusion of the latter is that fast tagging speed is an important requirement in the TTS system pipeline. Nevertheless, not all of the parameters are stringently optimised; only the most important ones are with crude grid searches. The goal is not to obtain perfect accuracy figures on the two languages alone, but ball-park ones to be able to choose a sufficiently accurate tagger suitable for the general TTS case. The reader is referred to the documentation that comes with the software packages for explanations of the syntax of the parameters and feature sets.

Hidden Markov Model tagger (HMM) Hunpos is an open source reimplementation of Brants' TnT tagger [11], available at `http://code.google.com/p/hunpos/`. It may be used commercially. The accuracy of Hunpos on English is on par with TnT as noted in [37]. Therefore, the work of Pilon [56], Karthik *et al.* [46] and Agrawal and Mani [1]—all using TnT—can also serve as an indication of the performance of Hunpos on RSLs.

The parameter configuration is:
`-t 2 -e 1 -f 3 -s 6`
Optimisation runs shows that deviation from these produces negligible differences in accuracy.

Transformation-based tagger (TBL) The implementation of Brill's tagger used in [12] is available at `http://www.tech.plym.ac.uk/soc/staff/guidbugm/software/RULE_BASED_TAGGER_V.1.14.tar.Z`. It is open source, but the licence does not allow redistribution for profit. Hasan *et al.* [38] used the NLTK [5] implementation of the TBL tagger; it is unclear whether this version is exactly based on the original tagger.

The tagger has no significant parameters to tune, except that the source code has to be modified at some places to accommodate different tagsets.

Tree tagger (Tree) Schmid's decision tree-based HMM tagger [65] can be downloaded at `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`. It is closed source (only the binaries are available) and may only be used for research and educational purposes.

The default parameters are found to be optimal.

Maximum entropy tagger (MaxEnt) Ratnaparkhi makes a closed source, non-commercial version of the tagger he used in [59] available at `ftp://ftp.cis.upenn.edu/pub/adwait/jmx/`. The MaxEnt taggers that Karthik *et al.* [46] and Agrawal and Mani [1] used are not this original version.

The tagger has no documented significant parameters to optimise.

Memory-based tagger (MBL) Tilburg University distributes a tagger, MBT, based upon their TiMBL software, under the GNU General Public Licence. MBT and TiMBL may be down-

loaded at `http://ilk.uvt.nl/mbt` and `http://ilk.uvt.nl/timbl`, respectively. MBT is the tagger used by Daelemans *et al.* in [22], as well as by Karthik *et al.* in [46] and Agrawal and Mani in [1].

The optimal configuration of the significant parameters is:
`-p dwdwfWaw -P dwdwFawpssschn -O"-a IGTREE -m 0 -w 1"`
that includes larger, more expressive feature template sets (`-p` and `-P`). The fast IGTREE classification algorithm is used to optimise speed.

SNoW tagger (SNoW) The University of Illinois maintains a tagger based on the SNoW architecture at `http://l2r.cs.uiuc.edu/~cogcomp/download.php?key=FLBJPOS`. It uses their Learning Based Java framework that can be downloaded at `http://l2r.cs.uiuc.edu/~cogcomp/download.php?key=LBJ`. It is open source and may be used commercially. No publications have used this tagger implementation, yet it appears to be the only one available.

The only adjustable parameter is the separator thickness and this has a negligible effect. However, the source code has to be modified to work effectively with different tagsets.

Conditional random fields tagger (CRF) CRF++ is an open source implementation (GNU Lesser General Public Licence and Berkeley Software Distribution Licence) of CRFs that can perform POS tagging. It is available at `http://crfpp.sourceforge.net/`. CRF++ is not the implementation used by Lafferty *et al.* in [48], but is the one used by Agrawal and Mani in [1] and Singh and Bandyopadhyay in [75].

The default parameters suffice, except that the data-fitting hyperparameter is optimised to `-c 3` and a less expressive feature set needs to be used:

```
# Unigram
U00:%x[-2,0]
U01:%x[-1,0]
U02:%x[0,0]
U03:%x[1,0]
U04:%x[2,0]

# Bigram
B
```

Cyclic dependency network tagger (Depnet) Toutanova *et al.*'s bidirectional tagger [89] can be downloaded at `http://nlp.stanford.edu/software/tagger.shtml`. It uses the GNU General Public Licence.

The model architecture is:
`arch = bidirectional,naacl2003unknowns,wordshapes(-1,1)`
that is based upon the one used in the original paper and includes the rich feature set as described in Section 2.3.8. Deviation from the default value of the conjugate gradient learning parameter has negligible influence.

Support vector machine tagger (SVM) SVMTool is an open source SVM tagger available at `http://www.lsi.upc.edu/~nlp/SVMTool/` under the GNU Lesser General Public Licence.

It uses the SVM^{light} software package at `http://svmlight.joachims.org/`, but this requires permission from its author to use commercially. Giménez and Màrquez used SVMTool in [32], but Singh and Bandyopadhyay used another tagger in [75].

The basic left-to-right model `do M0 LR` is used so as not to incur speed costs. The C parameters for the known and unknown word SVM classifiers are optimised to `CK = 0.11` and `CU = 0.09`, although their effects do not differ significantly from those of the defaults. The known word feature set is:

```
A0k = C(0;-2) C(0;-1) C(0;0) C(0;1) C(0;2) C(0;-2,-1) C(0;-1,0)
      C(0;0,1) C(0;-1,1) C(0;1,2) C(0;-2,-1,0) C(0;-2,-1,1)
      C(0;-1,0,1) C(0;-1,1,2) C(0;0,1,2) C(1;-2) C(1;-1) C(1;-2,-1)
      C(1;-1,1) C(1;1,2) C(1;-2,-1,1) C(1;-1,1,2) k(0) k(1) k(2)
      m(0) m(1) m(2)
```

and the unknown word feature set:

```
A0u = C(0;-2) C(0;-1) C(0;0) C(0;1) C(0;2) C(0;-2,-1) C(0;-1,0)
      C(0;0,1) C(0;-1,1) C(0;1,2) C(0;-2,-1,0) C(0;-2,-1,1)
      C(0;-1,0,1) C(0;-1,1,2) C(0;0,1,2) C(1;-2) C(1;-1) C(1;-2,-1)
      C(1;-1,1) C(1;1,2) C(1;-2,-1,1) C(1;-1,1,2) k(0) k(1) k(2)
      m(0) m(1) m(2) a(2) a(3) a(4) z(2) z(3) z(4) ca(1) cz(1)
      L SA AA SN CA CAA CP CC CN MW
```

3.2.1.3 Statistical Significance

The task of determining the accuracy of a tagger on a test data set may be viewed as a sequence of N independent Bernoulli trials, where N is the number of tokens in the test set. A Bernoulli trial is an experiment with two possible outcomes: *success* and *failure*, or in the POS tagging case *correctly tagged* and *incorrectly tagged* [10]. Let p be the probability of a success in a single Bernoulli trial; $1-p$ is then the probability of failure. The probability of k successes in N Bernoulli trials is described by the binomial distribution $B(N,p)$:

$$P(k) = \binom{N}{k} p^k (1-p)^{N-k} \tag{3.1}$$

The observed accuracy of the tagger is an estimation of p (converted to the range $[0,1]$), such that the estimated mean and variance of $B(N,p)$ may be calculated as:

$$\mu_N = Np \tag{3.2}$$

$$\sigma_N^2 = Np(1-p) \tag{3.3}$$

An unpaired two-sample t-test can then be used to determine the statistical significance between the accuracies of two taggers or, more technically, between the means of their distributions [10, 60]. The null hypothesis states that the means are equal. To disprove the null hypothesis, the first step is to calculate the t-value from the means and variances of the two distributions:

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \tag{3.4}$$

t has a distribution with $N_1 + N_2 - 2$ degrees of freedom that approaches the normal distribution, especially if $N > 30$.

The second step is to set the significance level α. A probability greater or equal to $\alpha = 0.05$ is generally considered to be significant. The t-test can be one- or two-tailed: if the alternative hypothesis states the expected direction of the results—that tagger 1 will be more accurate than tagger 2 (or vice-versa)—then t will fall into one tail of the normal distribution and α is used as is. If the alternative hypothesis does not assume any direction—it only states that there is some difference between the taggers—then t can fall into either tail of the distribution and $\frac{\alpha}{2}$ must be used. Since no assumption can be made about the tagger accuracies beforehand, $\frac{\alpha}{2}$ is used.

The final step compares t to the appropriate value in a t-test table (Table B.1 in Appendix B) by lining up $\frac{\alpha}{2} = 0.025$ with $N_1 + N_2 - 2 = 9,900 + 9,900 - 2 = 19,798 \approx \infty$ degrees of freedom (the number 9,900 is the test set size). This gives a value of 1.960. If $t \geq 1.960$ the null hypothesis is disproved, meaning that the accuracies of the two taggers are significantly different. If $t < 1.960$ the null hypothesis is proved and the two taggers do not differ significantly.

3.2.2 Results

Table 3.2 lists the accuracies obtained by the taggers. Not surprisingly, for both eng.wsj1 and afr.nwu1 the accuracies increase consistently as the training data increase. The taggers fare better on the eng.wsj1 data mainly because the afr.nwu1 tagset is much larger than the one of eng.wsj1, and hence there is more ambiguity to resolve (compare the ambiguous word counts in Table 3.1).

Figure 3.1 presents a graphic illustration of the results. From this it is evident that the accuracies are starting to converge (the rate of change is flattening out) from 20,000 training tokens for eng.wsj1, indicating that the data window with which was experimented, is near optimal for the practical trade-off between accuracy and data requirements. In other words, additional data will have a much less significant effect on the accuracy than the data used in the window. Therefore, the expense is not justified in a resource-scarce environment. However, the window is not yet optimal for afr.nwu1: more data should counter the effect of the ambiguity caused by the large tagset. It is also noted that the Afrikaans data increment of 20,000–25,000 tokens is particularly rich in contributing to the accuracies and, should it have been used elsewhere, the curve would look slightly different. Nevertheless, this apparent non-trivial effect of the choice of increments does become smaller at the end of the data size spectrum.

The two top-scoring taggers according to Table 3.2 for each tagset are the HMM and SVM taggers, with little differences between them (the two highest accuracies are highlighted in bold). They consistently perform the best across the training data ranges, with the exception of one data set in each of the tagset experiments for the SVM tagger. *The HMM tagger is the preferred candidate, however, since its tagging speed is orders of magnitude faster.* It tagged the 9,900-token eng.wsj1 test set in 0.187s and the afr.nwu1 test set in 0.210s, whereas the SVM tagger took 8.838s and 12.600s, respectively (the difference in speed has also been noted in [32]).

The statistical significance of the results is given in Table 3.3. The left column compares the difference in accuracies between the HMM and SVM taggers, using (3.2), (3.3) and (3.4) for $N = 9,900$. All differences but the one at 5,000 tokens for eng.wsj1 are insignificant, confirming that

Table 3.2: Supervised tagging: overall accuracies using the full tagsets

Tokens	Tagger Accuracy (%)								
	HMM	**TBL**	**Tree**	**MaxEnt**	**MBL**	**SNoW**	**CRF**	**Depnet**	**SVM**
eng.wsj1									
5,000	**88.97**	85.81	84.64	85.07	84.91	81.99	71.00	**86.31**	86.15
10,000	**91.62**	90.07	87.72	89.03	89.57	86.50	77.70	89.24	**91.60**
15,000	**92.76**	91.67	89.26	91.44	90.81	88.09	81.29	90.73	**92.85**
20,000	**93.25**	92.11	90.06	92.26	91.71	89.84	83.32	91.39	**93.56**
25,000	**93.39**	91.71	90.66	92.73	92.25	90.23	84.27	91.66	**93.79**
30,000	**93.55**	92.90	91.05	92.94	92.28	90.79	85.20	91.80	**94.10**
35,000	**93.83**	93.53	91.23	93.21	92.43	91.08	86.18	92.05	**94.35**
40,000	**94.24**	93.55	91.49	93.65	92.90	91.63	87.16	92.48	**94.61**
afr.nwu1									
5,000	**79.32**	74.94	76.57	72.61	73.42	60.69	65.45	78.00	**78.82**
10,000	**82.70**	79.82	79.27	77.54	79.01	65.82	70.71	81.74	**82.52**
15,000	**83.85**	81.07	81.10	79.62	80.85	68.54	73.02	83.61	**84.00**
20,000	**84.79**	81.95	82.65	81.11	81.57	70.08	75.01	**84.77**	84.70
25,000	**87.75**	84.13	85.54	83.63	85.16	76.76	80.72	86.36	**87.20**
30,000	**88.67**	85.63	86.32	84.39	85.97	78.15	82.07	87.32	**88.14**
35,000	**89.42**	86.19	86.58	85.30	86.42	75.93	82.80	88.13	**88.68**
40,000	**90.26**	86.91	87.49	86.21	87.40	77.25	84.02	89.09	**89.54**

these two taggers are equally accurate. The right column compares the preferred HMM tagger to the next-best one (excluding the SVM tagger). Here it is seen that the HMM tagger is significantly more accurate roughly half of the time.

Table 3.4 lists the accuracies for known, ambiguous and unknown words on eng.wsj1. Performance is relatively consistent on known and ambiguous words across all taggers and all data subsets (with the exception of the MaxEnt tagger on the smaller data sets and the CRF tagger throughout). On unknown words the behaviour varies considerably, with the HMM, MaxEnt and SVM taggers outperforming the rest by a large margin. Regarding the low-performing CRF tagger, aside from the low accuracy on unknown words (which indicates no special unknown word handling such as affix analysis), is the low accuracy on known words relative to the accuracy on the subset of ambiguous known words. Consider the scores at 20,000 training tokens. All of the taggers are 91–92% accurate on ambiguous known words, but the CRF tagger is at 89% for the total amount of known words, whereas the others are around 95%. This means that the CRF tagger tags more unambiguous known words incorrectly than the other taggers. This trend is seen throughout the rest of the data sets of eng.wsj1.

Table 3.5 lists the word category scores for afr.nwu1. The same observations as on eng.wsj1 apply here; the absolute accuracies are just lower because of the larger tagset. The unknown word accuracies are proportionally much lower though. This is probably indicative of the unknown word handling procedures being optimised for English. The unknown word scores of 0.00% for the SNoW

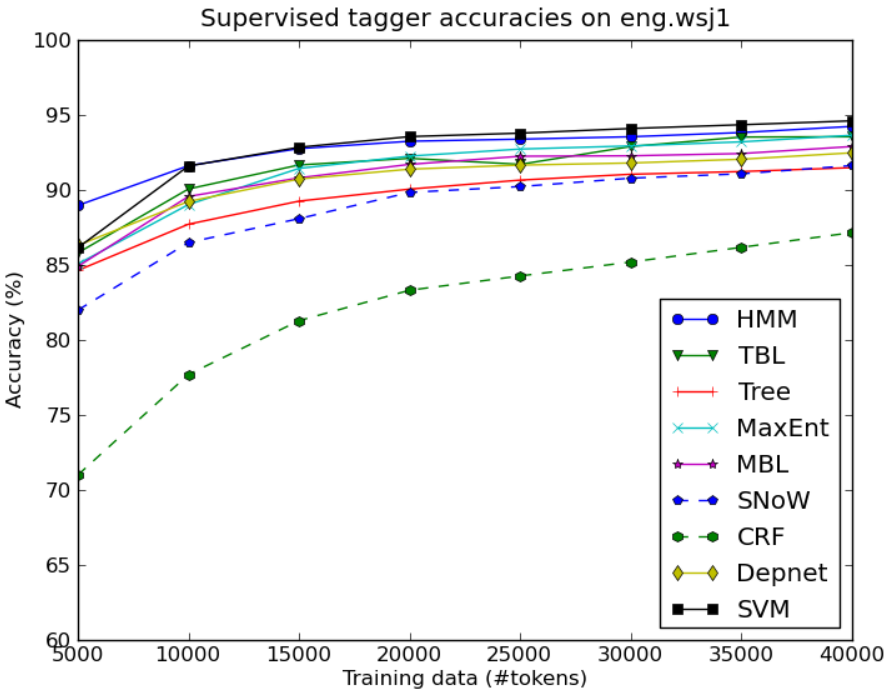
Table 3.3: Supervised tagging: statistical significance using the full tagsets

Tokens	**Statistical Significance**					
	HMM	**SVM**	t	**HMM**	**Next**	t
eng.wsj1						
5,000	88.97	86.15	6.017	88.97	86.31	5.691
10,000	91.62	91.60	0.051	91.62	90.07	3.783
15,000	92.76	92.85	0.245	92.76	91.67	2.863
20,000	93.25	93.56	0.879	93.25	92.26	2.687
25,000	93.39	93.79	1.149	93.39	92.73	1.827
30,000	93.55	94.10	1.608	93.55	92.94	1.710
35,000	93.83	94.35	1.552	93.83	93.53	0.867
40,000	94.24	94.61	1.135	94.24	93.65	1.741
afr.nwu1						
5,000	79.32	78.82	0.865	79.32	78.00	2.267
10,000	82.70	82.52	0.334	82.70	81.74	1.767
15,000	83.85	84.00	0.287	83.85	83.61	0.457
20,000	84.79	84.70	0.176	84.79	84.77	0.039
25,000	87.75	87.20	1.169	87.75	86.36	2.914
30,000	88.67	88.14	1.165	88.67	87.32	2.923
35,000	89.42	88.68	1.667	89.42	88.13	2.876
40,000	90.26	89.54	1.681	90.26	89.09	2.706

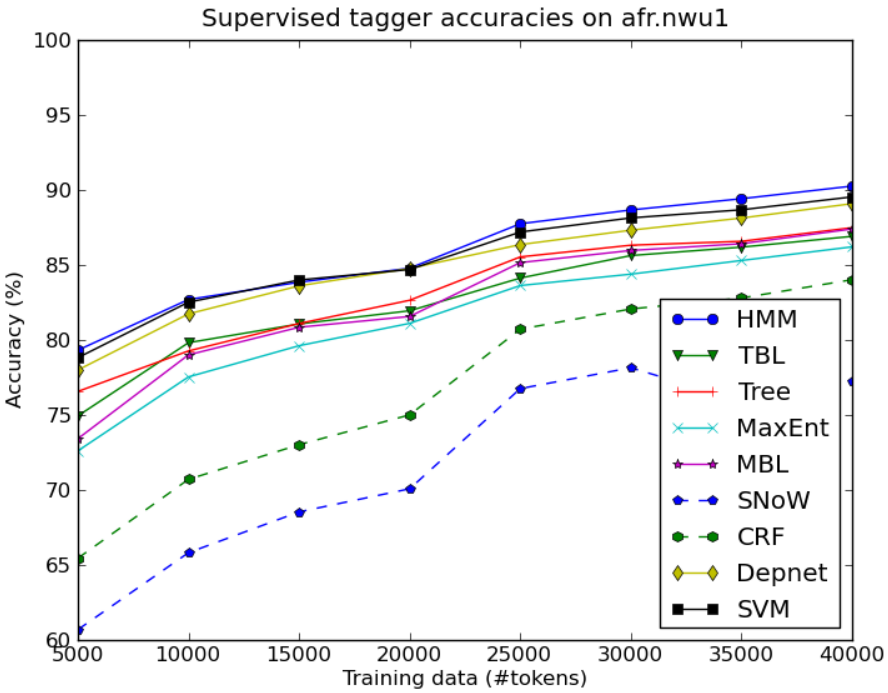
tagger on afr.nwu1 explain the unusually big dip in performance compared to the other taggers when going from eng.wsj1 to afr.nwu1. The unknown word classifier component had somehow broken during training for Afrikaans, even though care had been taken to modify the source code to accommodate the different tagsets (Section 3.2.1.2). Since the SNoW tagger is in any case not among the best taggers for English, trying to solve the problem for Afrikaans is not justified.

The POS tagging algorithms obtain much better results on English (Table 3.2) than on the Indian languages in literature (Table 2.2) for comparable training data amounts (20,000 versus 21–39,000 tokens) and tagset sizes (46 versus 26–29 tags). These differences can probably be ascribed to the Indian languages being more (morphologically) complex than English [45]. The HMM score for Afrikaans at 20,000 tokens is slightly lower than the score obtained by Pilon in [56] for the same corpus and tagset (refer to Section 2.3.1). The reason is that her setup deviates slightly: she uses the closed source TnT tagger and a much smaller test set.

In summary, this experiment has delivered more concrete comparative results for all the tagging algorithms because of the controlled environment. The HMM tagger proves to be the most effective and efficient tagger on little data for both English and Afrikaans. The optimal point of data use has been found to be around 20,000 tokens for English using the eng.wsj1 tagset. The point is beyond the tested 40,000-token mark for Afrikaans because of the detrimental effect of the ambiguity in the large afr.nwu1 tagset size. The question may thus be asked what the performance figures will be if smaller tagsets are used. The next section explores this idea.



(a)



(b)

Figure 3.1: Supervised tagging: overall accuracies using the full tagsets

Table 3.4: Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **eng.wsj1** (using the full tagset)

Tokens	**Tagger Accuracy (%)**								
	HMM	**TBL**	**Tree**	**MaxEnt**	**MBL**	**SNoW**	**CRF**	**Depnet**	**SVM**
5,000									
K	95.41	95.31	95.31	89.42	95.27	95.15	82.85	95.13	95.22
A	84.38	83.27	83.27	77.94	82.72	81.25	73.16	81.07	84.01
U	76.49	67.38	63.94	76.64	64.83	56.49	48.03	69.22	68.57
10,000									
K	95.88	95.59	95.52	92.48	95.40	95.38	86.53	95.56	95.53
A	90.94	89.32	89.02	87.10	88.25	88.17	85.25	89.17	89.17
U	80.19	75.28	66.80	79.78	73.94	62.68	54.01	72.30	81.04
15,000									
K	95.89	95.40	95.56	94.02	95.36	95.07	88.66	95.70	95.55
A	93.74	91.95	92.29	91.85	91.80	90.73	90.88	93.06	92.77
U	82.86	79.87	69.37	83.28	76.43	66.05	58.03	75.00	84.33
20,000									
K	95.71	95.41	95.56	94.37	95.30	95.27	89.59	95.56	95.54
A	92.78	91.73	92.10	91.91	91.32	91.23	91.00	92.23	92.37
U	84.44	80.27	70.36	84.72	78.83	70.36	60.86	76.47	86.43
25,000									
K	95.82	94.82	95.56	94.81	95.39	95.40	90.04	95.53	95.71
A	92.73	89.33	91.70	91.78	91.27	91.31	90.54	91.74	92.39
U	83.95	79.60	71.60	84.64	80.05	70.12	61.83	76.59	86.32
30,000									
K	95.75	95.46	95.44	94.90	95.35	95.37	90.61	95.47	95.76
A	93.06	92.13	91.88	92.57	91.76	91.84	91.43	92.17	93.31
U	84.28	82.12	72.61	84.70	79.39	71.50	62.46	76.34	87.12
35,000									
K	95.84	95.83	95.55	95.01	95.29	95.58	91.44	95.49	95.94
A	93.17	93.13	92.12	92.31	91.53	92.35	91.88	92.08	93.48
U	84.96	83.37	72.16	85.28	79.76	71.23	62.96	76.86	87.36
40,000									
K	96.03	95.91	95.63	95.26	95.28	95.58	91.72	95.61	95.93
A	93.73	93.35	92.24	93.16	91.40	92.32	92.13	92.39	93.46
U	85.21	81.60	70.54	85.51	80.87	71.64	64.12	76.65	87.90

Table 3.5: Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **afr.nwu1** (using the full tagset)

Tokens	**Tagger Accuracy (%)**								
	HMM	**TBL**	**Tree**	**MaxEnt**	**MBL**	**SNoW**	**CRF**	**Depnet**	**SVM**
5,000									
K	90.38	89.75	89.56	82.79	89.64	88.38	80.51	91.90	90.11
A	85.33	83.73	83.31	81.75	83.46	83.69	80.07	87.46	84.66
U	55.06	42.49	48.10	50.29	37.85	0.00	32.43	50.40	54.06
10,000									
K	90.29	89.97	88.83	84.51	89.28	88.52	82.42	91.00	89.65
A	84.75	84.01	81.45	82.09	82.38	83.53	81.20	85.67	83.27
U	60.69	50.37	51.56	57.31	49.23	0.00	36.75	57.05	61.84
15,000									
K	90.16	89.39	89.33	85.59	89.37	88.53	83.32	91.30	90.06
A	84.34	82.54	82.67	83.94	82.48	83.33	81.97	86.43	84.09
U	62.21	52.55	52.91	59.12	51.65	0.00	37.70	59.02	63.24
20,000									
K	90.32	89.13	89.74	86.33	89.02	88.39	84.09	91.02	89.91
A	84.66	81.98	83.74	83.68	81.72	82.96	82.93	86.06	83.74
U	63.63	54.46	55.49	61.14	53.05	0.00	40.27	61.52	64.75
25,000									
K	91.68	89.49	90.48	87.38	90.57	89.45	86.82	90.79	90.99
A	86.81	82.14	84.40	84.28	84.45	84.38	85.10	84.58	85.38
U	63.99	51.74	55.66	60.93	52.46	0.00	43.84	58.76	64.27
30,000									
K	92.24	90.19	90.95	87.77	91.02	89.85	87.68	91.50	91.51
A	88.99	85.00	86.53	85.70	86.62	86.42	86.94	87.16	87.59
U	64.78	55.16	55.39	61.75	52.21	0.00	44.61	57.28	65.63
35,000									
K	92.64	90.05	90.86	88.38	91.05	86.54	88.12	91.72	91.72
A	89.41	84.44	86.01	86.10	86.36	79.73	86.87	87.01	87.67
U	66.31	58.57	55.93	63.26	53.21	0.00	44.73	59.83	66.89
40,000									
K	93.12	90.51	91.20	89.04	91.56	86.62	88.84	92.37	92.21
A	90.03	84.96	86.32	86.65	87.00	79.46	87.44	87.93	88.25
U	66.67	57.24	56.86	62.93	53.03	0.00	44.26	60.38	67.51

3.3 Experiment 2: Reducing the Tagsets

In this experiment the tagsets of the two languages are reduced to note the effect on the tagger accuracies. A reduction may be justified by the fact that, for TTS purposes, fine-grained POS factors such as subcategorisation (number, gender) often become redundant. For example, English word-level stress depend only on the distinction among the broad POS categories such as noun, verb, adjective, adverb and preposition [61]. Sentence-level stress from a content-function word perspective and phrasing based on chunking require very few categories as well. It has also been shown that phrasing based directly on POS sequences benefits from reduced tagsets [84]. Finally, "WH" words for intonation can have a single category and the tags of punctuation and other non-alphabetic symbols can typically be collapsed without detrimental effects.

3.3.1 Setup

3.3.1.1 Data and Tagsets

The reduced English POS tagset **eng.wsj2** consists of 18 tags and the reduced Afrikaans tagset **afr.nwu2** consists of 17 tags (see Appendix A). It is noted that Afrikaans has another reduced set of 39 tags developed by Haselbach and Heid [39], but it is probably still unnecessarily verbose for TTS. The training and test data are relabelled and the new word distributions are listed in Table 3.6. The ambiguous word counts are now less because the tagsets are smaller.

Table 3.6: Statistics of the test data using the reduced tagsets

Training data	**Test data**		
	Known	**Ambiguous**	**Unknown**
eng.wsj2			
5,000	6,531	467	3,369
10,000	7,210	1,179	2,690
15,000	7,520	1,899	2,380
20,000	7,741	2,000	2,159
25,000	7,875	2,114	2,025
30,000	7,998	2,227	1,902
35,000	8,072	2,271	1,828
40,000	8,264	2,308	1,636
afr.nwu2			
5,000	6,798	2,277	3,102
10,000	7,361	2,536	2,539
15,000	7,664	2,659	2,236
20,000	7,849	2,754	2,051
25,000	8,495	3,038	1,405
30,000	8,611	3,517	1,289
35,000	8,686	3,596	1,214
40,000	8,829	3,616	1,071

3.3.1.2 Taggers

The same taggers as in Section 3.2 are used. Optimisation results did not necessitate parameter configurations different from the full tagset counterparts.

3.3.1.3 Statistical Significance

The same criteria as in Section 3.2 are used to test for statistical significance.

3.3.2 Results

Table 3.7 lists the overall results for the reduced tagsets eng.wsj2 and afr.nwu2. The absolute accuracies of all of the taggers are higher than those on the full tagsets because the degree of ambiguity in the data is reduced by the smaller tagsets. Furthermore, since the afr.nwu2 tagset is basically equal in size to the eng.wsj2 one, their accuracy figures differ significantly less. The remaining dissimilarities can be explained by the interesting observation in Table 3.6: despite the almost equal tagsets of the two languages, Afrikaans still has significantly more ambiguous words than English per data subset.

Table 3.7: Supervised tagging: overall accuracies using the reduced tagsets

Tokens	**Tagger Accuracy (%)**								
	HMM	**TBL**	**Tree**	**MaxEnt**	**MBL**	**SNoW**	**CRF**	**Depnet**	**SVM**
eng.wsj2									
5,000	**90.95**	87.99	89.08	87.67	87.52	86.17	82.50	88.77	**91.55**
10,000	**93.27**	91.84	91.02	91.32	90.52	89.54	85.94	91.28	**93.54**
15,000	**94.62**	93.10	92.54	93.06	92.40	90.64	88.05	92.56	**94.33**
20,000	**94.95**	93.07	92.97	93.45	93.58	91.66	89.20	93.06	**95.01**
25,000	**95.20**	94.22	93.04	94.07	94.16	92.40	89.90	93.26	**95.02**
30,000	**95.46**	94.61	93.36	94.52	94.45	92.52	90.37	93.46	**95.31**
35,000	**95.59**	94.71	93.71	94.58	95.03	92.85	91.16	93.73	**95.52**
40,000	**95.90**	95.10	94.04	95.03	95.21	93.35	91.85	94.07	**95.85**
afr.nwu2									
5,000	**87.95**	84.76	86.90	84.32	83.20	64.25	80.17	87.13	**88.64**
10,000	**90.52**	87.99	89.12	87.65	86.95	70.07	83.65	89.08	**90.84**
15,000	**91.31**	87.50	90.24	88.66	88.17	72.37	84.42	90.46	**91.79**
20,000	**91.87**	89.03	90.85	89.74	89.00	74.31	85.67	91.19	**92.08**
25,000	**93.51**	91.22	92.47	91.55	91.58	80.91	89.40	93.15	**93.72**
30,000	**93.98**	92.08	92.90	91.90	92.28	81.39	90.27	93.58	**94.52**
35,000	**94.16**	92.43	93.38	92.46	92.48	79.17	90.71	93.80	**94.83**
40,000	**94.64**	92.49	93.75	93.13	93.26	80.74	91.61	94.33	**95.17**

Figure 3.2 illustrates the same convergence for eng.wsj2 from 20,000 training tokens onwards. Now, however, the same sign is beginning to show from 30,000 tokens for afr.nwu2. The latter

optimal data point is "delayed" with respect to the former because of the larger ambiguous word count noted earlier.

The relative performance of the taggers remain the same as for the full tagsets: the HMM and SVM taggers are still the top-scorers. Table 3.8 gives the statistical significance of the results. Again, the left column shows that the differences between the HMM and SVM taggers are insignificant (all but in the one case at 35,000 tokens for afr.nwu2). The right column shows that the preferred HMM tagger is significantly more accurate than the next-best one (excluding the SVM tagger) for eng.wsj2, but mostly insignificantly for afr.nwu2.

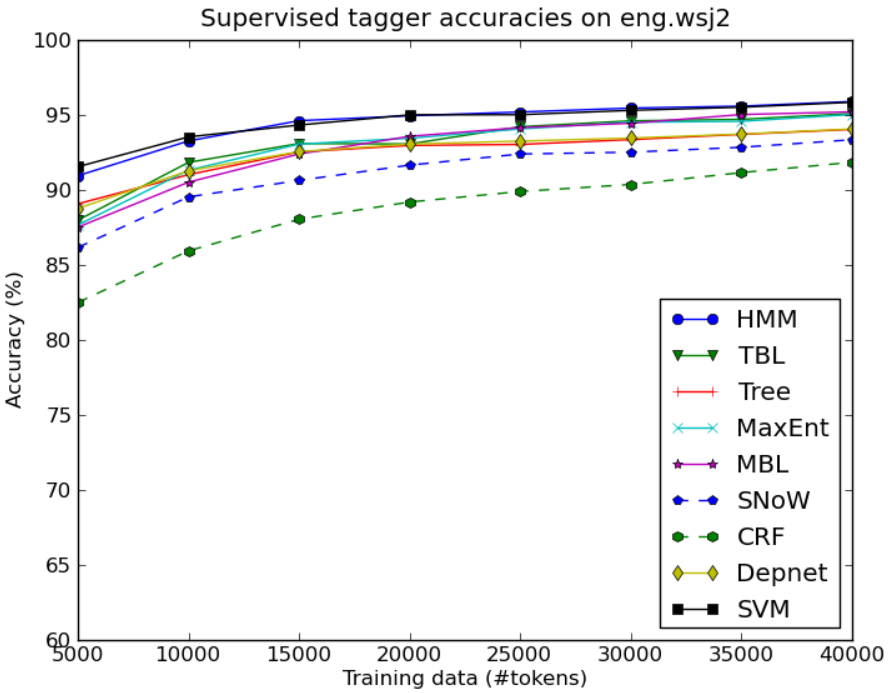
Table 3.8: Supervised tagging: statistical significance using the reduced tagsets

Tokens	**Statistical Significance**					
	HMM	**SVM**	t	**HMM**	**Next**	t
eng.wsj2						
5,000	90.95	91.55	1.494	90.95	89.08	4.391
10,000	93.27	93.54	0.765	93.27	91.84	3.834
15,000	94.62	94.33	0.893	94.62	93.10	4.457
20,000	94.95	95.01	0.193	94.95	93.58	4.147
25,000	95.20	95.02	0.587	95.20	94.22	3.081
30,000	95.46	95.31	0.503	95.46	94.61	2.754
35,000	95.59	95.52	0.239	95.59	95.03	1.864
40,000	95.90	95.85	0.177	95.90	95.21	2.356
afr.nwu2						
5,000	87.95	88.64	1.510	87.95	87.13	1.747
10,000	90.52	90.84	0.774	90.52	89.12	3.258
15,000	91.31	91.79	1.214	91.31	90.46	2.078
20,000	91.87	92.08	0.544	91.87	91.19	1.718
25,000	93.51	93.72	0.604	93.51	93.15	1.015
30,000	93.98	94.52	1.632	93.98	93.58	1.165
35,000	94.16	94.83	2.067	94.16	93.80	1.065
40,000	94.64	95.17	1.696	94.64	94.33	0.955

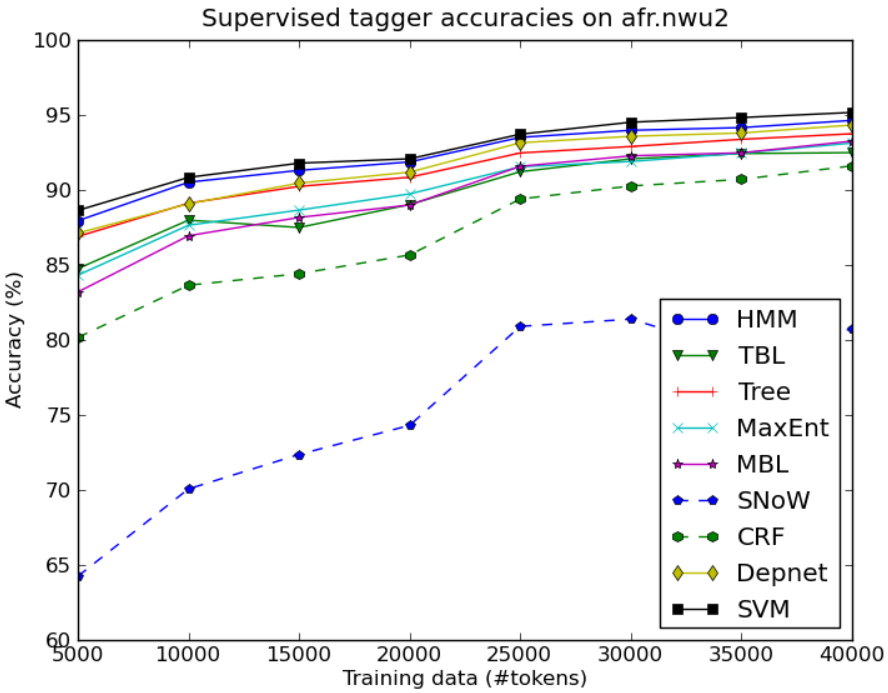
Tables 3.9 and 3.10 show the same consistent performance on known and ambiguous words and varied performance on unknown words (the HMM, MaxEnt and SVM taggers still the best); the absolute accuracies are just higher because of the reduced tagsets. Furthermore, they show that the unknown word classifier for the SNoW tagger can still not be trained successfully on afr.nwu2, and that the CRF tagger exhibits the same behaviour towards unambiguous known words.

3.4 Conclusion

The first experimental investigation tested the performance of the popular supervised POS tagging algorithms on small training corpora of English and Afrikaans. The aim was to provide a more concrete answer than the literature study to the research question of which algorithm performs the



(a)



(b)

Figure 3.2: Supervised tagging: overall accuracies using the reduced tagsets

Table 3.9: Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **eng.wsj2** (using the reduced tagset)

Tokens	**Tagger Accuracy (%)**								
	HMM	**TBL**	**Tree**	**MaxEnt**	**MBL**	**SNoW**	**CRF**	**Depnet**	**SVM**
5,000									
K	96.68	96.46	96.37	91.15	96.42	96.17	88.41	96.49	96.19
A	85.22	82.23	81.58	81.37	81.58	78.16	79.01	82.66	82.01
U	79.85	71.56	74.95	80.91	70.26	66.79	71.03	73.79	82.55
10,000									
K	96.95	97.05	96.60	94.31	96.67	96.46	90.54	96.98	96.85
A	90.67	91.26	88.72	89.40	88.97	87.70	87.19	90.84	90.67
U	83.42	77.88	76.06	83.31	74.01	70.97	73.61	76.02	84.65
15,000									
K	97.05	96.81	96.53	94.81	96.90	96.54	91.85	96.95	96.70
A	93.58	92.63	91.36	91.89	93.00	91.57	92.26	93.21	92.84
U	86.93	81.39	79.92	87.52	78.19	71.97	76.05	78.66	86.85
20,000									
K	96.91	96.81	96.64	95.09	96.64	96.55	92.43	96.90	96.78
A	93.05	92.65	92.00	92.60	92.00	91.65	92.15	93.00	93.40
U	87.91	79.67	79.81	87.54	82.58	74.11	77.63	79.30	88.65
25,000									
K	97.08	97.03	96.65	95.62	96.77	96.76	92.95	96.85	96.83
A	93.19	93.00	91.58	92.38	92.05	92.01	92.10	92.34	92.90
U	87.90	83.31	79.01	88.05	84.00	75.46	78.02	79.31	88.00
30,000									
K	97.19	96.94	96.66	96.01	96.86	96.80	93.40	96.87	96.90
A	94.03	93.13	92.14	93.49	92.86	92.64	92.73	92.91	93.53
U	88.17	84.81	79.50	88.22	84.28	74.50	77.66	79.07	88.64
35,000									
K	97.24	97.05	96.79	96.01	97.19	96.88	94.02	96.96	96.99
A	94.01	93.35	92.38	93.13	93.84	92.73	93.22	93.04	93.26
U	88.29	84.35	80.09	88.24	85.50	75.05	78.56	79.43	89.00
40,000									
K	97.28	96.97	96.76	96.15	97.24	96.85	94.47	96.95	97.05
A	94.15	93.07	92.24	93.46	93.98	92.63	93.46	92.98	93.46
U	88.94	85.64	80.32	89.36	84.96	75.67	78.61	79.52	89.79

Table 3.10: Supervised tagging: accuracies for known (K), ambiguous (A) and unknown (U) words on **afr.nwu2** (using the reduced tagset)

Tokens	**Tagger Accuracy (%)**								
	HMM	**TBL**	**Tree**	**MaxEnt**	**MBL**	**SNoW**	**CRF**	**Depnet**	**SVM**
5,000									
K	94.78	95.26	94.59	90.66	94.60	93.56	89.79	95.20	95.44
A	89.42	90.87	89.50	89.20	88.89	89.81	88.93	90.69	91.39
U	72.99	61.73	70.05	70.44	58.22	0.00	59.09	69.44	73.73
10,000									
K	95.22	95.00	94.80	91.81	94.81	94.24	90.73	95.11	95.18
A	90.02	90.62	89.39	89.87	88.84	90.81	89.39	89.71	89.94
U	76.88	67.66	72.67	75.58	64.16	0.00	63.10	71.60	78.26
15,000									
K	95.17	93.01	94.74	92.09	94.64	93.48	90.37	95.30	95.25
A	89.54	85.03	89.17	89.06	88.00	88.12	88.53	89.92	89.81
U	78.09	68.60	74.82	76.88	66.01	0.00	64.00	73.84	79.92
20,000									
K	95.07	93.87	94.88	92.53	94.50	93.72	90.97	95.36	95.09
A	88.82	87.51	88.82	88.78	87.18	88.31	88.56	89.65	88.93
U	79.62	70.50	75.43	79.03	67.97	0.00	65.38	75.23	80.55
25,000									
K	95.74	94.60	95.46	93.53	95.42	94.29	92.83	95.94	95.69
A	90.52	90.19	90.13	90.26	89.63	89.50	90.72	91.08	90.39
U	80.00	70.82	74.38	79.57	68.33	0.00	68.61	76.30	81.78
30,000									
K	95.99	94.93	95.65	93.60	95.76	93.57	93.53	96.32	96.42
A	91.92	91.78	91.41	91.50	91.36	88.60	92.38	92.72	93.01
U	80.53	73.08	74.55	80.53	69.05	0.00	68.50	75.25	81.77
35,000									
K	96.02	95.06	95.83	94.07	95.72	90.24	93.76	96.32	96.36
A	91.99	92.10	91.57	92.16	91.27	80.59	92.74	92.71	92.85
U	80.89	73.56	75.86	80.89	69.28	0.00	68.86	75.78	83.86
40,000									
K	96.19	95.06	95.96	94.55	96.18	90.53	94.27	96.51	96.66
A	92.23	91.92	91.68	92.62	92.20	80.95	93.06	93.00	93.36
U	81.79	71.24	75.54	81.42	69.19	0.00	69.65	76.38	82.91

best on little data. In order to realise the aim, a controlled experimental environment was set up in which the language, amount of training data and tagset size was kept constant. The result is a comparative list of accuracies for all the tagger implementations for increasing subsets of training data, so that not only the best tagger may be identified, but also its point of optimal data use. The list was compiled for comprehensive tagsets of the two languages, as well as reduced TTS-efficient versions.

For both tagset versions, the HMM and SVM taggers were consistently the most accurate. This is contrary to the research hypothesis, which states that the tagger performing the best on much data—the cyclic dependency network tagger—should do the same on little data. To break the tie between the HMM and SVM taggers, the HMM tagger is chosen because it was shown to be much faster. It is then also duly noted that classifier combination should increase accuracy, but that its computational expense makes it impractical for TTS.

The top accuracy of the HMM tagger is 95.90% for English on a 40,000-token training data set using the TTS-efficient tagset of 18 tags. At the optimal data use point of 20,000 tokens, its accuracy is 94.95%. For Afrikaans the HMM tagger is 94.64% accurate on the maximum of 40,000 training tokens using the TTS-efficient tagset of 17 tags. Here the optimal data use point is at 30,000 tokens with an accuracy of 93.98%. The point is further because of the significantly higher number of ambiguous words in the Afrikaans test set.

The research question has been answered for the Germanic languages of English and Afrikaans. It would be prudent though to test algorithm performance on the other South African languages, such as those of the Sotho and Nguni families. This is left as future work. Now that an optimal POS tagger has been identified, the research can move onto the effects of POS information on the naturalness of TTS synthesis in the next chapter.

Chapter 4

Part-of-Speech Effects on Text-to-Speech Synthesis

4.1 Introduction

It was motivated in Chapter 1 that the naturalness of a TTS voice is mainly determined by prosody and that, in turn, many aspects of prosody are dependent on POS information. In the light of the scarceness of resources, the question arises whether it is perhaps possible to circumvent the traditional approaches to prosodic modelling by learning the latter directly from the speech data using POS information. In other words, does the addition of POS features to the HTS context labels improve the naturalness of a TTS voice?

This is the question that the second experimental investigation aims to answer. HTS voices are trained for Speect from English and Afrikaans prosodically rich speech. The voices are compared with and without POS features incorporated into the HTS context labels, analytically and perceptually. For the analytical experiments, measures of prosody to quantify the comparisons are explored. It is then also noted whether the results of the perceptual experiments correlate with their analytical counterparts.

Section 4.2 describes various setup details common to all the experiments that follow. Section 4.3 records the experiment in which the POS information is added to already feature-rich HTS labels. The experiment in Section 4.4 repeats the first one, now only with minimum features in the HTS labels. Section 4.5 explores whether the results of the second experiment can be duplicated when using a less accurate POS tagger. The last experiment is recorded in Section 4.6 and it compares voices with maximum and minimum features in the HTS labels. The investigation concludes in Section 4.7.

4.2 Common Setup

4.2.1 Tagger

The fast, accurate HMM tagger from the previous chapter is incorporated into Speect to obtain the POS information at synthesis time. To review: for English, the tagger is trained on a 40,000-token subset of the Penn Treebank WSJ corpus [54] using a reduced set of 18 tags. The tagger obtains

an accuracy of 95.90% on a separate 10,000-token test set from the same corpus. For Afrikaans, 40,000 tokens of the balanced corpus developed in [56] are used as training data. The tagset is reduced to 17 tags. The tagger is 94.64% accurate on the test set of 10,000 tokens, also from the corpus.

4.2.2 Voices

The data requirements for voice building are recorded speech segmented into utterances (spoken sentences), and transcriptions thereof. For the English voice, data from the CMU_ARCTIC speech synthesis databases (available at `http://festvox.org/cmu_arctic/`) is used: the speaker is "US bdl", a United States English-speaking male. The data consist of 1,132 utterances. For the Afrikaans voice, speech data recorded in-house for the Lwazi project is used; the speaker is female. The number of utterances is 1,005. For each voice 100 random utterances from the data are kept aside as a test data set. These corpora are also quite small because the recording process is very cumbersome in terms of obtaining enough utterances at a sufficient quality.

An HTS voice is built in a two-stage process: phonetic alignment and HMM training. The phonetic alignment is performed by the Speect NLP front-end and an additional tool (as part of a toolkit released with Speect) that uses forced-alignment based on HTK [96]. The NLP front-end maps the transcriptions to phoneme sequences. The alignment tool allows model initialisation from manually aligned speech data transcribed in a different language or phoneset by mapping to broad phonetic categories. This is highly beneficial as a bootstrapping technique for the alignment of a small corpus of an RSL [92]. The TIMIT corpus [30] is used as bootstrapping data.

The HMM training from the aligned utterances is performed by the demonstration script released with the HTS engine. It uses HTS version 2.1 [97] to build the models for the hts_engine API version 1.02 [86]. The script is only slightly altered to accommodate Speect as a front-end.

The question file of the demonstration script is mostly used as is for the model tying decision tree. Only the POS-related questions are modified to reflect the tagsets chosen in Section 4.2.1 and, for the Afrikaans voice, the phonetic category questions are altered to reflect the Afrikaans phoneset.

The HTS context labels utilise by default a set of linguistic features defined in [87] and included in the demonstration package. The full context labels comprise, inter alia, features based on the identities of the current and neighbouring phonemes, the number and relative positions of phonemes, syllables, words and phrases, whether syllables are stressed or not, and the POS of words.

The voices are built using two versions of the context labels: one with maximum features and the other with minimum features. The maximum feature set comprises all of the features provided by the demonstration package, including the segmental counting and positional features, but excluding the stress and intonation-related ones (so that prosody is not modelled explicitly but only implicitly by the POS information). The minimum features only include the phonemic identities. For the experiments, the two versions are then used with and without POS information. Table 4.1 lists these differences. The content word features e_5 to e_8 are considered part of the POS information, because whether a word is a content or function word is directly derivable from its POS.

Table 4.1: Features to be used in the HTS context labels

Feat	Description	Min feats		Max feats	
		w/o POS	w/ POS	w/o POS	w/ POS
p_1	the phoneme identity before the previous phoneme	✓	✓	✓	✓
p_2	the previous phoneme identity	✓	✓	✓	✓
p_3	the current phoneme identity	✓	✓	✓	✓
p_4	the next phoneme identity	✓	✓	✓	✓
p_5	the phoneme after the next phoneme identity	✓	✓	✓	✓
p_6	position of the current phoneme identity in the current syllable (forward)	✗	✗	✓	✓
p_7	position of the current phoneme identity in the current syllable (backward)	✗	✗	✓	✓
a_1	whether the previous syllable stressed or not (0: not stressed, 1: stressed)	✗	✗	✗	✗
a_2	whether the previous syllable accented or not (0: not accented, 1: accented)	✗	✗	✗	✗
a_3	the number of phonemes in the previous syllable	✗	✗	✓	✓
b_1	whether the current syllable stressed or not (0: not stressed, 1: stressed)	✗	✗	✗	✗
b_2	whether the current syllable accented or not (0: not accented, 1: accented)	✗	✗	✗	✗
b_3	the number of phonemes in the current syllable	✗	✗	✓	✓
b_4	position of the current syllable in the current word (forward)	✗	✗	✓	✓
b_5	position of the current syllable in the current word (backward)	✗	✗	✓	✓
b_6	position of the current syllable in the current phrase (forward)	✗	✗	✓	✓
b_7	position of the current syllable in the current phrase (backward)	✗	✗	✓	✓
b_8	the number of stressed syllables before the current syllable in the current phrase	✗	✗	✗	✗
b_9	the number of stressed syllables after the current syllable in the current phrase	✗	✗	✗	✗
b_{10}	the number of accented syllables before the current syllable in the current phrase	✗	✗	✗	✗
b_{11}	the number of accented syllables after the current syllable in the current phrase	✗	✗	✗	✗
b_{12}	the number of syllables from the previous stressed syllable to the current syllable	✗	✗	✗	✗
b_{13}	the number of syllables from the current syllable to the next stressed syllable	✗	✗	✗	✗
b_{14}	the number of syllables from the previous accented syllable to the current syllable	✗	✗	✗	✗
b_{15}	the number of syllables from the current syllable to the next accented syllable	✗	✗	✗	✗
b_{16}	name of the vowel of the current syllable	✗	✗	✓	✓
c_1	whether the next syllable stressed or not (0: not stressed, 1: stressed)	✗	✗	✗	✗
c_2	whether the next syllable accented or not (0: not accented, 1: accented)	✗	✗	✗	✗
c_3	the number of phonemes in the next syllable	✗	✗	✓	✓
d_1	pos (part-of-speech) of the previous word	✗	✓	✗	✓
d_2	the number of syllables in the previous word	✗	✗	✓	✓
e_1	pos (part-of-speech) of the current word	✗	✓	✗	✓
e_2	the number of syllables in the current word	✗	✗	✓	✓
e_3	position of the current word in the current phrase (forward)	✗	✗	✓	✓
e_4	position of the current word in the current phrase (backward)	✗	✗	✓	✓
e_5	the number of content words before the current word in the current phrase	✗	✓	✗	✓
e_6	the number of content words after the current word in the current phrase	✗	✓	✗	✓
e_7	the number of words from the previous content word to the current word	✗	✓	✗	✓
e_8	the number of words from the current word to the next content word	✗	✓	✗	✓
f_1	pos (part-of-speech) of the next word	✗	✓	✗	✓
f_2	the number of syllables in the next word	✗	✗	✓	✓
g_1	the number of syllables in the previous phrase	✗	✗	✓	✓
g_2	the number of words in the previous phrase	✗	✗	✓	✓
h_1	the number of syllables in the current phrase	✗	✗	✓	✓
h_2	the number of words in the current phrase	✗	✗	✓	✓
h_3	position of the current phrase in utterance (forward)	✗	✗	✓	✓
h_4	position of the current phrase in utterance (backward)	✗	✗	✓	✓
h_5	TOBI endtone of the current phrase	✗	✗	✗	✗
$i1$	the number of syllables in the next phrase	✗	✗	✓	✓
$i2$	the number of words in the next phrase	✗	✗	✓	✓
j_1	the number of syllables in this utterance	✗	✗	✓	✓
j_2	the number of words in this utterance	✗	✗	✓	✓
j_3	the number of phrases in this utterance	✗	✗	✓	✓

4.2.3 Analytical Test

In order to measure the prosodic effects of POS information on synthesised speech, it is necessary to understand what the physical manifestations of prosody are. *Duration*, *pitch (F0)* and *intensity* have been shown to be acoustic correlates of prosody [26, 93]. Duration is simply the temporal length of segments of speech, whether it be phones, syllables, words or phrases. Pitch is the relative highness or lowness of a tone and is determined by the rate of vibration of the vocal cords [16]. Intensity is the energy of the signal of a speech segment and is related to the amplitude of the vocal cord vibrations [15]. These three measures are used to determine the closeness of each synthesised utterance to its natural speech counterpart from the 100-utterance test set.

Duration The natural speech utterance is phonetically aligned to determine the duration of each phoneme. For each phoneme, the corresponding duration of the synthesised utterance is subtracted and the absolute value is taken to represent the distance between the natural phoneme and the synthesised phoneme.

Pitch The F0 contour of the natural speech utterance is extracted with Praat [9] and divided according to the aligned durations so that each phoneme is assigned its corresponding section of F0 values. For the synthesised utterance, the HTS engine is forced to use the same duration alignments and output the synthesised F0 values. The distance between the natural and the synthesised phoneme is taken as the Mean Squared Error (MSE) of the synthesised F0 values $\hat{\mathbf{f_0}}$ to the natural ones $\mathbf{f_0}$:

$$MSE\left(\hat{\mathbf{f_0}}\right) = E\left[\left(\hat{\mathbf{f_0}} - \mathbf{f_0}\right)^2\right] \tag{4.1}$$

where

$$\begin{aligned} E\left[\mathbf{x}\right] &= \sum_i p_i x_i \\ &= \frac{1}{n}\sum_i x_i \quad \text{if } p_i = \frac{1}{n} \end{aligned} \tag{4.2}$$

Any differences involving undefined F0 values are taken as zeros in the summations.

Intensity The intensity contour is extracted in similar fashion to the F0 contour. Praat is used for both the natural speech and synthesised utterances, the latter once again being aligned on the phoneme boundaries of the former. The distance between the phoneme sections of intensity values is also the MSE.

When comparing two TTS voices, for example with and without POS information, the one voice is deemed more natural than the other for a particular utterance if its synthesised version is closer to the natural speech counterpart than the synthesised version of the other voice.

An experiment is thus compiled from the test set by synthesising the 100 test sentences with both voices. Each utterance pair is scored by counting the number of phonemes that are closer to the natural speech for a particular measure. The utterance with the highest number of phonemes wins, and the corresponding voice is accredited with that test sentence for the measure. The voice

accredited with the most test sentences in the end is then more natural. Finally, McNemar's test is used to examine the statistical significance of the result.

McNemar's test is a chi-square test for paired sample data [10, 81]. A most intuitive example is a scenario in the medical industry where sick and healthy test subjects are counted before and after receiving a drug. McNemar's test is applied to see whether the drug has an effect or not. Test data are recorded in a 2-by-2 contingency table as depicted in Table 4.2.

Table 4.2: Contingency table for McNemar's test

		After Treatment	
		Sick	**Healthy**
Before Treatment	**Sick**	A	B
	Healthy	C	D

The null hypothesis for McNemar's test assumes that the sum of the rows in the contingency table is equal to the sum of the columns. This implies that $B = C$ and that the drug has no effect on the illness. The alternative hypothesis assumes that the sum of the rows is not equal to the sum of the columns, hence that the drug has an effect.

From the above McNemar's test can be calculated as follows:

$$\chi^2 = \frac{(|B - C| - 0.5)^2}{B + C} \tag{4.3}$$

χ^2 has a chi-squared distribution with one degree of freedom (if $B + C$ is large enough). To test for significance, χ^2 is compared to the appropriate chi-square table value (Table B.2 in Appendix B). A result of probability greater or equal to 0.05 is generally considered to be significant. Lining up this boundary probability with one degree of freedom in the table gives a value of 3.841. If $\chi^2 \geq 3.841$ the null hypothesis is rejected in favour of the alternative hypothesis. If $\chi^2 < 3.841$ the null hypothesis is accepted.

Returning to the naturalness results of a pair of TTS voices, let n_1 be the number of utterances accredited to the first voice and n_2 to the second voice. McNemar's test can then be applied by setting $B = n_1$ and $C = n_2$. If $\chi^2 \geq 3.841$ the winning voice is significantly more natural than the other voice. If $\chi^2 < 3.841$ the result is insignificant and the two voices can be said to be similar in their degree of naturalness.

4.2.4 Perceptual Test

A perceptual test is also set up in an effort to validate the analytical results. Respondents are asked to listen to pairs of utterances from the test set. In a pair the two utterances, A and B, are synthesised from the same sentence, but each by a different TTS voice. The respondents must then choose which utterance out of the pair sounds more natural relatively, or if both sound the same (without listening to the original natural speech). Duration, pitch and intensity are not distinguished; only an aggregate judgement is required.

This "A versus B" approach (with McNemar's test for significance) is preferred above a mean opinion score (with the Wilcoxon signed rank test [53] for significance) that is used, for example, in the Blizzard Challenge [42]. The reason is that it is more robust against respondent subjectivity:

different respondents are more likely to judge the same utterance out of a pair as more natural than assign it the same score on a scale of 1 to 5.

McNemar's test can be used in reverse to calculate how many pairs will be needed to obtain a significant result. Recall that

$$\frac{(|B-C|-0.5)^2}{B+C} \geq 3.841 \tag{4.4}$$

is required for statistical significance. This may be rewritten in terms of the total number of pairs N:

$$\frac{(xN-0.5)^2}{yN} \geq 3.841 \qquad x,y \in [0,1] \tag{4.5}$$

where $xN = |B-C|$, the ratio of N estimated to be equal to the difference between the discordant pairs. For fixed y, the smaller this difference (or x) is, the bigger N must be for significance. $yN = B+C$, the ratio of N estimated to be equal to the sum of the discordant pairs, in other words, the number of pairs not judged equal in naturalness. For fixed x, the smaller this sum (or y) is, the smaller N needs to be for significance. A change in x varies N to a greater degree than a change in y does.

For conservative estimates of $x = 0.2$ and $y = 0.8$, $N \geq 82$. Nevertheless, a large safety margin is built into the test by setting $N = 200$. The 200 pairs are divided up among 10 respondents so that each respondent must listen to 20 pairs. The 20 sentences that make up the pairs are randomly selected from the 100-utterance test set, such that every two respondents listen to a unique subset.

For the two languages, the 20 respondents are mother-tongue speakers with an average age of between 30 and 35. Out of the 10 English respondents, 6 are male and 4 female. 7 Afrikaans respondents are male and 3 female. A web-based interface facilitates the playback of the audio samples and recording of answers.

4.3 Experiment 1: POS Effects Using Maximum Features

The first experiment compares the naturalness of two TTS voices of which the HTS context labels use the maximum features as listed in Table 4.1. The English voices are dubbed **eng_maxlab_nopos** for the version without POS information, and **eng_maxlab_pos40k** for the version with POS information. Similarly, the Afrikaans voices are named **afr_maxlab_nopos** and **afr_maxlab_pos40k**. The aim is to observe the effect of the POS information in an already feature-rich environment for maximum benefit.

Table 4.3 shows the results for English and Afrikaans. The first column lists the measures and the second column the total number of utterances evaluated. Columns 3 and 4 list the number of utterances accredited to each voice and column 5 the number found equal. The last column lists the McNemar χ^2-scores for significance (which are independent of the equal counts).

The analytical figures across the two languages show no significant bias towards a particular voice, and the perceptual figures confirm this (there are basically as many votes for the one voice as for the other). Therefore, it may be deduced that the two voices are similar in their degree of naturalness and that the POS information has no effect when using the maximum features. It may

Table 4.3: Naturalness results when using maximum features

Measure	**Utterances**				χ^2
	Total	**eng_maxlab_nopos**	**eng_maxlab_pos40k**	**Equal**	
Duration	100	46	49	5	0.066
Pitch	100	52	42	6	0.960
Intensity	100	41	52	7	1.185
Perception	200	72	83	45	0.711
Measure	**Utterances**				χ^2
	Total	**afr_maxlab_nopos**	**afr_maxlab_pos40k**	**Equal**	
Duration	100	48	47	5	0.003
Pitch	100	48	45	7	0.067
Intensity	100	49	43	8	0.329
Perception	200	72	74	54	0.015

be that the POS effects are "drowned out" by the other features, which inherently carry similar information beneficial towards naturalness.

Finally, it is observed that the equal count among the perceptual figures is proportionally much higher than among the analytical figures. The simple explanation is that it is much more difficult for respondents to hear a distinction between two utterances than what it is to calculate the difference between two discrete values.

4.4 Experiment 2: POS Effects Using Minimum Features

Since the use of the maximum features is suspected to suppress the POS effects, it is prudent to retest the TTS voices with reduced features in order to lift out the effects. The English voice without POS information is **eng_minlab_nopos** and with **eng_minlab_pos40k**. The corresponding Afrikaans voices are **afr_minlab_nopos** and **afr_minlab_pos40k**. The results are shown in Table 4.4.

For both languages, pitch dominates the results by clearly favouring the voices with POS information as more natural. The near significant perceptual figures tend to suggest the same. Of note is the disparate results for intensity: it manifests as a deciding factor only for Afrikaans. It is unclear from this experiment as to what the cause might be; see trends observed in the rest of the experiments.

4.5 Experiment 3: POS Effects Using a Less Accurate Tagger

From the previous experiment it was seen that, when only minimum features are available, adding POS information improves the pitch component of naturalness. Within the context of an RSL, the question now arises whether the same effect is possible when a less accurate POS tagger, trained on fewer resources, is used. The third experiment thus compares two TTS voices, both with POS information on top of the minimum features, but where the tagger of the one voice has

Table 4.4: Naturalness results when using minimum features

Measure	**Utterances**				χ^2
	Total	**eng_minlab_nopos**	**eng_minlab_pos40k**	**Equal**	
Duration	100	51	44	5	0.445
Pitch	100	26	66	8	16.959
Intensity	100	46	46	8	0.003
Perception	200	76	101	23	3.391
Measure	**Utterances**				χ^2
	Total	**afr_minlab_nopos**	**afr_minlab_pos40k**	**Equal**	
Duration	100	51	40	9	1.212
Pitch	100	15	79	6	42.896
Intensity	100	35	55	10	4.225
Perception	200	71	95	34	3.327

been trained on only 5,000 tokens. For English, the 5,000-token tagger is 90.95% accurate and the corresponding voice is called **eng_minlab_pos05k**. The voice of the normal 40,000-token, 95.90% accurate tagger is called **eng_minlab_pos40k**. For Afrikaans, the tagger trained on 5,000 tokens is 87.95% accurate and its voice is dubbed **afr_minlab_pos05k**. The voice of the 40,000-token, 94.64% accurate tagger used so far is dubbed **afr_minlab_pos40k**. Table 4.5 shows the naturalness results for this experiment.

Table 4.5: Naturalness results when using a less accurate tagger

Measure	**Utterances**				χ^2
	Total	**eng_minlab_pos05k**	**eng_minlab_pos40k**	**Equal**	
Duration	100	48	46	6	0.024
Pitch	100	45	49	6	0.130
Intensity	100	39	55	6	2.556
Perception	200	69	94	37	3.683
Measure	**Utterances**				χ^2
	Total	**afr_minlab_pos05k**	**afr_minlab_pos40k**	**Equal**	
Duration	100	46	46	8	0.003
Pitch	100	48	46	6	0.024
Intensity	100	44	53	3	0.745
Perception	200	57	89	54	6.796

Pitch, the prominent measure in the previous experiment, does not feature here for any of the two languages, nor do any of the other analytical measures. These insignificant differences between the voices with the more and less accurate taggers may support the hypothesis that one can achieve the same prosodic effects with the less accurate tagger (read fewer resources). However, the perceptual figures do show a bias towards the voices using the more accurate tagger, near significantly for English and significantly for Afrikaans. This mismatch between the analytical and

perceptual results renders the experiment inconclusive.

4.6 Experiment 4: Comparing Minimum and Maximum Features

The final experiment revisits the implication of the first, namely that the maximum features might compensate for the effect of adding POS information. The minimum feature voices with POS information, **eng_minlab_pos40k** for English and **afr_minlab_pos40k** for Afrikaans, are set against the maximum feature voices without POS information, **eng_maxlab_nopos** for English and **afr_maxlab_nopos** for Afrikaans. The results are shown in Table 4.6.

Table 4.6: Results of the comparison between minimum and maximum features

Measure	**Utterances**				χ^2
	Total	**eng_minlab_pos40k**	**eng_maxlab_nopos**	**Equal**	
Duration	100	35	59	6	5.875
Pitch	100	44	52	4	0.586
Intensity	100	46	46	8	0.003
Perception	200	69	104	27	6.880
Measure	**Utterances**				χ^2
	Total	**afr_minlab_pos40k**	**afr_maxlab_nopos**	**Equal**	
Duration	100	42	48	10	0.336
Pitch	100	35	61	4	6.773
Intensity	100	61	34	5	7.392
Perception	200	64	93	43	5.174

The duration and pitch figures of the two languages suggest that the maximum feature voices without POS information are more natural (duration and pitch being significant for each language in turn). The perceptual test results favour these voices significantly as well. Therefore, the extra counting and positional features in the maximum feature set (Table 4.1) not only compensate for the POS information (they would then have insignificant differences as a result), they improve the naturalness beyond what the POS tags can affect.

The contradicting Afrikaans intensity figures indicate another anomaly, as has been noted in Section 4.4 as well. This behaviour can possibly be ascribed to speaker variability (choice) in the speech corpora of the two languages. During the reinspection of the Afrikaans data, alignment errors as a result of unexpected pauses, transcription errors, mispronunciations and G2P errors were also found. Such imperfections of the data—which are likely to be more common for RSLs—will generally induce more variance in the experimental results. As long as the errors are not systematic, however, reliable trends can be deduced, as has been done here for pitch.

4.7 Conclusion

In summary, it has been shown that POS information does contribute to the naturalness (specifically in terms of pitch) of a TTS voice when it forms part of a small phoneme identity-based feature

set in the HTS labels. This proves the research hypothesis correct. However, the same effect, even an improvement, can be accomplished by including segmental counting and positional information instead of the POS tags in the HTS labels—and no extra resources are used. Therefore, it is not necessary to incur the cost of POS tagging when the traditional route of prosodic modelling cannot be followed in the development of a TTS voice.

The experiments were limited though to English and Afrikaans. As noted in Section 3.4 regarding the POS tagging results, the next step would be to test the POS effects as well on the other South African languages. In particular, results for the tone-driven Bantu languages would be very informative.

Notwithstanding the above outcome, it is problematic that the correlation between the analytical and perceptual methods is not yet clear-cut. This is because the analytical measures did not always behave in a consistent way across the two languages and the four experiments. The problem can be addressed from both sides: either the perceptual tests should be more fine-grained (that is duration, pitch and intensity must be judged separately), or a new analytical framework can be used where, for example, the three measures are combined into a single one. The former is very difficult to achieve since the human ear cannot discern such differences well. The latter is possible by constructing a classifier such as the Gaussian discriminative function presented in [93]. In either case, it warrants a much more thorough study of the acoustic and perceptual factors of prosody.

Chapter 5

Conclusion

5.1 Summary and Conclusions

The Lwazi project in South Africa collects and applies human language technology resources for all eleven of the official languages. The challenge, however, is that the South African languages are resource-scarce: little data and linguistic knowledge are available to them. The consequence is that the project requires innovative solutions to meet its outcomes. One of the deliverables of the project is more natural voices for TTS synthesis. Naturalness is primarily determined by prosody and it has been shown in Section 1.2.2 that many aspects of prosodic modelling is dependent on POS information.

In a resource-scarce environment it is difficult to exploit this information for the following reasons:

1. An automatic tagger is required to obtain the POS information from the text to be synthesised, but state-of-the-art POS taggers are data-driven and thus require large amounts of labelled training data.

2. The subsequent processes in TTS that are used to apply the POS information towards prosodic modelling are resource-intensive themselves: some require non-trivial linguistic knowledge; others require labelled data as well.

This research addressed the following questions arising from the above problem statements:

1. Which POS tagging algorithm achieves the highest accuracy on little data?

2. Can the direct incorporation of POS features into the context labels of an HTS voice improve its naturalness and, so, provide a more efficient alternative to the traditional processes of pronunciation and prosodic modelling?

5.1.1 An Optimal Part-of-Speech Tagging Algorithm

Chapter 2 set out to answer the first research question by reviewing the most popular data-driven algorithms. Since literature to date consists mostly of isolated papers discussing one specific algorithm, the aim of the review was to consolidate the research into a single point of reference. Where the focus in the past has been on improving performance on large corpora, the review studied the

behaviour of the algorithms on small corpora as well. It was noted that the resource requirements of the data-driven POS tagging algorithms are determined by the learning paradigms they employ: supervised algorithms need the data to be labelled manually, a very expensive task. Unsupervised algorithms only require raw, unlabelled data—making them very attractive—but their accuracies are still far below those of their supervised counterparts. Semisupervised algorithms draw a compromise by operating on a small set of labelled data and a large set of unlabelled data. This partial supervision improves accuracy beyond that of unsupervised alternatives, while alleviating some of the burden imposed by manual labelling. The scope of the research only allowed supervised algorithms to be explored in depth.

It was shown that the supervised tagging algorithms perform very well on large English corpora, with most achieving an accuracy in the 96–97% range. The best algorithm implementation is the cyclic dependency network tagger. The logarithmic behaviour inherent to data-driven algorithms were also discussed. They require more and more additional data as the training data grows to improve their accuracies by the same percentage point. Such small increases in accuracy do not justify the extra data in a resource-scarce context. In the case of RSLs, it is, therefore, not only prudent to find the most accurate tagger, but also the point where the data-accuracy trade-off is optimal.

The performance is notably worse on little data and less consistent at a range of 72–85%, with the HMM tagger being the top scorer. The results are inconclusive, however, because the literature review could only find accuracy figures on small corpora for some of the tagging algorithms. Notably, scores for the cyclic dependency network tagger, which performs best on much data, are missing. Another contributing factor to the incongruity was that the taggers were applied to different RSLs.

Chapter 3, therefore, revisited the research question of which available POS tagging algorithm will be the most accurate given little training data, by testing existing, freely available implementations of the algorithms. The experimental environment was controlled to obtain conclusive results. The same language, amount of data and tagset were used across the different taggers. English featured again as a reference language by restricting its data to emulate resource scarceness. In addition, another Germanic language, Afrikaans, was used as a true RSL. The taggers were tested on incremental subsets of the training data so that the optimal data-accuracy trade-off points could be identified. The experiments employed comprehensive and reduced TTS-efficient POS tagsets for both languages; the reduced tagsets were used to note the effect on the tagger accuracies.

For both tagset versions, the HMM and SVM taggers consistently outperformed the rest. This result proves the research hypothesis incorrect: the cyclic dependency network tagger did not achieve the highest accuracy on little data. The HMM tagger is chosen above the SVM tagger: the HMM is not only highly accurate, but much more efficient in terms of speed, and thus truly makes for a well-rounded solution to the tagging problem.

The following exact figures summarise the outcome of the experiment: the top accuracy of the HMM tagger is 95.90% for English on a 40,000-token training data set using the TTS-efficient tagset of 18 tags. At the optimal data use point of 20,000 tokens, its accuracy is 94.95%. For Afrikaans the HMM tagger is 94.64% accurate on the maximum of 40,000 training tokens using the TTS-efficient tagset of 17 tags. Here the optimal data use point is at 30,000 tokens with an

accuracy of 93.98%.

5.1.2 The Effects of Part-of-Speech Features in the HTS Labels

Chapter 4 explored the second research question of whether POS information in the HTS labels improves the naturalness of synthesised speech. HTS voices were trained from English and Afrikaans prosodically rich speech. In four different experimental environments two voices were compared each time with and without POS features incorporated into the HTS context labels, analytically and perceptually. For the analytical tests, the one voice was deemed more natural than the other for a particular utterance if its synthesised version was closer to the natural speech counterpart than the synthesised version of the other voice. Three measures of prosody were employed to quantify the comparisons: duration, pitch and intensity. For the perceptual tests, respondents had to compare the utterances of the two voices without listening to the original natural speech. McNemar's test was used to determine the statistical significance of all the results. It was then also noted whether the results of the perceptual tests correlate with their analytical counterparts.

In the first experiment the POS information was incorporated into already feature-rich HTS labels that included not only phoneme identities, but also other segmental positional and counting information. It was shown that the two voices, one with and the other without POS information, are similar in their degree of naturalness and that the POS information has no effect when using the maximum features.

Since the use of the maximum features was suspected to suppress the POS effects, the second experiment retested the TTS voices with reduced features in order to lift out the effects. The HTS labels only contained the phoneme identity-based features. It was seen that the pitch of the voices with POS information improved considerably.

The third experiment tried to answer a question that arose from the previous results, namely whether the same pitch improvement was possible when a less accurate POS tagger, trained on fewer resources, was to be used. A mismatch between the analytical and perceptual results rendered the experiment inconclusive though.

The last experiment revisited the implication of the first, namely that the maximum features might compensate for the effect of adding POS information. Comparing a minimum feature voice with POS information to a maximum feature voice without it, the analytical and perceptual figures showed that the latter voice is more natural.

Hence, although POS information contributes to the naturalness (specifically in terms of pitch) of a TTS voice when it forms part of a small phoneme identity-based feature set, the same effect, even an improvement, can be accomplished by including segmental counting and positional information instead of the POS tags in the HTS labels. Furthermore, this latter information requires no extra resources. Therefore, it is not necessary to incur the cost of POS tagging when the traditional route of prosodic modelling cannot be followed in the development of a TTS voice.

5.2 Future Work

At the conclusion of this study, the insights gained are limited to the Germanic languages of English and Afrikaans. It is necessary to investigate the POS effects on other language types as well to see if any general trends are present. It will be important to establish whether counting and positional features profit other languages in the place of POS features, or whether the latter remains helpful.

If POS features in the HTS context labels are shown to be beneficial in general, it opens up an interesting door to save resources. Since the pronunciation and prosody are then modelled inherently and automatically by the data, it does not matter what the names of the POS categories are, only that there are distinctions (traditional rule-based approaches require sensible names because a human needs to understand them). This allows the anonymous POS categories or *syntactic clusters* of unsupervised distributional clustering [66] (mentioned in Section 2.2.2) to feature in the context labels. If POS features are not beneficial, then the traditional route of pronunciation and prosodic modelling will have to be explored to determine a "second-best" resource-efficient solution.

Another avenue that needs further attention is the correlation between the analytical and perceptual measures of the naturalness of a TTS voice. The physical manifestations of prosody and the relationships among them need to be studied in greater depth to gain a better understanding of how they aggregate into that which the human ear distinguishes perceptually.

Bibliography

[1] H. Agrawal and A. Mani. Part of speech tagging and chunking with conditional random fields. In *Proceedings of the NLPAI Workshop on Machine Learning 2006 Contest*, 2006. http://ltrc.iiit.ac.in/nlpai_contest06/papers/tilda.pdf.

[2] H. Ali. An unsupervised parts-of-speech tagger for the Bangla language. Department of Computer Science, University of British Columbia. http://www.cs.ubc.ca/~carenini/TEACHING/CPSC503-10/FINAL-REPORTS-08/hammad-report1.1.pdf.

[3] Answers.com. Acoustic resonance. http://www.answers.com/topic/acoustic-resonance.

[4] C. Biemann. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 7–12. Association for Computational Linguistics, 2006. http://portal.acm.org/citation.cfm?id=1557856.1557859.

[5] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, 2009. http://www.nltk.org/book.

[6] C. M. Bishop. *Pattern Recognition And Machine Learning*. Springer, first edition, 2006. http://research.microsoft.com/~cmbishop/prml/index.htm.

[7] A. W. Black, H. Zen, and K. Tokuda. Statistical parametric speech synthesis. In *Proceedings of ICASSP 2007*, 2007. http://www-2.cs.cmu.edu/~awb/papers/icassp2007/0401229.pdf.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. http://portal.acm.org/citation.cfm?id=944937.

[9] P. Boersma and D. Weenink. Praat: doing phonetics by computer. http://www.praat.org/.

[10] S. Boslaugh and P. A. Watters. *Statistics in a nutshell*. O'Reilly Media, Inc., first edition, 2008.

[11] T. Brants. TnT – a statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing (ANLP-2000)*, 2000. http://www.coli.uni-saarland.de/~thorsten/publications/Brants-ANLP00.pdf.

[12] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155. Association for Computational Linguistics, 1992. http://www.aclweb.org/anthology/A/A92/A92-1021.pdf.

[13] E. Brill. Some advances in transformation-based part of speech tagging. In *Proceedings of the twelfth national conference on artificial intelligence*, pages 722–727, 1994. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.8357.

[14] E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In *Natural Language Processing Using Very Large Corpora*, pages 1–13. Kluwer Academic Press, 1995. http://www.aclweb.org/anthology/W/W95/W95-0101.pdf.

[15] Britannica Online Encyclopedia. Intensity. http://www.britannica.com/EBchecked/topic/559032/speech/68986/Intensity.

[16] Britannica Online Encyclopedia. Pitch. http://www.britannica.com/EBchecked/topic/1357164/pitch.

[17] K. W. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, pages 136–143. Association for Computational Linguistics, 1988. http://www.aclweb.org/anthology/A/A88/A88-1019.pdf.

[18] A. Clark. Inducing syntactic categories by context distribution clustering. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 91–94, Morristown, NJ, USA, 2000. Association for Computational Linguistics. www.aclweb.org/anthology/W/W00/W00-0717.pdf.

[19] A. Clark. Combining distributional and morphological information for part of speech induction. In *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pages 59–66, Morristown, NJ, USA, 2003. Association for Computational Linguistics. www.cs.rhul.ac.uk/home/alexc/papers/eacl2003.pdf.

[20] S. Clark, J. R. Curran, and M. Osborne. Bootstrapping POS taggers using unlabelled data. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, pages 49–55. Association for Computational Linguistics, 2003. http://acl.ldc.upenn.edu/W/W03/W03-0407.pdf.

[21] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, pages 133–140. Association for Computational Linguistics, 1992. http://www.aclweb.org/anthology/A/A92/A92-1018.pdf.

[22] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis. MBT: A memory-based part of speech tagger-generator. *The Computing Research Repository*, 1996. http://www.ldc.upenn.edu/acl/W/W96/W96-0102.pdf.

[23] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972. http://www.cs.toronto.edu/~roweis/csc412-2006/extras/gis.pdf.

[24] S. Dasgupta and V. Ng. Unsupervised part-of-speech acquisition for resource-scarce languages. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 218–227. Association for Computational Linguistics, 2007. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.128.6225.

[25] Merriam-Webster Online Dictionary. Allophone. http://www.merriam-webster.com/dictionary/allophone.

[26] M. Dong, K. Lua, and J. Xu. Selecting prosody parameters for unit selection based Chinese TTS. In K. Su, J. Tsujii, J. Lee, and O. Y. Kwong, editors, *Natural Language Processing – IJCNLP 2004*, volume 3248 of *Lecture Notes in Computer Science*, pages 272–279. Springer Berlin / Heidelberg, 2005. http://dx.doi.org/10.1007/978-3-540-30211-7_29.

[27] DSPRelated.com. What is a filter? http://www.dsprelated.com/dspbooks/filters/What_Filter.html.

[28] M. Elsabrouty. DMET 1003 lecture: Automatic speech recognition, 2006. http://cs.guc.edu.eg/courses/_Spring2008/DMET1003/slides/lecture12.pdf.

[29] Festvox. Limited domain synthesis. http://festvox.org/ldom/index.html.

[30] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren. *Darpa Timit: Acoustic-phonetic Continuous Speech Corps CD-ROM*. US Dept. of Commerce, National Institute of Standards and Technology, 1993.

[31] R. Garside, G. Leech, and G. Sampson. *The computational analysis of English: A corpus-based approach*. Longman, London, 1987.

[32] J. Giménez and L. Màrquez. Fast and Accurate Part-of-Speech Tagging: The SVM Approach Revisited. In *Proceedings of the 4th International Conference on Recent Advances in Natural Language Processing (RANLP)*, pages 158–165, 2003. http://www.lsi.upc.edu/~jgimenez/PUBS/ranlp2003-gm.pdf.

[33] J. Goldsmith. Ngram models and the sparsity problem. http://humanities.uchicago.edu/faculty/goldsmith/Industrial/Ngrams.ppt.

[34] S. Goldwater and T. Griffiths. A fully Bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751. Association for Computational Linguistics, 2007. http://cocosci.berkeley.edu/tom/papers/bhmm.pdf.

[35] B.B. Greene and G. M. Rubin. Automatic grammatical tagging of English. Providence RI: Department of Linguistics, Brown University, 1971.

[36] A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology of the North American Chapter of*

the Association of Computational Linguistics, pages 320–327. Association for Computational Linguistics, 2006. http://portal.acm.org/citation.cfm?id=1220876.

[37] P. Halácsy, A. Kornai, and C. Oravecz. HunPos - an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 209–212, Prague, Czech Republic, 2007. http://code.google.com/p/hunpos/.

[38] F. M. Hasan, N. UzZaman, and M. Khan. Comparison of unigram, bigram, HMM and Brill's POS tagging approaches for some South Asian languages, 2007. http://crblp.bracu.ac.bd/papers/POS_south_asian_clt07.pdf.

[39] B. Haselbach and U. Heid. The development of a morphosyntactic tagset for Afrikaans and its use with statistical tagging. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2010/pdf/318_Paper.pdf.

[40] D. Hindle. Acquiring disambiguation rules from text. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics*, 1989.

[41] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Pearson Education, second edition, 2009.

[42] S. King and V. Karaiskos. The blizzard challenge 2010. In *Proceedings of the Blizzard Challenge 2010 Workshop*, Kansai Science City, Japan, September 2010. http://festvox.org/blizzard/bc2010/summary_Blizzard2010.pdf.

[43] S. Klein and R. Simmons. A grammatical approach to grammatical coding of English words. *JACM 10*, pages 334–347, 1963.

[44] R. Koeling. Chunking with maximum entropy models. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*, pages 139–141. Association for Computational Linguistics, 2000. http://portal.acm.org/citation.cfm?id=1117634.

[45] D. Kumar and G. S. Josan. Part of speech taggers for morphologically rich Indian languages: A survey. *International Journal of Computer Applications (0975–8887)*, 6(5), September 2010. http://www.ijcaonline.org/volume6/number5/pxc3871409.pdf.

[46] G. Karthik Kumar, K. Sudheer, and P. V. S. Avinesh. Comparative study of various machine learning methods for Telugu part of speech tagging. In *Proceedings of the NLPAI Workshop on Machine Learning 2006 Contest*, 2006. http://ltrc.iiit.ac.in/nlpai_contest06/papers/indians.doc.

[47] C. Kuun. Lwazi project final report. Contract Report, 2009. http://www.meraka.org.za/lwazi/publications/CKuun09LwaziFinalRep.pdf.

[48] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289. Morgan Kaufmann Publishers Inc., 2001. http://www.cis.upenn.edu/~pereira/papers/crf.pdf.

[49] E. D. Liddy. *Encyclopedia of Library and Information Science*, chapter Natural Language Processing. Marcel Dekker, second edition, 2001. http://www.cnlp.org/publications/03NLP.LIS.Encyclopedia.pdf.

[50] Lions Voice Clinic. Anatomy 201: Anatomy and muscles of the larynx. http://www.lionsvoiceclinic.umn.edu/page2.htm#anatomy201.

[51] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, pages 285–318, 1988. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.130.9013.

[52] J. A. Louw. Speect: A multilingual text-to-speech system. In *Proceedings of the 19th Annual Symposium of the Pattern Recognition Association of South Africa (PRASA)*, pages 165–168, 2008. http://www.meraka.org.za/lwazi/publications/louw08speectmultilingual.pdf.

[53] R. Lowry. The Wilcoxon signed-rank test. http://faculty.vassar.edu/lowry/ch12a.html.

[54] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1994. http://www.ldc.upenn.edu/acl/J/J93/J93-2004.pdf.

[55] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997. http://portal.acm.org/citation.cfm?id=251709.

[56] S. Pilon. Outomatiese Afrikaanse woordsoortetikettering. Master's thesis, North-West University, 2005. http://hdl.handle.net/10394/726.

[57] Y. Qi, P. Kuksa, R. Collobert, K. Sadamasa, K. Kavukcuoglu, and J. Weston. Semi-supervised sequence labeling with self-learned features. In *ICDM '09: Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, pages 428–437. IEEE Computer Society, 2009. http://cs.nyu.edu/~koray/publis/qi-icdm-09.pdf.

[58] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[59] A. Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-96)*, Philadelphia, PA, 1996. http://www.aclweb.org/anthology/W/W96/W96-0213.pdf.

[60] Research Methods Knowledge Base. The t-test. http://www.socialresearchmethods.net/kb/stat_t.php.

[61] P. Roach. *English phonetics and phonology: a practical course.* Cambridge University Press, fourth edition, 2009.

[62] J. Romportl. Structural data-driven prosody model for TTS synthesis. In *Proceedings of the Speech Prosody 2006 Conference*, pages 549–552, 2006. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.60.9909.

[63] D. Roth and D. Zelenko. Part of speech tagging using a network of linear separators. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 1136–1142. Association for Computational Linguistics, 1998. http://www.aclweb.org/anthology/P/P98/P98-2186.pdf.

[64] Saarland University. The NEGRA corpus – a syntactically annotated corpus of German newspaper texts. http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/.

[65] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49, 1994. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.1139.

[66] H. Schütze. Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, pages 141–148. Morgan Kaufmann Publishers Inc., 1995. http://www.ldc.upenn.edu/acl/E/E95/E95-1020.pdf.

[67] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994. http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf.

[68] SIL International. LinguaLinks library: Comparison of morpheme-morph-allomorph and phoneme-phone-allophone. CD-ROM, 2003. http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/ComparisonOfMorphemeMorphAllom.htm.

[69] SIL International. LinguaLinks library: What is a lexicon? CD-ROM, 2003. http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsALexicon.htm.

[70] SIL International. LinguaLinks library: What is a morpheme? CD-ROM, 2003. http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsAMorpheme.htm.

[71] SIL International. LinguaLinks library: What is a phoneme? CD-ROM, 2003. http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsAPhoneme.htm.

[72] SIL International. LinguaLinks library: What is morphology? CD-ROM, 2003. http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsMorphology.htm.

[73] SIL International. LinguaLinks library: What is phonology? CD-ROM, 2003. http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsPhonology.htm.

[74] SIL International. LinguaLinks library: What is semantics? CD-ROM, 2003. http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsSemantics.htm.

[75] T. D. Singh and S. Bandyopadhyay. Manipuri POS tagging using CRF and SVM: A language independent approach. In *Proceedings of the 6th International Conference on Natural Language Processing*, 2008. http://ltrc.iiit.ac.in/icon2008-confsys/FILES/p95.pdf.

[76] J. O. Smith. Physical audio signal processing: Formant synthesis models. Online Book, December 2008. http://ccrma.stanford.edu/~jos/pasp/Formant_Synthesis_Models.html.

[77] J. O. Smith. Physical audio signal processing: Fundamental frequency estimation. Online Book, December 2008. http://ccrma.stanford.edu/~jos/pasp/Fundamental_Frequency_Estimation.html.

[78] N. A. Smith and J. Eisner. Contrastive estimation: training log-linear models on unlabeled data. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics, 2005. http://www.cs.cmu.edu/~nasmith/papers/smith+eisner.acl05.pdf.

[79] SoftVoice Inc. Formant synthesis. http://www.text2speech.com/formant.html.

[80] R. Sproat, A. W. Black, S. Chen, S. Kumar, M. Ostendorf, and C. Richards. Normalization of non-standard words. Article submitted to Computer Speech and Language, 2001. http://www.clsp.jhu.edu/people/skumar/pubs/nsw.pdf.

[81] Statistics Solutions. Mcnemar's test. http://www.statisticssolutions.com/methods-chapter/statistical-tests/mcnemar-test/.

[82] T. Styger and E. Keller. *Fundamentals of Speech Synthesis and Speech Recognition: Basic Concepts, State of the Art, and Future Challenges*, chapter Formant synthesis, pages 109–128. John Wiley and Sons, Inc., 1994. http://www.unil.ch/webdav/site/imm/users/ekeller/public/keller/Styger-Keller-94-FundVol.pdf.

[83] P. Taylor. *Text-to-Speech Synthesis*. Cambridge University Press, first edition, 2009.

[84] P. Taylor and A. W. Black. Assigning phrase breaks from part-of-speech sequences. *Computer Speech and Language*, 12:99–117, 1998. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.130.8423.

[85] The South African Department of Arts and Culture. Lwazi – a telephone-based, speech-driven information system. http://www.meraka.org.za/lwazi/.

[86] K. Tokuda, S. Sako, H. Zen, K. Oura, K. Nakamura, and K. Saino. The hts_engine API. http://hts-engine.sourceforge.net/.

[87] K. Tokuda, H. Zen, and A. W. Black. An HMM-based speech synthesis system applied to English. In *Proceedings of the 2002 IEEE Speech Synthesis Workshop*, 2002. http://www.sp.nitech.ac.jp/~tokuda/selected_pub/pdf/conference/tokuda_TTSworkshop2002.pdf.

[88] K. Toutanova and M. Johnson. A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Advances in Neural Information Processing Systems*, volume 20, pages 1521–1528. MIT Press, 2007. http://books.nips.cc/papers/files/nips20/NIPS2007_0964.pdf.

[89] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180. Association for Computational Linguistics, 2003. http://portal.acm.org/citation.cfm?id=1073445.1073478.

[90] University of Sheffield NLP Research Group. What is NLP. http://nlp.shef.ac.uk/.

[91] H. van Halteren, W. Daelemans, and J. Zavrel. Improving accuracy in word class tagging through the combination of machine learning systems. *Computational Linguistics*, 27(2):199–229, 2001. http://portal.acm.org/citation.cfm?id=972667.972669.

[92] D. R. van Niekerk and E. Barnard. Phonetic alignment for speech synthesis in under-resourced languages. In *Proceedings of the 10th Annual Conference of the International Speech Communication Association (Interspeech 2009)*, pages 880–883, Brighton, UK, 2009. http://www.meraka.org.za/lwazi/publications/vanniekerk09ttsalignment.pdf.

[93] A. Waibel. *Prosody and Speech Recognition*. Pitman Publishing, London, first edition, 1988.

[94] B. Wilson. Introduction to natural language processing. http://www.cse.unsw.edu.au/~billw/cs9414/notes/nlp/nlp-intro/nlp-intro-2007.html.

[95] D. Yarowsky. Homograph disambiguation in text-to-speech synthesis. *Computer Speech and Language*, 1996. http://www.cs.jhu.edu/~yarowsky/pubs/ssynth.ps.

[96] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Veltchev, and P. Woodland. *The HTK Book (for HTK Version 3.3)*. Cambridge University Engineering Department, 2005. http://htk.eng.cam.ac.uk/.

[97] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda. The HMM-based speech synthesis system version 2.0. In *Proceedings of ISCA SSW6*, pages 294–299, 2007. http://www.sp.nitech.ac.jp/~zen/english/index.php?plugin=attach&refer=Publications%2FInternational%20conferences&openfile=zen-ssw6.pdf.

[98] S. Zerbian and E. Barnard. Word-level prosody in Sotho-Tswana. *Speech Prosody 2010*, 100861:1–4, 2010. http://www.speechprosody2010.illinois.edu/papers/100861.pdf.

Appendix A

Part-of-Speech Tagsets

The English tagsets are listed in Table A.1.

Table A.1: English POS tagsets

eng.wsj1		**eng.wsj2**	
Tag	**Description**	**Tag**	**Description**
CC	Coordinating conjunction	CC	Coordinating conjunction
CD	Cardinal number	CD	Cardinal number
DT	Determiner	DT	Determiner
IN	Preposition or subordinating conjunction	IN	Preposition or subordinating conjunction
JJ	Adjective	JJ	Adjective
JJR	Adjective, comparative		
JJS	Adjective, superlative		
MD	Modal	MD	Modal
NN	Noun, singular or mass	NN	Noun
NNP	Proper noun, singular		
NNPS	Proper noun, plural		
NNS	Noun, plural		
PRP	Personal pronoun	PRP	Pronoun
PRP$	Possessive pronoun		
RB	Adverb	RB	Adverb
RBR	Adverb, comparative		
RBS	Adverb, superlative		
TO	*to*	TO	*to*
VB	Verb, base form	VB	Verb
VBD	Verb, past tense		
VBG	Verb, gerund or present participle		
VBN	Verb, past participle		
VBP	Verb, non-3rd person singular present		
VBZ	Verb, 3rd person singular present		
WDT	Wh-determiner	W	Wh-word
WP	Wh-pronoun		
WP$	Possessive wh-pronoun		
WRB	Wh-adverb		
EX	Existential *there*		
FW	Foreign word		
LS	List item marker		

eng.wsj1 (Continued)		eng.wsj2 (Continued)	
Tag	**Description**	**Tag**	**Description**
PDT	Predeterminer		
POS	Possessive ending		
RP	Particle	O	Other
SYM	Symbol		
UH	Interjection		
#	Hash		
-NONE-	Non-linguistic mark		
$	Dollar	$	Dollar
-LRB-	Left round bracket	ZL	Opening punctuation
“	Opening quotation mark		
-RRB-	Right round bracket	ZR	Closing punctuation
”	Closing quotation mark		
,	Comma	Z	Mid-sentence punctuation
:	Colon		
.	Full-stop	.	Full-stop

The Afrikaans tagsets are listed in Table A.2. The descriptions are taken as-is from [56]. “???” indicates a tag appearing in the data, yet not described in the original work. The tag name is most likely an orthographic error.

Table A.2: Afrikaans POS tagsets

afr.nwu1		afr.nwu2	
Tag	**Description**	**Tag**	**Description**
AOA	Adjektief: oortreffend/attributief		
AOP	Adjektief: oortreffend/predikatief		
ASA	Adjektief: stellend/attributief	A	Adjektief
ASP	Adjektief: stellend/predikatief		
AVA	Adjektief: vergrotend/attributief		
AVP	Adjektief: vergrotend/predikatief		
BOM	Bywoord: oortreffend/maat		
BOW	Bywoord: oortreffend/ wyse		
BSD	Bywoord: stellend/modaliteit		
BSF	Bywoord: stellend/frekwensie		
BSG	Bywoord: stellend/graad		
BSM	Bywoord: stellend/maat		
BSO	Bywoord: stellend/ontkenning		
BSP	Bywoord: stellend/plek		
BSR	Bywoord: stellend/rigting	B	Bywoord
BSS	Bywoord: stellend/skakeling		
BST	Bywoord: stellend/tyd		
BSW	Bywoord: stellend/wyse		
BVG	Bywoord: ???		
BVM	Bywoord: vergrotend/maat		
BVP	Bywoord: ???		
BVT	Bywoord: ???		
BVW	Bywoord: vergrotend/ wyse		
KN	Konjunk: neweskikkend	K	Konjunk

afr.nwu1 (Continued)		afr.nwu2 (Continued)	
Tag	**Description**	**Tag**	**Description**
KO	Konjunk: onderskikkend		
LB	Lidwoord: bepaald	L	Lidwoord
LO	Lidwoord: onbepaald		
NA	Naamwoord: Abstrak	N	Naamwoord
NEE	Naamwoord: eienaam/enkelvoud/basis		
NEM	Naamwoord: eienaam/meervoud/basis		
NLW	Naamwoord: ???		
NM	Naamwoord: Massanaam		
NME	Naamwoord: maatnaam/enkelvoud/basis		
NMED	Naamwoord: maatnaam/enkelvoud/diminutief		
NSE	Naamwoord: soortnaam/enkelvoud/basis		
NSED	Naamwoord: soortnaam/enkelvoud/diminutief		
NSM	Naamwoord: soortnaam/meervoud/basis		
NSMD	Naamwoord: soortnaam/meervoud/diminutief		
NVE	Naamwoord: versamelnaam/enkelvoud/basis		
NVED	Naamwoord: versamelnaam/enkelvoud/diminutief		
NVM	Naamwoord: versamelnaam/meervoud/basis		
PA	Voornaamwoord: Aanwysend	P	Voornaamwoord
PB	Voornaamwoord: betreklik		
PDHEB	Voornaamwoord: derde/manlik/enkelvoud/besitlik		
PDHEDP	Voornaamwoord: derde/manlik/enkelvoud/gemarkeerd/persoonlik		
PDHENP	Voornaamwoord: derde/manlik/enkelvoud/ongemarkeerd/persoonlik		
PDHEW	Voornaamwoord: derde/manlik/enkelvoud/wederkerend		
PDMB	Voornaamwoord: derde/meervoud/besitlik		
PDMP	Voornaamwoord: derde/meervoud/persoonlik		
PDMW	Voornaamwoord: derde/meervoud/wederkerend		
PDOENP	Voornaamwoord: derde/onsydig/enkelvoud/ongemarkeerd/persoonlik		
PDOEP	Voornaamwoord: ???		
PDOEW	Voornaamwoord: derde/onsydig/enkelvoud/wederkerend		
PDVEB	Voornaamwoord: derde/vroulik/enkelvoud/besitlik		
PDVEDP	Voornaamwoord: derde/vroulik/enkelvoud/gemarkeerd/persoonlik		
PDVENP	Voornaamwoord: derde/vroulik/enkelvoud/ongemarkeerd/persoonlik		
PDVEW	Voornaamwoord: derde/vroulik/enkelvoud/wederkerend		
PEEB	Voornaamwoord: eerste/enkelvoud/besitlik		
PEEDP	Voornaamwoord: eerste/enkelvoud/gemarkeerd/persoonlik		
PEENP	Voornaamwoord: eerste/enkelvoud/ongemarkeerd/persoonlik		
PEEW	Voornaamwoord: eerste/enkelvoud/wederkerend		
PEMB	Voornaamwoord: eerste/meervoud/besitlik		
PEMNP	Voornaamwoord: eerste/meervoud/persoonlik		
PEMP	Voornaamwoord: ???		
PEMW	Voornaamwoord: eerste/meervoud/wederkerend		
PO	Voornaamwoord: onbepaald		
PR	Voornaamwoord: ???		
PTEB	Voornaamwoord: tweede/enkelvoud/besitlik		
PTEDP	Voornaamwoord: tweede/enkelvoud/gemarkeerd /persoonlik		
PTENP	Voornaamwoord: tweede/enkelvoud/ongemarkeerd /persoonlik		
PTEW	Voornaamwoord: tweede/enkelvoud/wederkerend		
PTMB	Voornaamwoord: tweede/meervoud/besitlik		
PTMNP	Voornaamwoord: tweede/meervoud/persoonlik		

afr.nwu1 (Continued)		afr.nwu2 (Continued)	
Tag	**Description**	**Tag**	**Description**
PTMW	Voornaamwoord: ???		
PX	Voornaamwoord: ???		
PV	Voornaamwoord: Vraend	W	Wh-woord
PW	Voornaamwoord: wederkerig		
RSE	Residu: simbool/enkelvoud	RSE	Simbool
SVS	Setsel: voorsetsel	SVS	Voorsetsel
THAB	Telwoord: hooftelwoord/adjektief/bepaald	T	Telwoord
THAO	Telwoord: hooftelwoord/adjektief/onbepaald		
THBB	Telwoord: hooftelwoord/bywoord/bepaald		
THBO	Telwoord: hooftelwoord/bywoord/onbepaald		
THNB	Telwoord: hooftelwoord/naamwoord/bepaald		
THPB	Telwoord: hooftelwoord/voornaamwoord/bepaald		
THPO	Telwoord: hooftelwoord/voornaamwoord/onbepaald		
TRAB	Telwoord: rangtelwoord/adjektief/bepaald		
TRAO	Telwoord: rangtelwoord/adjektief/onbepaald		
TRBB	Telwoord: rangtelwoord/bywoord/bepaald		
TRNB	Telwoord: ???		
TRPB	Telwoord: rangtelwoord/voornaamwoord/bepaald		
VNAHG	Werkwoord: ???	V	Werkwoord
VTHOG	Werkwoord: teenwoordig/hoof/onskeibaar/oorganklik		
VTHOK	Werkwoord: teenwoordig/hoof/onskeibaar/koppel		
VTHOO	Werkwoord: teenwoordig/hoof/onskeibaar/onoorganklik		
VTHOV	Werkwoord: teenwoordig/hoof/onskeibaar/voorsetsel		
VTHSG	Werkwoord: teenwoordig/hoof/skeibaar/oorganklik		
VTHSK	Werkwoord: ???		
VTHSO	Werkwoord: teenwoordig/hoof/skeibaar/onoorganklik		
VVHOG	Werkwoord: verlede/hoof/onskeibaar/oorganklik		
VVHOK	Werkwoord: verlede/hoof/onskeibaar/koppel		
VVHOO	Werkwoord: verlede/hoof/onskeibaar/onoorganklik		
VVHOV	Werkwoord: verlede/hoof/onskeibaar/voorsetsel		
VOUT	Werkwoord: ???	VU	Hulpwerkwoord
VTUOA	Werkwoord: teenwoordig/hulp/onskeibaar/aspek		
VTUOM	Werkwoord: teenwoordig/hulp/onskeibaar/modaliteit		
VTUOP	Werkwoord: teenwoordig/hulp/onskeibaar/modus		
VUOT	Werkwoord: hulp/onskeibaar/tyd		
VVUOA	Werkwoord: verlede/hulp/onskeibaar/aspek		
VVUOM	Werkwoord: verlede/hulp/onskeibaar/modaliteit		
VVUOP	Werkwoord: verlede/hulp/onskeibaar/modus		
RAE	Residu: afkorting/enkelvoud		
RAM	Residu: afkorting/meervoud		
RFE	Residu: formule/enkelvoud		
RKE	Residu: akroniem/enkelvoud		
RLE	Residu: letternaamwoord/enkelvoud		
RLM	Residu: letternaamwoord/meervoud		
RO	Residu: ongeklassifiseerd		
RSF	Residu: suffiks		
RVE	Residu: vreemdetaalwoord/enkelvoud		
RVM	Residu: vreemdetaalwoord/meervoud		

afr.nwu1 (Continued)		afr.nwu2 (Continued)	
Tag	**Description**	**Tag**	**Description**
RWD	Residu: woorddeel		
UDS	Uniek: dis	O	Ander
UE	Uniek: enklities		
UPB	Uniek: partikel/betreklik		
UPD	Uniek: partikel/deel		
UPE	Uniek: partikel/skakel		
UPG	Uniek: partikel/graad		
UPI	Uniek: partikel/infinitief		
UPO	Uniek: partikel/ontkenning		
UPS	Uniek: partikel/genitief		
UPV	Uniek: partikel/vergelyking		
UPW	Uniek: partikel/ww.		
UXD	Uniek: eksistensile daar		
W	Tussenwerpsel		
ZE	Punktuasie: sinseinde	ZE	Sinseinde punk.
ZM	Punktuasie: sinsmiddel	ZM	Sinsmiddel punk.
ZPL	Punktuasie: links-parentese	ZPL	Links-parentese
ZPR	Punktuasie: regs-parentese	ZPR	Regs-parentese

Appendix B

Tables of Significance

Table B.1: t-test table

df	**0.10**	**0.05**	**0.025**	**0.01**	**0.005**	**0.001**
1	3.078	6.314	12.706	31.821	63.657	318.313
2	1.886	2.920	4.303	6.965	9.925	22.327
3	1.638	2.353	3.182	4.541	5.841	10.215
4	1.533	2.132	2.776	3.747	4.604	7.173
5	1.476	2.015	2.571	3.365	4.032	5.893
6	1.440	1.943	2.447	3.143	3.707	5.208
7	1.415	1.895	2.365	2.998	3.499	4.782
8	1.397	1.860	2.306	2.896	3.355	4.499
9	1.383	1.833	2.262	2.821	3.250	4.296
10	1.372	1.812	2.228	2.764	3.169	4.143
11	1.363	1.796	2.201	2.718	3.106	4.024
12	1.356	1.782	2.179	2.681	3.055	3.929
13	1.350	1.771	2.160	2.650	3.012	3.852
14	1.345	1.761	2.145	2.624	2.977	3.787
15	1.341	1.753	2.131	2.602	2.947	3.733
16	1.337	1.746	2.120	2.583	2.921	3.686
17	1.333	1.740	2.110	2.567	2.898	3.646
18	1.330	1.734	2.101	2.552	2.878	3.610
19	1.328	1.729	2.093	2.539	2.861	3.579
20	1.325	1.725	2.086	2.528	2.845	3.552
30	1.310	1.697	2.042	2.457	2.750	3.385
40	1.303	1.684	2.021	2.423	2.704	3.307
50	1.299	1.676	2.009	2.403	2.678	3.261
60	1.296	1.671	2.000	2.390	2.660	3.232
70	1.294	1.667	1.994	2.381	2.648	3.211
80	1.292	1.664	1.990	2.374	2.639	3.195
90	1.291	1.662	1.987	2.368	2.632	3.183
100	1.290	1.660	1.984	2.364	2.626	3.174
∞	1.282	1.645	1.960	2.326	2.576	3.090

Table B.2: Chi-square table

df	**0.995**	**0.99**	**0.975**	**0.95**	**0.90**	**0.10**	**0.05**	**0.025**	**0.01**	**0.005**
1	–	–	0.001	0.004	0.016	2.706	3.841	5.024	6.635	7.879
2	0.010	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210	10.597
3	0.072	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345	12.838
4	0.207	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277	14.860
5	0.412	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086	16.750
6	0.676	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548
7	0.989	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475	20.278
8	1.344	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090	21.955
9	1.735	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209	25.188
11	2.603	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725	26.757
12	3.074	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217	28.300
13	3.565	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688	29.819
14	4.075	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141	31.319
15	4.601	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578	32.801
16	5.142	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000	34.267
17	5.697	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409	35.718
18	6.265	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805	37.156
19	6.844	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191	38.582
20	7.434	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566	39.997
30	13.787	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892	53.672
40	20.707	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691	66.766
50	27.991	29.707	32.357	34.764	37.689	63.167	67.505	71.420	76.154	79.490
60	35.534	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379	91.952
70	43.275	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425	104.215
80	51.172	53.540	57.153	60.391	64.278	96.578	101.879	106.629	112.329	116.321
90	59.196	61.754	65.647	69.126	73.291	107.565	113.145	118.136	124.116	128.299
100	67.328	70.065	74.222	77.929	82.358	118.498	124.342	129.561	135.807	140.169