UNIVERSITÉ DE FRANCHE-COMTÉ
CENTRE LUCIEN TESNIÈRE
FRANCE

THE UNIVERSITY OF WOLVERHAMPTON
SCHOOL OF LAW, SOCIAL SCIENCES AND
COMMUNICATIONS
UNITED KINGDOM

Binyam Gebrekidan Gebre

Part of Speech Tagging for Amharic

A Project submitted as part of a programme of study for the award of MA
Natural Language Processing & Human Language Technology

Supervisors:

Prof. Sylviane Cardey
Prof. Ruslan Mitkov

June 2010

Declaration Form

UNIVERSITY OF WOLVERHAMPTON
SCHOOL OF LAW, SOCIAL SCIENCES AND COMMUNICATIONS
MA NATURAL LANGUAGE PROCESSING & HUMAN LANGUAGE TECHNOLOGY

Name: _____

Date: _____

Title: _____

Module Code: _____

Presented in partial fulfilment of the assessment requirements for the above award.

Supervisor: _____

Declaration:

i. This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. Save for any express acknowledgements, references and/or bibliographies cited in the work, I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

ii. It is acknowledged that the author of any project work shall own the copyright. However, by submitting such copyright work for assessment, the author grants to the University a perpetual royalty-free licence to do all or any of those things referred to in section 16(i) of the Copyright Designs and Patents Act 1988 (viz: to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make adaptation of the work.

iii. This project did not involve contact with human subjects, and hence did not require approval from the LSSC Ethics Committee.

Signed: Date:

Abstract

Amharic, the second most spoken Semitic language, is a written language that poses its own challenges to natural language processing. One basic NLP task is part of speech tagging, which is the process of assigning tags to words in text. Tags can be as simple as noun, verb, adjective, etc or as complex as noun singular feminine, verb past tense third person, etc. POS tagging is not useful by itself but is generally accepted to be the first step to understanding a natural language. Most NLP tasks and applications including parsing, information extraction, machine translation, speech synthesis/recognition heavily depend on it.

Ambiguity is the reason POS tagging is not an easy problem. All languages have some form of ambiguities. Resolving ambiguities requires efficient and accurate methods. Previous attempts of POS tagging Amharic texts resulted in performances worse than those reported for Arabic, Hebrew and English.

In this dissertation, theoretical and practical POS tagging issues have been discussed with the view to improving POS tagging performance for Amharic, which was never above 90%. Knowledge of Amharic morphology, the given annotated data and the tagging algorithms have been examined and shown to play critical roles in the final performance result. With the experiments carried out using state-of-the-art machine learning algorithms, POS tagging accuracies for Amharic have crossed above the 90% limit for the first time.

The reasons for such relatively higher performance have come from three factors: usage of partially cleaned version of a corpus, good feature selection, resulting from morphological study of the language, and parameter tuning, resulting from understanding the tagging algorithms and experimenting with them.

Key words: Language, Semitic, Amharic, Part of speech, POS, tagging, HMM, CRF, SVM, Brill, TnT, NLTK

Acknowledgements

This dissertation would not have been initiated and completed without the constant support, guidance and encouragement of my co-supervisors: Professor Sylviane Cardey and Prof. Ruslan Mitkov, who are both renowned experts in natural language processing research areas. In the first year of my research in France, Prof. Madame Cardey helped me in choosing and formulating my topic. She helped me focus and structure my dissertation. In the second year of the research in the United Kingdom, Prof. Ruslan Mitkov encouraged me to attend many seminars which he made it possible so that I deepen my research knowledge and despite his busy schedule, he also periodically checked my progress and gave me guidance to make sure that I did my best. I am forever grateful to both of my supervisors.

Special thanks also go to Prof. Peter Greenfield, Prof. Izabella Thomas, Prof. Henri Madec, Dr. Lucia Specia and Dr. Constantin Orasan for contributing directly or indirectly to this research. Gabriel Sekunda and Dilber DeVitre also deserve special mentions for being kind and helpful to me in administrative and other matters.

I am also grateful to my classmates and friends whom I met both in France and the United Kingdom for making the two-year masters programme more illuminating and more enjoyable. My classmates were international with diverse nationalities and cultures who showed me that human groups are more similar than history depicts it.

Last but not least, I would like to thank the European Union. This project was supported by the European Commission, Education & Training, Erasmus Mundus: EMMC 2008-0083, Erasmus Mundus Masters in NLP & HLT programme.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

- Background
- Research Focus and Research Objectives
- Value of this Research
- Organization of the Dissertation

1.1 Background

With the ever increasing availability of information and knowledge in many languages and with the ever increasing interaction between cultures, the need for language technologies is becoming more necessary than ever before. Natural language processing is a multidisciplinary area where the goal is to design and build software that will analyze, understand and generate all languages that humans use naturally. This exciting area has long been the center of attention of some researchers for a long time now. In fact, one of the earliest applications for computers was machine translation, translating from one language into another by computers. Recent applications cover other aspects of natural language processing including speech recognition/synthesis, information extraction/retrieval, question answering and other emerging applications.

Much of the research in natural language processing has been dedicated to resource-rich languages like English, French and other major European and Asian languages. African languages have, however, received far too little attention. In fact, most are being spoken by less and less people. One exception that is seeing

an increase in use and number of speakers is Amharic, a language that is mainly spoken in Ethiopia. Currently, it has an estimated 30 million speakers(Gambäck et al., 2009), which puts it in second position as the most spoken Semitic language in the world (after Arabic). Amharic is also spoken in Egypt, Israel, and Sweden by some 2.7 million emigrants (Wikipedia, 2010).

The number of speakers of the language is on the rise for two reasons. First, it is the working language of the federal democratic republic of Ethiopia, a country with more than 85 million people (CIA, 2010). Second, unlike most other African languages, Amharic is a written language with its own alphabet and written materials, actively being used everyday.

However, even under these favorable conditions, Amharic has been one of the under-resourced languages both in terms of electronic resources and processing tools. Recently, however, there have been independent attempts to develop them. One outcome of such an attempt is the publicly available medium-sized part-of-speech-tagged news corpus (Demeke and Getachew, 2006) and a morphological analyzer (Gasser, 2009*b*). The availability of these resources has encouraged researchers to process the language by using and applying different NLP models that have proven effective for analyzing English and other most-studied languages.

One basic task in natural language processing is part-of-speech tagging or POS tagging for short. It is the process of assigning a part-of-speech tag like noun, verb, pronoun, preposition, adverb, adjective or other lexical class markers to each word in a text. POS tagging is not useful by itself but it is generally accepted to be the first step to understanding a natural language. Most other tasks and applications heavily depend on it.

In addition to that, any NLP problem can be reduced to a tagging problem and so POS tagging serves as a prototype problem. For example, machine translation can be seen as the tagging of words in a given language by words of another language; speech recognition can be seen as the tagging of signals by letters and so on. In general, the input-output relationship can be as complex as sequences, sets, trees and others that can be imagined. POS tagging represents the simplest of these problems.

At first thought, the solution to this POS tagging problem may seem trivial, but it is actually a very hard problem. There is no known method that solves the problem with complete accuracy for any language. The reason for this is partly related to inconsistencies of our understanding of categories of words. Even trained human annotators do not agree as to the category of a word 3-4% of the

times(Marcus et al., 1993). The other reason arises from language ambiguities and the ineffectiveness of the resolving methods.

Language expressions are ambiguous and computers do not have the commonsense and the world knowledge that humans have when they communicate. For example, **I made her duck** has at least 5 meanings (Jurafsky et al., 2000)

1. I cooked waterfowl for her.
2. I cooked waterfowl belonging to her.
3. I created the (plaster?) duck she owns.
4. I caused her to quickly lower her head or body.
5. I waved my magic wand and turned her into undifferentiated waterfowl.

These different meanings are caused by a number of ambiguities. The first one is part of speech ambiguity. **Duck** can be a verb or a noun and **her** can be a dative pronoun or a possessive adjective. The other ambiguities are related to semantics and syntax (ie make can mean cook or create and it can take one or two arguments).

To a human being, the intended meaning of the above sentence is clear depending on the circumstances but for a computer it is far from obvious. Therefore, the purpose of tagging is to give the computer as much information and knowledge as necessary to enable it to assign each word the correct tag as used in the given context.

Tags are designed to be more abstract representation of words. A set of tags (tagset) is chosen and designed to give linguistic information about words. Besides categorizing words into major classes (nouns, verbs, etc), the linguistic information may include distinctions between verbs in different tenses and between nouns with different genders and numbers (singular/plural), etc. In any case, the nature and number of tags are debatable as they are highly dependent on the purpose and the nature of the given language. Designing a tagset by itself is a research problem which requires its own careful analysis.

In POS tagging problems, it is usually assumed that the tags have already been designed. Given a finite tagset for a given language, the computational problem is to map words of a given text to their correct tags based on context and lexical information. In other words, the infinite words are classified into finite tags.

There are three approaches to solving this classification problem based on two fundamental concepts- rules and statistics. Rule-based taggers use handcrafted linguistically-motivated rules. Stochastic taggers, by contrast, use probabilistic mathematical models and a corpus. The third approach combines the best of both concepts. None of them is perfect for all languages and for all purposes. The relevance and effectiveness of each approach depends on the purpose and the given language.

This dissertation explores some selected tagging methods and tests their effectiveness on Amharic, a morphologically-rich language.

1.2 Research Focus and Research Objectives

The objective of this dissertation is to develop a part of speech tagger for Amharic. This task entails the examination of Amharic morphology from computational point of view to identify the word and context features that can be used to form the features that are required in the state-of-the-art POS tagging systems.

The goals of the dissertation are:

1. Develop a tokenizer and a tagger for Amharic
2. Explore state-of-the-art tagging methods
3. Examine Amharic morphology
4. Implement and adapt the tagging methods on Amharic
5. Evaluate and report results

1.3 Value of this Research

This research advances the knowledge of NLP issues in Amharic and hence Semitic languages. The challenges faced and the experiments done in this dissertation will highlight future directions to take in computational studies of and resource development for under-resourced languages in general and Semitic languages in particular. This dissertation also highlights some of what needs to be done in developing language independent tools and applications. For example, vowel

patterns embedded in Amharic roots have special functions that could be useful in syntax and semantics processing, but these patterns cannot be easily extracted with existing language independent tools.

1.4 Organization of the Dissertation

The first chapter attempts to motivate the topic of part of speech tagging for Amharic. It also outlines the goals to be achieved and the value it is expected to contribute to NLP research.

Chapter 2 discusses previous research work in the area of POS tagging for Semitic languages and English. It discusses the different tagging approaches in the literature and reports the best results for English, Arabic, Hebrew and Amharic.

Chapter 3 presents and discusses the mathematics behind the tagging models. It discusses briefly the mathematics in the selected widely used machine learning algorithms, namely Hidden Markov Models, Conditional Random Fields, Support Vector Machines and Transformation-based Error-driven Learning.

Chapter 4 examines Amharic orthography, morphology and discusses the challenges and opportunities they present in natural language processing. Particularly, the vowel patterns are extensively covered to motivate their use as features in the machine learning algorithms covered in chapter 3.

Chapter 5 attempts to address the issues in the POS tagging process. It starts by presenting and evaluating the ELRC tagset and the POS tagged corpus (WIC). It discusses the errors and inconsistencies in the corpus and presents the corrections made. It also discusses the steps in the tagging process from tokenization, feature extraction to disambiguation. Tools and programming language used are also briefly presented.

Chapter 6 presents and analyzes the results using standard evaluation methods widely used in NLP. Specifically, accuracy, precision, recall, f-measure and confusion matrix are the metrics or the methods used to report and analyze the results obtained in chapter 6.

Chapter 7 provides the conclusion to the dissertation with summaries of results, limitations and recommendations.

Chapter 2

Literature Review

- Introduction
- Rule-based Taggers
- Machine Learning-based Taggers
- Hybrid Taggers
- Work on Related Languages
- Previous Work on Amharic

2.1 Introduction

There is much literature in the issues surrounding part of speech tagging for most major languages. It has been the topic of many publications including masters and PhD dissertations. These research works, by their presence, show how important and difficult POS tagging is and they also show that there is no one method that works for all languages. Every human language family poses its challenges and requires specific methods. The major issues surrounding POS tagging have been the following:

- Resources
- Tokenization
- Tagset design

- Tagging algorithms
- Evaluation
- Problem areas
  - Ambiguity
  - Unknown words
  - Proper nouns (in languages where there is no concept of capitalization)

Some of the POS tagging challenges posed by the above issues are shared by most languages; others are specific to given one or more languages. Some issues are resolved with simple methods for some languages, but with more sophisticated methods for other languages. For example, with respect to resources, English has huge labeled/unlabeled electronic corpora - and so tagging methods take advantage of that. Amharic, on the other hand, has a limited resource and so tagging methods should be designed to overcome the lack of it. However, even with comparable resources, languages still pose challenges in many other ways. Some are morphologically or syntactically more complex than others. Some have writing systems which make automatic identification of word boundaries more difficult. These differences in resources and in nature have motivated researchers to design and experiment with different tagging methods and have reported results with different accuracies.

The highest tagging accuracies reported so far are for English. In fact, POS tagging is generally considered to be a solved problem for English. Accuracies have reached around 97%. For other languages especially those less-resourced ones, there is an unsatisfied need for new or adapted methods. This is partly the reason why much of recent research into POS tagging has been dedicated to developing or adapting existing methods for such languages.

Even though the literature of POS tagging is full of many methods, they all fall into three categories - rule-based, stochastic and hybrid taggers. Methods that rely on handcrafted language dependent rules are classified as rule-based taggers whereas methods that use probabilistic mathematical models and a corpus are classified as stochastic taggers. Methods that combine the best of both probabilistic and rule-based approaches are classified as hybrid taggers. All these methods differ in design and how they handle the major issues in tagging, which are usually lack of linguistic resources, ambiguity, unknown word and sometimes proper names. In this chapter, we will discuss the major tagging methods and some results for English and Semitic languages.

2.2 Rule-based Taggers

Rule-based taggers generally depend on rules hand-written by humans. The first rule-based taggers used a two-stage architecture (Greene and Rubin, 1971). In the first stage, a dictionary is used to assign a set of possible parts of speech to a particular instance of a word. This dictionary does not contain all word variants, but only the word stems. This means that some morphological analysis is done before accessing the dictionary. In the second stage, a set of disambiguating rules is applied to narrow down the choice of tags to just one in cases where words receive multiple parts of speech.

The first rule-based tagger based on the above two-level architecture is called TAGGIT (Greene and Rubin, 1971), which was developed on the Brown corpus[1]. The tags assigned were from a set of some 77 tags (the Brown tagset). Its basic idea is to associate with each word a set of potential tags, and then use the context to choose the correct one. The mechanism for the initial assignment of tags to a word relied on a lexicon, a word-ending list, and a set of other rules for dealing with capitalized words, etc. This tagger achieved an accuracy of 70%.

Similarly, the ENGTWOL rule-based tagger (Voutilainen, 1995) is based on the same two-stage architecture. It employs 139 tags and 56,000 entries for English word stems. The lexicon includes separate entries for a word with multiple parts of speech but does not include inflected and derived forms. For the second stage, it applies more than 1000 constraint rules to disambiguate words which receive multiple parts of speech.

The above two-stage architecture is the most common approach but not the only one. A French rule-based tagger called Labelgram (Cardey and Greenfield, 2003) applies only disambiguating rules based on syntax and word endings to tag French words without using a dictionary. This method has been successfully applied for English and Spanish (Birocheau, 2003; Morgadinho, 2004).

2.3 Machine Learning-based Taggers

The second approach to POS tagging is based on machine learning techniques. The general idea in this approach is to calculate corpus-learned probabilities for all tag sequences for a given sequence of words and then choose the sequence

[1]The Brown corpus has 1 million English words

with the highest probability. Such stochastic methods have been used since the 1960s (Stolz et al., 1965; Bahl and Mercer, 1976; Marshall, 1983; Garside, 1987; Church, 1988; Derose, 1988) and many others. If the corpus used to calculate the probabilities is labeled (ie if a large text is available where every word is tagged with its corresponding part of speech as used in the text), then the algorithm used to develop the tagging model is called a supervised machine learning algorithm. If, on the other hand, only unlabeled corpus is available (ie a text with no POS tags for each word), then unsupervised machine learning algorithms are used. Algorithms that use small labeled corpus and then large unlabelled corpus are called semi-supervised algorithms.

All of these different classes of learning algorithms have been applied to part of speech tagging. Specifically, the most common taggers use variations of Hidden Markov Models, Maximum Entropy, Conditional Random Fields and Support Vector Machines. These methods have been shown to perform with the highest accuracies. Table 2.1 shows the state-of-the-art POS tagging accuracy results for English tested on the Wall Street Journal corpus.

Memory based learning algorithms have also been applied to POS tagging with accuracies comparable to the state-of-the-art (Daelemans et al., 1996; 1998; Zavrel and Daelemans, 1999). Combining taggers with the objective of outperforming the best tagger has been tried too(Brill and Wu, 1998; Aires et al., 2000; Halteren et al., 2001; De Pauw et al., 2006; Sjöbergh, 2003; Loftsson, 2006; Shacham, 2007; Spoustová et al., 2007). With few exceptions, the combined taggers have outperformed their best component.

Advantages of machine learning-based taggers include ease of repeatability of experiments using different algorithms and the ability to learn rules or patterns that may escape human ingenuity. Their disadvantages are that they usually require large annotated data, which is expensive and laborious to build.

2.4 Hybrid Taggers

Hybrid taggers combine the best of both probabilistic and rule-based methods. These methods are also called transformation-based taggers. A relatively recent successful transformation based tagger is Brill's tagger (Brill, 1992). Like the rule-based taggers, words are assigned tags based on a set of disambiguating rules and like the stochastic taggers, these rules are automatically learned from pre-labeled data.

Table 2.1: State-of-the-art POS Tagging Accuracies for English

System name	Short description	Unknown	Overall
TnT	Hmm-based (Brants, 2000)	85.86	96.46
GENiA Tagger	Maximum entropy cyclic dependency network (Tsuruoka et al., 2005)	-	97.05
Averaged Perceptron	Theory and Experiments with Perceptron Algorithms, Collins (2002)	-	97.11
Maxent easiest-first	Maximum entropy bidirectional easiest-first inference (Tsuruoka, 2005)	-	97.15
SVMTool	SVM-based tagger and tagger generator(Giménez and Marquez, 2004)	89.01	97.16
Stanford Tagger 1.0	Maximum entropy cyclic dependency network (Toutanova et al., 2003)	89.04	97.24
LTAG-spinal	Bidirectional perceptron learning (Shen et al., 2007)	-	97.33

The Brill tagger has two stages: in the first stage, the tagger gives the most common tag to each known word (without context). Capitalized unknown words are tagged as nouns and the non-capitalized unknown words are given the most common tag based on affix and other lexical cues. In the second stage, the tagger iteratively learns the most effective rules to correct errors, thereby incrementally improving its performance. The resulting rules can then be applied to a new corpus after passing it through the baseline tagging (i.e. after assigning the most frequent tag for each word).

An experiment on 1.1 million words of the Penn Treebank Wall Street Journal showed that this transformation-based tagger can achieve an overall tagging accuracy of 96.6% (Brill, 1995) using 690 transformation rules learned from 950k words (86.3%). 447 of the 690 rules were contextual rules learned from 600k words and the rest 243 rules were learned from 350k words and were used to tag unknown words. The tagging accuracy for unknown words is 82%.

Another hybrid tagger is the CLAWS tagger, which started in the 1980s and after a number of modifications, the latest version is now called CLAWS4 (Garside and Smith, 1997). CLAWS (the Constituent Likelihood Automatic Word-tagging System) is used to POS tag 100 million words of the British National Corpus (BNC), which consists of words of English written texts and spoken transcriptions, sampled from a comprehensive range of text genres. Two tagsets are used

- C7: A detailed tagset of 146 tags
- C5: A less refined tagset of 61 tags

As a hybrid system, the CLAWS4 tagger has both probabilistic and rule-based components. In the rule-based part, a number of tests are applied to assign potential parts of speech tags for a given word. These tests are 8 in total and include looking up the word in a lexicon to assign potential parts of speech. It is important to note at this point that these rules are different from Brill's rules as the latter are not learned automatically but engineered by humans.

The probabilistic component of CLAWS4 is a variation of an HMM tagger, where the goal is to choose the tag sequence that maximizes $P(w/t)p(t/t')$. The result of applying the rules to each word in the text is that each word receives one or more part-of-speech tags. If a given word receives multiple tags, then the probabilistic component is applied to choose a single tag. CLAWS4 operates with an accuracy rate of some 96-97 percent across the whole range of texts in the BNC.

2.5 Work on Related Languages

Amharic belongs to the Semitic family of languages which include Arabic, Hebrew, Tigrigna, Maltese and many others. These languages share a number of common characteristics and so experiences and results obtained in NLP tasks for one of these languages can be useful for the others. With respect to POS tagging, these languages share common problems.

The first common problem arises from their nature of word formation and the writing systems. The major Semitic languages have been written languages for a long time now. Their writing systems allow words to be delimited by space. However, the words in these languages are different from those of English. They are formed by the concatenation of lexical units, most of which may belong to various word classes (POS). In other words, two or more words in English can be considered as one word in Semitic languages.

For that reason, the process of POS tagging for these languages is more complicated. It is not very clear from the beginning what the POS tagging units should be as the words are the concatenation of various morphemes with potential boundary ambiguities. Should the words as they appear in text (separated by space) be POS tagged or the morphemes? The answer to that question de-

termines the tokenization algorithm and the tagset design. Both extreme and intermediate approaches have been tried for Hebrew (Bar-Haim et al., 2005) as is discussed in section 2.5.2.

The second potential POS tagging problem in Semitic languages comes from the nature of their writing systems. For example, capital letters are non-existent in most of these languages making the task of identifying proper nouns more difficult. Arabic and Hebrew also have another problem. They leave out short vowels and write only consonants and long vowels expecting readers to fill out the missing vowels from the context. This problem increases ambiguity for both Arabic and Hebrew. Amharic, however, does not have such a problem. Consonants and vowels in Amharic are inseparable.

Given the similarities and the fact that Arabic and Hebrew have seen more research work in the NLP community since recently, it will be important to examine such works to set standards against which to compare Amharic NLP resources and tools.

2.5.1 Related work on Arabic

Arabic, as the most spoken Semitic language with hundreds of millions of speakers as first or second language, it is receiving considerable attention in NLP research. Arabic comes in many dialects. Much of the research in Arabic has, however, concentrated on the Modern Standard Arabic (henceforth Arabic). It has a larger and modern vocabulary with relatively simplified and standardized grammars.

An Arabic word, like words in other Semitic languages, is usually composed of a stem, plus affixes and clitics. The stem usually consists of a consonantal root and a template. The affixes include inflectional markers for tense, gender, and/or number. The clitics include one or more of conjunctions, prepositions, determiners, pronouns and possessive pronouns. Some of these attach to the beginning of a stem (proclitics) and others attach to the end of the stem (enclitics).

For such a language, designing a POS tagger requires first designing a different tagset than English. Khoja (2001) designed a tagset of 131 tags. Similarly, Buckwalter's Aramorph (Buckwalter, 2002), an Arabic morphological analyzer, uses 135 morphological labels. Both tagsets are similar as they are based on some basic tags. For example, Khoja's tagset is based on five main tags: noun, verb, particle, residual information and punctuation, which are first extended to

35 to account for clitics which, with further sub categorization, became 131 tags. In the subcategorized tags is included information about verb aspects (perfect, imperfect, imperative), person, number and gender. For example, **VPPl2M** will be the tag of the Arabic word *ksrtm* which means "you [plural, masculine] broke". **V** stands for verb, **P** for perfective, **Pl** for plural, **2** second person and **M** for masculine. Nouns and personal pronouns are tagged in a similar fashion. The particle category included prepositions, adverbs, conjunctions, interjections, exceptions and negative markers. Separate tags are also used to indicate dates, numbers, punctuation marks and abbreviations.

Khoja is also credited for introducing the first Arabic POS tagger called APT. He developed a hybrid type of tagger, where both rules and statistics are used. Lacking in annotated corpora for his experiment at the time, Khoja built and manually tagged 50,000 words extracted from the Saudi Al-Jazirah newspaper. However, instead of using the 131 tagset, he used the collapsed version, 35, as it was more laborious to manually tag the corpus using the larger set.

The initial tagging stage in his approach (in APT) involved looking up a word in a lexicon. If it is found, then it is given all the possible tags of that word as specified in the lexicon. If the word is not found in the lexicon as it is usually the case because of the complex morphology of the language, it is reduced to its stem or root form by the process of stemming. Rules are used in the stemming process to strip off the affixes. In most cases, a combination of affixes is used to determine the tag of the word. In some cases, a single affix can determine the tag of a word. For example, the definite article prefix in a word indicates that the word is actually a noun. The pattern of the root of the word is also used to determine its tag.

The stemming process in APT is not perfect for three reasons. Firstly, some "affixes" that look like real affixes are actually part of the stem/root and not affixes. Second, word formation by the concatenation of lexical units sometimes entails spelling changes and these are not accounted for by the stemmer. Third, the lexicon, which he built himself, was small as it is derived from 50,000 words. In spite of such shortcomings, the stemmer achieved an accuracy of 97% using a dictionary of 4,748 triliteral and quadrilateral roots.

The statistical part of APT is used to disambiguate words that received multiple tags (ambiguous words and unknown words). He used a Viterbi decoding algorithm that finds efficiently the sequence of tags that maximizes the product of the lexical probabilities and the transition probabilities, which were calculated

from his 50,000-token corpus. Tested on a text of four corpora of total size 85,159 tokens, he obtained an average accuracy of 90%. Given that on average 70-80% words are unknown and that even after stemming 20% of the words in test corpora are unknown, the result is not bad but it can be reasonably assumed that with better stemming algorithms and larger lexicon files, the result could be improved.

Later efforts resulted in AraMorph (Buckwalter, 2002), a more sophisticated Arabic morphological analyzer with a larger lexicon. It gives multiple morphological analyses of words using 135 morphological labels. This analyzer contains three lexicon files and three compatibility tables. The lexicon files contain all Arabic prefixes, stems and suffixes. The three compatibility tables contain information about permissible prefix-stem, prefix-suffix and stem-suffix pairs. The morphological analyzer has six functions: tokenization, word segmentation, dictionary look-up, compatibility check, analysis report and second look up (for orthographic variants).

The morphological analyzer works as follows: words are segmented into different prefixes, stems and suffixes and these are checked against the lexicon files and the compatibility tables. If the segments are available in the lexicon files and if they are compatible, then the analysis is reported. If the analysis does not return anything, the orthography is checked and alternative spellings are created and the process repeats.

This morphological analyzer has been effectively used to develop the Arabic Treebank. The Arabic Treebank is an annotated data constructed by manually choosing the right analysis from the output of AraMorph.

Both Diab et al. (2004) and Habash and Rambow (2005) applied SVM classifiers using the Arabic Treebank distribution. Their main difference is that the former did not use features extracted from AraMorph but from the words themselves whereas the latter used features extracted from morphological analyses returned by a modified version of AraMorph, called ALMORGEANA (Habash, 2005), a lexeme-based morphological generator and analyzer using the lexical files of AraMorph.

Specifically, the features in Diab's experiment are extracted from a window of five words (current word and the previous and the next two words). The features include every character N-gram ($N \leq 4$) that occurs in the focus token, the five tokens themselves, their type from the set {alpha, numeric}, and POS tags of previous tokens. The class labels for Diab's are the collapsed tagset of 24.

Habash's approach to use morphological analysis was inspired by the conclusion of Hajič (2000) who showed using 5 Eastern European (plus English) that for highly inflectional languages "the use of an independent morphological dictionary is the preferred choice [over] more annotated data".

The training data consists of a set of all possible morphological analyses for each word, with the unique correct analysis marked. Using this data, SVM classifiers are trained for ten morphological features. These classifiers are then applied on every analysis returned by the morphological analyzer to choose the correct one. The classifiers are combined by different mechanisms, one of which is choosing the analysis that has the majority of agreements with the classifiers. Testing on two 12,000 words derived from Arabic Treebank shows an accuracy score of 97.6% where Diab et al. (2004) reported a score of 95.5% on a similar test corpus. It is difficult to attribute this increase in performance to just morphological analysis because the experimental setup was not exactly the same. However, the results are comparable because they use the same collapsed tagset (24) and the training and test sets are approximately equal. In Habash's case, the training was done on 120,000 tokens and 12,000 tokens were used for development and another set of 12,000 for testing. In Diab's case, the development set, training set and test set are derived from 4519 sentences from the Arabic Treebank. 4000 randomly distributed sentences are used for training; 119 for development set and the rest 400 are used for evaluation.

It is interesting to note that most of POS tagging errors that are encountered result from confusing adjectives, JJ, with nouns, NN, or vice versa. For example, 50% of the errors in Diab's experiment are the result of the confusion between adjectives and nouns.

2.5.2 Related work on Hebrew

Hebrew is a Semitic language that has received more attention from the NLP community next to Arabic. Most research before 2000 for Hebrew tagging centered on unsupervised techniques driven primarily because of lack of annotated data (Levinger, 1992; Levinger et al., 1995; Carmel and Maarek, 1999; Adler, 2001).

For example, Adler applied HMM for Hebrew segmentation and POS tagging. The HMM parameters are learned from an untagged corpus using the Baum-Welch algorithm (Baum, 1972). Adler outlines the different possible levels of

segmentation and POS tagging for Hebrew. The first one is the usage of word-level tags, which determines the level of segmentation and the tag of each word. The second one is the usage of morpheme-level tags, with second order Markov model. This too determines the level of segmentation and the nature of the tags for each morpheme. Tests on the word-level tagging achieved an accuracy of 82%. The morpheme-level tagging was not tested.

Adler's approaches to segmentation and POS tagging have been taken up by Bar-Haim et al. (2005); Bar-haim et al. (2008). They developed a segmenter and a tagger for Hebrew based on Hidden Markov Models (HMMs) and compared the two approaches empirically.

Their approach to segmenting and tagging is also similar to that taken by Khoja (2001) for Arabic. A Hebrew morphological analyzer (Segal, 2000) is used to assign a set of possible candidate analyses to each word, where each candidate analysis consists of a segmentation of the word into morphemes. Words that receive multiple analyses from the morphological analyzer are disambiguated using the parameters learned in the Hidden Markov Model. However, Bar-haim's work is different from that of Khoja because the experiments for Hebrew were done for two different levels of segmentation, namely word-level and tag-level.

HMM parameters are learned for both word-level tokenization and morpheme-level tokenization. Second order Markov model is used for morpheme-level tagging and first order Markov model for the word-level tagging. The different orders of Markov models are chosen in order to minimize the data sparsity problem. The lexical and the language model probabilities are calculated from the Hebrew Treebank (Simaan et al., 2001). This corpus is a syntactically and morphologically annotated corpus collected and built from articles from a daily newspaper called Ha'aretz. Words along with their pos-tagged morphemes are extracted from the Treebank version that contains 35,848 tokens and 48,332 morphemes.

In addition to the manually tagged corpus, which is not more than 4% of the Wall Street Journal, 337,651 tokens are used to improve the lexical probability estimations. To avoid probabilities of zeros (on average, 31.3% on average of the test words do not appear in the training corpus), the standard backoff smoothing method of Katz (1987) is employed. All of these tasks are done using SRILM (Stolcke, 2002), a toolkit for constructing language models and for disambiguation.

A tagset of 28 is used for training, which comes from the morphologically tagged corpus by leaving out person, gender, number and tense features. A further

collapsed version of size 21 is used by grouping some tags together for testing, which should normally reduce tagging errors.

5-fold cross validation using a training set of about 1,598 sentences (on average, 28,738 words and 39,282 morphemes) and test set of 250 sentences shows an accuracy of 88.50% for POS tagging and 96.74% for segmentation when tokenization is done at word level. For tokenization done at morpheme level, accuracy for POS tagging is 89.27% and 96.55% for segmentation. From the results the following hypothesis is formulated.

> "Morpheme-level taggers outperform word-level taggers in their tagging accuracy, since they suffer less from data sparseness. However, they lack some word-level knowledge that is required for segmentation."

Error analysis of the tagging methods shows that the most common error type is related to definiteness marker **h**. This error is more common in morpheme tagging than in word tagging. In order to handle the problems associated with the **h**, a modified model was developed that improved the accuracy to 89.59% for tagging and to 97.05% for segmentation.

It is interesting to note that Mansour (2008) adapted this tagger into Arabic by replacing the Hebrew morphological analyzer with an Arabic one and achieved an accuracy of 96.3% over 26 tags on a 89k token corpus.

2.6 Previous Work on Amharic

NLP research on Amharic has started fairly recently and has been constrained by lack of linguistic resources and an authoritative body to define and develop them. Unlike Arabic and Hebrew, which have syntactically annotated Treebanks, Amharic does yet have a Treebank. Even so, NLP researchers from native speakers to non-speakers have shown interest in the language and developed prototypes by applying some of the state-of-the-art tagging models (Getachew, 2001; Adafre, 2005; Gambäck et al., 2009; Tachbelie and Menzel, 2009).

Getachew (2001) is the pioneer for Amharic POS tagging experiments. He developed a tagging prototype using Hidden Markov models, which he trained and tested on a text of one page.

His contribution also included the definition of a tagset. He suggested using 25 tags. This tagset has served as a basis for the tagsets used by subsequent researchers. The tagset he used are N,NV, NB, NP, NC, V, AUX, VCO, VP, VC, J, JC, JNU, JPN, JP, PREP, ADV, ADVC, C, REL, ITJ, ORD, CRD, PUNC, and UNC. One good design strategy in designing a tagset is to start with major classes and to extend these with subclasses. Evaluating Getachew's tagset on this strategy shows that his tagset can be reduced to a smaller tagset with the capacity to capture a more abstract description of the language such as N* (nouns), V* (verbs), J* (adjectives), etc. For example, NP, NV etc can be reduced to N. Similarly, VCO, VP and VC can be reduced to V. In some cases, however, it is not clear how to find the major classes from the tags themselves. For example, REL (for relative verbs) is a kind of verb but it is not possible to see that from the tag itself.

Getachew's tagset also defines the context of its usage. The tags are used when words are tagged as they appear in text, separated by space. Because Amharic words usually consist of stems with prefixes and suffixes attached to them, he could have as well designed a tagset that would apply after splitting the word into prefixes, a stem and suffixes. Instead, he designed a tagset that deals with the words as the smallest units for tagging, which implied, in order not to lose information, that the tagset include information about the constituting prefixes and suffixes which usually belong to different word classes (POS). The tag NPC, for example, indicates that the word is a noun prefixed by a preposition and suffixed by a conjunction.

Adafre (2005), who did the next POS tagging experiment for Amharic, revised Getachew's tagset and reduced it to ten. The ten POS tags are Noun (N), Verb (V), Auxiliary verbs (AUX), Numerals (NU), Adjective (AJ), Adverb (AV), Adposition (AP), Interjection (I), Residual (R), and Punctuation (PU).

In addition to reducing the tag size, Adafre made also two important modifications to Getachew's tagging approach. From the tag descriptions of Getachew, it can be seen that grammatical functions take precedence over morphology in deciding the POS category of a word. This can also be inferred from his decision to take words that form collocations as one unit for tagging. So Adafre's first modification is to consider the words in collocations as separate units for tagging, avoiding the need of identifying them.

The second modification is related with tagging a **ye+NOUN** Amharic construction. To such a word, Getachew's approach assigns the tag JPN, signifying that it

functions as an adjective. For example, *yetaywan sahn* (=A Taiwan made plate) is given JPN. However, the **ye+NOUN** construction can be seen as a simple morphological variant of the NOUN and so be tagged as such. So, in Adafre's case, this would be categorized under noun subclasses rather than under adjective subclasses. This has the advantage of simplicity and improving performance as **ye + NOUN** is also used in other functions (eg: possession).

The motivation for reducing the tagset from 25 to 10 is lack of annotated resources. The more refined that the tags are, the more annotated data is needed for learning an accurate tagging model. As there were no POS annotated data at the time, Adafre collected five news articles and manually annotated them, which he then used for both training and testing of a stochastic model based conditional random fields (Lafferty, 2001).

He obtained an average accuracy of 74% on a 5-fold cross-validation where one file is used for testing and the other files for training. The reason for the poor performance (compared to the state-of-the-art results) is the small size of the dataset. 80% of the words in the test files consist of unseen words. From this result and successful experiences in other experiments for large datasets, it became clear that Amharic POS-annotated data is necessary to achieve performances comparable to the state-of-the-art results.

In 2006, a medium-sized corpus of reportedly 210,000[2] tokens annotated with parts of speech was released (Demeke and Getachew, 2006). The corpus consists of 1065 news articles collected from Walta Information Center (WIC), a private news agency located in Addis Ababa. It is tagged with 31[3] parts of speech and is publicly available on the Internet. This corpus has been a useful resource for the recent experiments on Amharic POS tagging.

Gambäck et al. (2009) and Tachbelie and Menzel (2009) applied different state-of-the-art tagging methods using the WIC corpus and obtained worse performances than the best results for Arabic or English.

Gambäck conducted detailed experiments using TnT (Brants, 2000), SVMTool (Giménez and Marquez, 2004) and Mallet (McCallum, 2002) on three different tagsets. The overall accuracies using the ELRC[4] tagset are 85.56% for TnT, 88.30% for SVM and 87.87% for MaxEnt as shown in table 2.2. Similarly, Tachbelie and Menzel (2009) also conducted similar experiments using TnT and SVM-

[2]actual counting reveals a number less than that
[3]30 is reported, actual counting shows 31
[4]Ethiopian Languages Research Center

Table 2.2: Previous POS Tagging Accuracies for Amharic

Tagger	**Known**	**Unknown**	**Overall**
TnT	90.00	52.13	85.56
SVM	89.58	78.68	88.30
MaxEnt	89.44	76.05	87.87
Basline			35.50

Tool models with overall accuracies of 82.99% for TnT and 84.44% for SVM. For both sets of experiments, the best performances are achieved by SVM but Gambäck's SVM performs better (88.30% against 84.44%).

Those poor performances (compared to English or Arabic) can be explained by four reasons. First, the corpus used is small; it is one-sixth of the size of the WSJ corpus. Second, the taggers use no more knowledge source than a pre-tagged training corpus. Third, the quality of the corpus is poor. Tagging errors and inconsistencies are considerable in the corpus. Fourth, little parameter tuning of the algorithms was done to suit the WIC corpus.

Except for Adafre (2005), who used dictionaries of affixes and some 15,000 entries (Aklilu, 1987) with their POS tags (Noun, Verb, Adjectives, Adverb, and Adposition), all other previous POS experiments for Amharic used language independent features. The features include a subset of the following:

- Lexical features consist of
  - the current word
  - the two words to the left and to the right of the current word
- Morphological/syntactical features
  - Prefixes and suffixes of length from 1 to 4/5
  - All digits
  - Is word capitalized
  - Contains digits
  - Contains hyphen
  - The previous two tags

Such features are quite effective for most languages but more can be done by examining more morphological features of the given language. For Amharic, one

feature that is important and not included by previous experiments is the vowel patterns embedded in words. For example, *kebad* and *kelal* are adjectives and share the same **e**,**a** vowels. Verbs also show similar vowel patterns. *manbebu* (that he read), *madregu* (that he did), *mabedu*(that he became mad), etc all share **a**, **e**, **u** vowel patterns. Another feature that may prove useful is the radicals (the consonants in the words). For example, *sebere* (he broke), *sebro* (having broken (he)), *sebra* (having broken(she)) can be reduced to just the radical *sbr* and be treated as a verb. Both the vowel pattern and the radical features have the advantage of reducing data sparsity problem and so language modeling techniques would perform better by capturing them.

On the other hand, the feature of capital letters that is so important in identifying names of people and places in English does not help in Amharic as the capitalization concept does not exist in the writing system. For that reason, it is not important to have it as a feature for learning. However, language independent tools include it. In fact, in SERA[5], the letters in lower or upper cases account for different letters.

The right features are not sufficient for performance improvement if the quality of the corpus is poor. The WIC corpus has significant errors and tagging inconsistencies. This problem has been acknowledged by researchers who worked on it and they have made efforts to correct some of them. For example, Gambäck's experiments were done on a partially corrected WIC corpus. The corrections included tagging non-tagged words, removing double tags, treating consistently ”’ and ’/’ as punctuation, retagging some wrongly tagged words and some spellings errors. However, they acknowledge that tagging inconsistencies related to time and number expressions had been left as they were. Therefore, this type of error and others left unnoticed have contributed to the relatively poor performance.

This dissertation will attempt to improve performance by doing three things. The first one is based on cleaning the corpus. This step is crucial and will determine the performance of any method. This is probably the reason Tachbelie and Menzel (2009) got worse results than Gambäck et al. (2009) as the former did not make any effort to clean the corpus. The second thing involves feature selection. The usual features used for POS tagging will be used. In addition, however, the vowel patterns and the radicals, which are characteristics unique to Semitic languages, will be included. The third is by applying the state-of-the-art tagging machine learning algorithms and doing necessary parameter tuning as much as possible. Algorithms used are conditional random fields, support vector machine, Brill

[5]System for Ethiopic Representation in ASCII

tagging and TnT.

All of these things combined have contributed to the most accurate part of speech tagger ever reported for Amharic.

Chapter 3

Tagging Models

- Introduction
- Hidden Markov Based Models
- Conditional Random Fields
- Support Vector Machines
- Brill Tagging

3.1 Introduction

Part of speech tagging can be done either using handcrafted linguistically-motivated rules or by stochastic methods. While rules are specific for languages, stochastic or machine learning based tagging methods are independent of languages. In this chapter, we will discuss the mathematical probabilistic models that have proven useful in part-of-speech tagging.

3.2 Hidden Markov Based Models

A hidden Markov model (HMM) is a finite state automaton with stochastic state transitions and observations. The automaton models a probabilistic generative process whereby a sequence of observations is produced by starting in some state, emitting an observation selected by that state, passing to a new state, emitting another observation-and so on until a designated destination state is reached.

More formally, an HMM model is characterized by the following (Rabiner, 1989).

1. N, the number of states in the model. Generally the states are interconnected in such a way that any state can be reached from any other state. The hidden states often represent important physical aspects. In part of speech tagging, the states represent the tags. We denote the individual states as $T = t_l, t_2, ..., t_N$, and the state at time t as q_t.

2. V, the number of distinct observation symbols per state, i.e., the discrete alphabet size. The observation symbols correspond to the physical output of the system being modeled. For part of speech tagging, the observation symbols are the words of a given language. We denote the individual symbols as $W = w_l, w_2, ..., w_M$

3. The state transition probability distribution $A = \{a_{ij}\}$ where

$$a_{ij} = P(q_{t+1} = t_j / q_t = t_i), \qquad 1 \leq i, j \leq N$$

$$a_{ij} \geq 0$$

   When there are no connections between states, the corresponding a_{ij} transition probability is zero. For all other cases, it is greater than zero.

4. The observation symbol probability distribution in state $j, B = \{b_j(k)\}$, where

$$b_j(k) = P(w_k / q_t = t_j) \qquad 1 \leq i, j \leq N, \quad 1 \leq k \leq V$$

5. The initial state distribution $\pi = \{\pi_i\}$, where

$$\pi_i = P(q_1 = t_i), \qquad 1 \leq i \leq N$$

The above enumeration shows that HMM models have two parameters and three probability distributions. The two parameters are N (number of states) and V (vocabulary size) and the three probability distributions are A, B and π. The three probability distributions are henceforth referred to as λ where

$$\lambda = (A, B, \pi)$$

Given an HMM model with values for N, V and λ, there are three problems that are of interest as formulated by Rabiner (1989):

Problem 1 Given the observation sequence $O = O_1 O_2 ... O_T$ and a model $\lambda =$

(A, B, π), how do we efficiently compute $P(O/\lambda)$? In other words, what is the probability of the observation sequence, given the model?

Problem 2 Given the observation sequence $O = O_1O_2...O_T$ and a model $\lambda = (A, B, \pi)$, how do we choose the underlying state sequence $Q = q_1q_2...q_T$?

Problem 3 How do we adjust the probability distributions $\lambda = (A, B, \pi)$ to maximize $P(O/\lambda)$?

Such are the questions that can be raised about an HMM model. Solving the first problem is important in comparing two or models. We choose a model that gives high probability to observation sequences. Solution to the second problem is important in discovering the underlying hidden state sequences that could have given rise to the observation. The solution to the third problem has the advantage of giving us the best model for the observations.

Usually, the relevant problem for part of speech tagging is the second problem. The problem in part of speech tagging is that of discovering the underlying tag sequences for a given sequence of words.

More formally, given a sequence of words $W = w_1, w_2...wi...w_n$ where $w_i \in V$(Vocabulary), what is the most probable sequence of tags $T = t_1, t_2...ti...t_n$ where $t_i \in T$(Tagset) that could have given rise to these words?

To solve this problem, we need to have appropriate values for the model $\lambda = (A, B, \pi)$ and an efficient algorithm to search through the space of tag sequences for the optimal one. The values for λ can be estimated from a tagged corpus. POS tagging algorithms that are based on HMM differ in the ways they estimate values for $\lambda = (A, B, \pi)$.

Viterbi algorithm, a dynamic programming based algorithm, is used to find the best sequence (Viterbi, 1967), which basically involves solving the following probability function.

$$\begin{aligned} T^* &= \arg\max_T P(T/W) \\ &= \arg\max_T P(T)P(W/T) \qquad \text{Bayes' Theorem} \end{aligned} \tag{3.1}$$

$P(T)$ and $P(W/T)$ in equation 3.1 are the transition probabilities and observation(lexical) probabilities, respectively. $arg\,max$ tells us that the function returns the tag sequence that maximizes the probability function value. Transition prob-

ability captures more of context and tag dependencies.

$$\begin{aligned} P(T) &= P(t_1, t_2 \dots t_n) \\ &= P(t_1)P(t_2/t_1)P(t_3/t_2t_1) \dots P(t_n/t_{n-1}t_{n-2} \dots t_1) \quad \text{Chain Rule} \end{aligned} \tag{3.2}$$

Equation 3.2 assumes that the current tag depends all previous tags. It is hard to find values for such an assumption. There is too little annotated data for this to work. So, what is common to do in this case is to assume that the current tag depends on some fixed number of previous tags. Depending on this fixed number, we have unigram, bigram and N-gram in general. Unigram means the current tag does not depend on previous tags. Bigram means the current tag depends on the previous tag and so on. For example, for the bigram case, the equation becomes as shown below.

Transition probability:

$$P(T) = P(t_1)P(t_2/t_1)P(t_3/t_1t_2)P(t_4/t_2t_3) \dots P(t_n/t_{n-2}t_{n-1}) \tag{3.3}$$

$$\approx \prod_{i=1}^{n} P(t_i/t_{i-2}t_{i-2}) \tag{3.4}$$

Lexical probability:

$$P(W|T) = P(w_1/t_1 - t_n)P(w_2/w_1t_1 - t_n) \dots P(w_n/w_1 - w_{n-1}t_1 - t_n) \tag{3.5}$$

$$\begin{aligned} &\approx P(w_1/t_1)P(w_2/t_2)P(w_3/t_3) \dots P(w_n/t_n) \\ &\approx \prod_{i=1}^{n} P(w_i/t_i) \end{aligned} \tag{3.6}$$

Equation 3.5 implies current word is determined by the tag sequence and all previous words, whereas equation 3.6 assumes current word is determined completely by its tag. The values for the transition and lexical probabilities are estimated by the maximum likelihood estimate given below for the bigram case.

$$P(t_i/t_{i-1}) = \frac{Count(t_{i-1}, t_i)}{Count(t_{i-1})} \qquad A \text{ parameter} \tag{3.7}$$

$$P(w_i/t_i) = \frac{Count(t_i, w_i)}{Count(t_i)} \qquad B \text{ parameter} \tag{3.8}$$

Using the above two equations, A and B parameters can easily be calculated by counting and dividing. In counting, it is possible that for some sequences of tokens we may get zero. This can be bad because it implies particular tokens can never occur. However, this is not necessarily true. It may just mean that the training corpus did not have the tokens. How we deal with this count zero

problems also determines the kind of our HMM model. The methods that try to solve this problem are called smoothing techniques and there are so many of them (Lidstone, 1920; Johnson, 1932; Good, 1953; Katz, 1987; Church and Gale, 1991; Chen and Goodman, 1999).

3.3 Conditional Random Fields

Conditional Random Fields (CRFs) are conditional probability distributions that take the form of exponential models (Lafferty, 2001). CRFs relax certain assumptions about the input and output sequence distributions. CRFs can contain any number of feature functions that can inspect the entire input sequence at any point during inference.

$$P(t/w) = \frac{1}{Z(w)} exp(\sum_{j=1}^{n} \sum_{i=1}^{k} \lambda_i f_i(t_{j-1}, t_j, w_{1:n}, j)) \tag{3.9}$$

Z is the normalization factor to make $P(t/w)$ a valid probability function over tag sequences and is equal to

$$Z = \sum_{t \in T} exp(\sum_{j=1}^{n} \sum_{i=1}^{k} \lambda_i f_i(t_{j-1}, t_j, w_{1:n}, j)) \quad \text{where } T \text{ is the set of tags}$$

Indices $j = 1$ to n represent word positions in a sentence.
Indices $i = 1$ to k represent the feature functions.
λ_i is the weight for feature f_i. The λ_is are the parameters of the CRF model that must be learned. The feature functions f_is are the key components of CRF. The general form of a feature function for a linear chain is

$$f_i(t_{j-1}, t_j, w_{1:n}, j)$$

This function looks at a pair of adjacent states $t_{j-1}t_j$, the whole input sequence $w_{1:n}$, and the current word position in the sentence (j). For example,

$$f_i(t_{j-1}, t_j, w_{1:n}, j) = \begin{cases} 1 & \text{if } j = n \text{ and } t_j = \text{"VERB"} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

The above feature function is inspired by Amharic syntax. The last word in an Amharic sentence is a verb and so it will be on for a word at the end of a sentence and off for all other positions.

For a feature function f_i that is active, the following conditions hold:

- If $\lambda_i > 0$, it increases the probability of the tag sequence $t_{1:n}$.
- If $\lambda_i < 0$, it decreases the probability of the tag sequence $t_{1:n}$.
- If $\lambda_i = 0$, it has no effect on the probability of the tag sequence $t_{1:n}$.

The difference between Maximum entropy models (Ratnaparkhi, 1996) and CRF is that a MEMM uses per-state exponential models for the conditional probabilities of next states given the current state, while a CRF has a single exponential model for the joint probability of the entire sequence of labels given the observation sequence. In other words, CRFs solve the label bias problem (Lafferty, 2001).

The per-state normalization requires that all the mass that arrives at a state must be distributed among all the successor states. An observation can affect which destination states get the mass, but not how much total mass to pass on. This causes a bias toward states with fewer outgoing transitions. In the extreme case, a state with a single outgoing transition effectively ignores the observation. This is a bias problem.

3.4 Support Vector Machines

SVM is a practical machine learning algorithm used for binary classification (Boser et al., 1992; Cortes and Vapnik, 1995). In its basic form, SVM learns a linear hyperplane that separates the set of positive examples from the set of negative examples with maximal margin (to increase generalization capacity of the model). The margin is defined as the distance between the hyperplane and the nearest of the positive and negative examples.

Given a set of m training examples $\{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$, where $x_i \in R^n$ and $y_i \in \{-1, +1\}$, the problem is to find two parameters a weight vector $\mathbf{w}$ and a bias $\mathbf{b}$ such that the margin between the support vectors of positive and negative examples is maximum. Finding the right dividing hyperplane requires solving the

following optimization problem.

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w^T w} + C\sum_{i=0}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0 \end{aligned} \tag{3.11}$$

ϕ is a mathematical function that maps feature vectors x_i into higher and possibly infinite dimensional feature space. In this feature space a linear decision surface (a hyperplane) with a maximal margin is constructed whose special properties ensure high generalization ability of the learning machine. $C > 0$ is the penalty parameter of the error term. Larger C values tend to overfit training data, whereas smaller values tend to underfit it.

ϕ functions for the input vectors are not directly calculated but a kernel trick is applied (MA et al., 1964). In other words, a kernel function that is equivalent to the transformed dot product of input vectors is used. Mathematically,

$$K(x_i, x_j) = \phi(x_i)^T\phi(x_j) \tag{3.12}$$

Commonly used kernels are:

Linear: $K(x_i, x_j) = x_i^T x_j$

Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$

Radial Basis Function (RBF): $K(x_i, x_j) = exp(-\gamma\|x_i - x_j\|^2),\ \gamma > 0$

Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

γ, r, and d are kernel parameters that are chosen depending on the given corpus. Since SVMs are binary classifiers and part-of-speech tagging is a multiclass classification problem, we will need as many SVMs as there are tags by adopting the one-versus-rest approach. If we have T tags, we will have T SVMs. For a new test data x_i, we will choose the SVM that is the most confident.

3.5 Brill Tagging

In stochastic tagging methods, tagging information is stored in thousands of lexical and contextual probabilities. The advantage of this is that tagging information

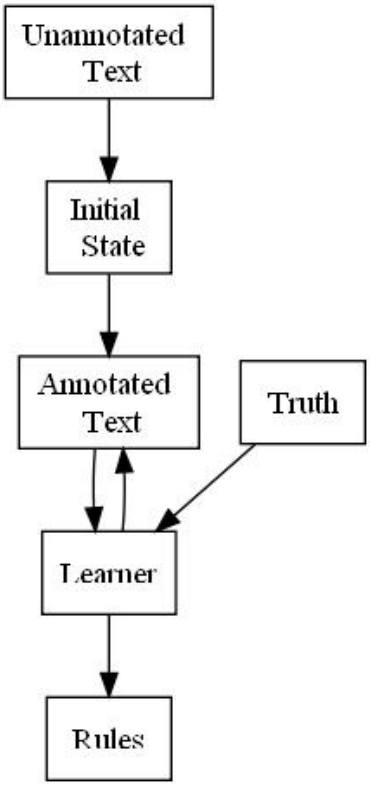

Figure 3.1: Transformation-based Error-driven Learning

is automatically learned from a corpus, saving the cost of laborious manual rule engineering. However, the same method has also a number of disadvantages: one, the tagging information stored is too much or too big; two, the information stored is a bunch of numbers with little linguistic meaning. The Brill (1992) tagger attempts to overcome those disadvantages by reducing the amount of information required for tagging and by using linguistically meaningful rules and still avoiding having to construct the rules manually.

The paradigm of Brill tagging is based on a more general learning technique called transformation-based error-driven learning. Figure 3.1 shows a simple graphical representation of the learning process. This learning process involves two important components. First, words of a given text are passed through an initial state tagger. The initial state tagger can be as simple as a tagger that assigns random tags or the same tag for all words or the most likely tag for each word. The result of the initial state tagger is a tagged text. The second component of Brill tagger takes the tagged text and applies transformations to make it more like the truth, which is text manually tagged by humans.

The transformations are tagging corrections that are applied when certain contexts or lexical properties are satisfied and they are instantiated from the following patch templates (Brill, 1992; 1995):

Change tag **a** to tag **b** when:

1. The preceding (following) word is tagged z.
2. The word two before (after) is tagged z.
3. One of the two preceding (following) words is tagged z.
4. One of the three preceding (following) words is tagged z.
5. The preceding word is tagged z and the following word is tagged y.
6. The preceding (following) word is tagged z and the word two before (after) is tagged y.
7. The current word is (is not) capitalized.
8. The previous word is (is not) capitalized.
9. The preceding (following) word is w.
10. The word two before (after) is w.
11. One of the two preceding (following) words is w.
12. The current word is w and the preceding (following) word is x.
13. The current word is w and the preceding (following) word is tagged z.

a, **b**, z and y belong to the set of parts of speech and w and x are words in the training set.

Every instantiation of the transformation template is then applied to every tagging error observed by comparing the output of the initial state tagger to the truth. Each such transformation may result in correcting x number of errors and introducing y number of errors. The real reduction in errors comes by subtracting y from x to get a score $s(s = x - y)$. The patches with the highest s scores (ie. resulting in higher error reductions) are chosen and become part of the set of rules that will finally be used to tag a new text.

The same approach can be used to tag unknown words. The only change that has to be made in the mentioned procedure is to add new transformation templates. Here are the templates for tagging unknown words as outlined by Brill.

Change the tag of an unknown word from **X** to **Y** if:

1. Deleting(Adding) the prefix (suffix) $x, |x| \leq 4$, results in a word x.
2. The first/last (1,2,3,4) of the word are x.
3. Word **W** ever appears immediately to the left(right) of the word.
4. Character **Z** appears in the word.

The templates shown above were originally made for English, where the morphology is less complex and where the word order is quite predictable. In other languages, where much of the tagging information is in parts of the words, there is a need for new templates. In this case, the templates suggested by Brill for tagging unknown words can be extended for tagging known words too. This can be done by creating templates that look at the affixes of words before and after the current word.

Chapter 4

Amharic Morphology

- Introduction
- Amharic Orthography
- Amharic Morphology
- NLP Challenges in Amharic

4.1 Introduction

Parting of speech tagging using stochastic methods can be done without directly using the morphology of the language by relying heavily on the size of the training corpus and this may achieve reasonably accurate results for morphologically less complex languages. However, for morphologically complex languages like Amharic, a significant performance improvement can be achieved by integrating the essential morphological elements in the features that are learned by stochastic methods. In this chapter, we will explore briefly Amharic orthography and morphology with the view to finding vowel patterns that will be used in improving POS tagging accuracies.

4.2 Amharic Orthography

Amharic is written in Ethiopic[1] or Fidel, which is the writing system also used by Tigrigna[2]. Unlike Arabic, Hebrew and Syriac, which have their vowel signs written independently above, below, or within the letters, the Ethiopic writing system attaches its vowel signs to the body of the consonant, so that there are as many modifications of the form of each consonant as there are vowels.

The Ethiopic alphabet has 33 basic characters. Each such character is modified in some regular fashion to reflect the seven vowels of the language. Therefore, there are in total $33 * 7 = 231$ characters. Even though Amharic alphabet is Unicode standard[3], it is sometimes convenient to represent it in ASCII. Written in SERA (Firdyiwek and Yaqob, 1997) (System for Ethiopic Representation in ASCII), the basic characters which can also be called the consonants of the language are in alphabetic order:

$$C = \{h, l, H, m, s, r,' s, x, q, b, t, c,' h, n, N, a, k, K, w,' a, z, Z, y, d, j, g, T, C, P, S,' S, f, p\}$$

The vowels are:

$$V = \{e, u, i, a, E, I, o\}$$

Out of the 33 basic consonants, Amharic identifies 28 unique sounds. This implies that, in some cases, more than one consonant is used to represent the same sound. These are:

- {h, H, h'}
- {s, s'}
- {S,'S} and
- {a, a' }

The above sets represent different sounds. The letters in each set represent the same sound. It is important to recognize these letters in natural language processing tasks. For example, an Amharic word that has the letter **h** can be written in three equivalent ways and must be treated as one for NLP tasks.

[1]Ethiopic is the name used outside Ethiopia to refer to the writing system; Fidel is more familiar to Ethiopians.

[2]Semitic language spoken in northern Ethiopia and Eritrea

[3]Ethiopic (U+1200 - U+137F), Ethiopic Extended (U+2D80 - U+2DDF), Ethiopic supplement(U+1380 - U+139F)

As the Ethiopic alphabet does not distinguish between lower and upper cases and as there are more sounds than can be handled by the 26 Latin letters, differences in letter case have been used to represent different sounds. The alphabet can be thought of a 33x7 matrix table of symbols for all possible consonant-vowel combinations (CV) where the rows are the consonants and the columns are the vowels.

Some punctuation marks used in Amharic are like those of English. However, there are important differences. Four points (2 consecutive colons) are used to mark the end of a sentence and where English uses comma, Amharic uses a colon with a bar.

Amharic is a SOV language where words are separated by space. Except in poems, the head verb is usually at the end of a sentence. Unlike Arabic and Hebrew, Amharic is written from left to right.

4.3 Amharic Morphology

Words in Amharic text can be classified into 'native' and 'borrowed' words. We refer to words that have not been borrowed from other languages as 'native'. We refer to words that have come from other languages as 'borrowed'. Such words are like 'kompiwter' for a computer and 'mobayl' for a mobile phone. 'Native' words can further be divided into derived words and non-derived words or primitives. Primitives and 'borrowed' words can be put in one class for natural language processing applications. This class usually consists of nouns.

Nouns in Amharic can be inflected for gender, number, definiteness, and case. The definite article and conjunctions attach to the end of a noun, while prepositions are mostly prefixed. A regular expression that recognizes all morphological changes for a noun is given below.

```
Surfaceform1 =  (prep OR genitive ) AND noun AND (fm OR pl)?
                AND definiteness AND case and conj

Surfaceform2 =  prep AND noun AND (fm OR pl)?
                AND (definiteness OR possession) AND case AND conj

Noun surface form = surfaceform1 OR surfaceform2
```

Table 4.1: Examples of Morphological Changes for Amharic Nouns

Amharic	English
bET	house
bET-u	the house
bET-u-n	the accusative case
bET-u-m	the house also
bET-u-na	the house and
bET-na	house and
bET-cew	their house
bET-oc	houses
ye-**bET**-oc-E	my house's
ke-**bET**-u	from the house
ye-**bET**-um	the house's also
ye-**bET**-oc-achu	your houses'
le-**bEt**-oc-can	for our houses
be-**bEt**-wa	by her house

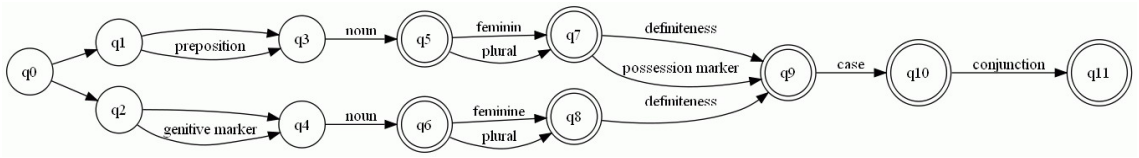

Figure 4.1: Finite State Automata that Recognizes Amharic Nouns

If there is a genitive marker, there is no possession marker. That is the reason we have surface_form1 and surface_form2. We have used one of the most important properties of regular expressions[4] to combine the two regular expressions to form another one. Figure 4.1 is the finite state automata version of the regular expression described above and table 4.1 shows examples of the morphological changes that a given noun can undergo.

Like many other Semitic languages, derived Amharic words are based mostly but not exclusively on tri-consonantal roots/radicals. In the previous section, we have said that Amharic unique consonants are 28. This gives us $28^3 = 21{,}952$ possible tri-consonantal roots. There are also a considerable number of frequent bi-consonantal roots, in theory $28^2 = 784$. So in total, we have 22,734 bi- and tri-consonantal roots. Different words and stems are formed by inserting vocalic patterns in these roots. For example, the verb root *sbr* for break has among others the forms listed in table 4.2. As can be seen from the table, the vowel patterns can take the form of [e,a,i], [0,a,i], etc. The process of inserting vocalic patterns in the roots is called intercalation or interdigitation. By their ordered presence and absence, the vocalic patterns are going to be useful in identifying the correct

[4]Union, intersection, etc operations on regular expressions give us other regular expressions

tags for words. After intercalation (interdigitation), the words or stems undergo the more familiar morphological changes that result from concatenation. Thus, the resulting different word forms reflect, among other things, subject, gender, number, object, possession, negation, tense, beneficative, malfactive, etc. The

Table 4.2: Examples of 'sbr' Productions to Form Amharic Words

'sbr' forms	**Meaning**
sebari	one who breaks
sbari	a fragment
sebara	broken
sebere	he broke
asebere	he made somebody break something
sebabere	he breaks something again and again
tesebere	it has got broken
asabere	he helped in breaking something
asebabere	he helped in breaking something into pieces
seberku	I broke
seberec	she broke
seberu	they broke
sebern	we broke
seberk	you broke
seberachu	you(pl) broke
Isebralehu	I will break
sebrealehu	I have been breaking
Iyeseberku	I am breaking
siseber	while it was being broken
yemiseber	something that can be broken
mesberia	an instrument for breaking

interesting problem with respect to Amharic derived words and especially verbs is whether one can come up with a regular expression/relation to represent all valid forms of a root (example: 'sbr'). This problem is challenging as there are some phonological changes, which need to be identified and encoded. However, this has just been done recently in Indiana University by Gasser (2009*b*).

In the following subsections, we will look into the details of word formation with emphasis on the vocalic patterns (Ymam, 2007). In the word derivations, C represents one of the thirty-three consonants. The set {e, u, i, a, I, o} represents the vowels. In writing, CI is equivalent to C. Amharic pronunciation does not allow the first two consonants of a word to follow one another without a vowel 'I'. Native speakers of the language pronounce this vowel when it is necessary (even though it is not normally written). 'C?' means C may or may not occur. A geminate C is written as CC. 0(zero) represents the absence of a vowel. The word derivation rules shown hereafter have matrix-like operations where '+' means

concatenation and 'T' means transpose. In most cases, instead of the general formulas, the right hand sides of the rules represent instances and their meanings.

4.3.1 Derived nouns

Nouns are derived from other nouns, adjectives and verb roots.

1. noun/adjective + *net* ⇒ abstract noun
   Example: (*deg* + *net* ⇒ *degnet* /kind + ness ⇒ kindness)
2. noun/ adjective + et ⇒ abstract noun
   Example: (*xum* + *et* ⇒ *xumet*/appintment(post))
3. Noun + *eNa* ⇒ *fereseNa*/horse-rider
4. $\begin{bmatrix} C & CC? & C \end{bmatrix} + \begin{bmatrix} I & I? & 0 \end{bmatrix}^T$ + *et/ox/at/o/ot/ota/na* ⇒ noun
   Example: *srk*(root for stealing) + *ox* ⇒ *srkox*/theft
5. $\begin{bmatrix} C & CC? & C \end{bmatrix} + \begin{bmatrix} I & a & 0 \end{bmatrix}^T$ + *E* ⇒ *wdasE*/praise
6. $\begin{bmatrix} C & CC \end{bmatrix} + \begin{bmatrix} I & 0 \end{bmatrix}^T$ + *Et* ⇒ *skEt*/success
7. $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} I & I? & 0 \end{bmatrix}^T$ + *it* ⇒ *tnbit*/prophecy
8. $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} I & 0 & 0 \end{bmatrix}^T$ + *iya* ⇒ *gTmiya*/match
9. $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} I & 0 & 0 \end{bmatrix}^T$ + *a* ⇒ *Cfra*/follower
10. $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} e & 0 & 0 \end{bmatrix}^T$ ⇒ *serg*/wedding
11. $\begin{bmatrix} C & CC? & C \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^T$ ⇒ *qereT*/tax
12. $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^T$ + *a* ⇒ *sebera*/breaking- event
13. $\begin{bmatrix} C & C? & C \end{bmatrix} + \begin{bmatrix} e & a & 0 \end{bmatrix}^T$ + *i* ⇒ *sebari*/breaker ('do'-er nouns)
14. $\begin{bmatrix} C & CC & C \end{bmatrix} + \begin{bmatrix} I & a & 0 \end{bmatrix}^T$ + *i* ⇒ *sbari*/(the broken thing) ('do'-ee nouns of the type 'appointee')
15. $\begin{bmatrix} C & CC? & C \end{bmatrix} + \begin{bmatrix} e? & e & 0 \end{bmatrix}^T$ + *i* ⇒ *mesber*/(the action of breaking)

16. *me* + $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} 0 & e & 0 \end{bmatrix}^{T}$ + *iya* ⇒ *mesberiya*/(noun for the instrument by which things are 'done')

17. *a* + $\begin{bmatrix} C_1 & C_2 & C_2 & C_3 \end{bmatrix} + \begin{bmatrix} e & a & e & 0 \end{bmatrix}^{T}$ ⇒ *asebaber*/ (noun for the way of 'doing' things, in this case way of breaking)

The vowel patterns which may indicate that a particular word is a noun are [I,a,E], [I,E], [I,I,i], [I,0,I,a], [I,0,a], [e,0,0], [e,e,0], [e,e,a], [e,a,i], [I,a,i], [e,e,e,0], [e,0,e,i,a], and [a,e,a,e,0]. The suffixes for most nouns are *-net*, *-et*, *-ox*, *-at*, *-o*, *-ot*, *-ota*, *-na* and *-eNa*.

4.3.2 Derived adjectives

Adjectives are derived from nouns and verb roots (Ymam, 2007).

1. noun + *eNa* ⇒ *hayleNa*/powerful (character adjectives)
2. noun + *ama* ⇒ *terarama*/mountainous (descriptive adjectives)
3. noun + *awi* ⇒ *hagerawi*/national (scope adjectives)
4. $\begin{bmatrix} C & CC? & CC? \end{bmatrix} + \begin{bmatrix} e & a & 0 \end{bmatrix}^{T}$ + *a* ⇒ *sebara*/broken
5. $\begin{bmatrix} C & CC? & C \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^{T}$ ⇒ *derek*/dry (character adjectives)
6. $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} I & u & 0 \end{bmatrix}^{T}$ ⇒ *nSuh*/clean
7. $\begin{bmatrix} C & C & C \end{bmatrix} + \begin{bmatrix} e & a & 0 \end{bmatrix}^{T}$ ⇒ *kebad*/heavy, *kelal*/light (content adjectives)

The vowel patterns which may indicate that a particular word is an adjective are [e,a,0], [e,e,0], [I,u,0] and [e,a,0]. The suffixes for most adjectives are *-eNa*, *-ama*, and *-awi*. From the vowel patterns and suffixes for adjectives and nouns, we can see that [e,e,0] and *-eNa* are shared between the two. In this case, we have to look for other means of distinguishing them. One simple way, which may serve as a baseline, is to say it is a noun, because there are a lot more nouns than adjectives.

4.3.3 Derived verbs

Amharic verb stems are used with different prefixes and suffixes. These affixes may express tense, mood, aspect and person. The verbs may also agree with the person, gender and number of their subjects and objects. The following derivations are used to form the stems to which the prefixes and suffixes are added(Ymam, 2007).

1. $\begin{bmatrix} C & CC & C \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^T \Rightarrow$ *lebes-*/dressed
   It is a stem for verbs that express past events (past aspect) and *'lebes-'* makes sense where there is an inflection for the subject ('lebesku' = I dressed)

2. $a + \begin{bmatrix} C & CC & C \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^T \Rightarrow$ *alebes-*
   dressed somebody (active)

3. $as + \begin{bmatrix} C & CC & C \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^T \Rightarrow$ *aslebes-*
   (causative: got somebody to dress somebody else)

4. $te + \begin{bmatrix} C & CC & C \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^T \Rightarrow$ *teseber-*/broken
   (passive)

5. $a + \begin{bmatrix} CC & CC & C \end{bmatrix} + \begin{bmatrix} a & e & 0 \end{bmatrix}^T \Rightarrow$ *asaber-*
   (participative : participate in breaking)

6. $te + \begin{bmatrix} C & CC & C \end{bmatrix} + \begin{bmatrix} a & e & 0 \end{bmatrix}^T \Rightarrow$ *tesaber-*
   (reciprocal: one broke the other)

7. $a + \begin{bmatrix} C & CC & C \end{bmatrix} + \begin{bmatrix} a & e & 0 \end{bmatrix}^T \Rightarrow$ *asaber-*
   (caused others for reciprocal action)

8. $\begin{bmatrix} C_1 & C_2 & C_2 & C_3 \end{bmatrix} + \begin{bmatrix} e & a & e & 0 \end{bmatrix}^T \Rightarrow$ *sebaber-*
   (repetitive : breaking happened repetitively)

9. $\begin{bmatrix} C_1C_1 & C_2 & C_2C_2 & C_3 \end{bmatrix} + \begin{bmatrix} e & a & e & 0 \end{bmatrix}^T \Rightarrow$ *asebaber-*
   (participative in the repetitive action)

10. $te + \begin{bmatrix} C_1 & C_2 & C_2C_2 & C_3 \end{bmatrix} + \begin{bmatrix} e & a & e & 0 \end{bmatrix}^T \Rightarrow$ *tesebaber-*
    (repetitive reciprocal)

11. $a + \begin{bmatrix} C_1 & C_2 & C_2C_2 & C_3 \end{bmatrix} + \begin{bmatrix} e & a & e & 0 \end{bmatrix}^T \Rightarrow$ *asebaber-*
    (caused others for repetitive reciprocal)

12. $\begin{bmatrix} C_1 & C_2C_2 & C_3C_3 \end{bmatrix} + \begin{bmatrix} I & I & 0 \end{bmatrix}^T \Rightarrow$ *sbr al-/adereg-*
(sudden events/actions: are used to form compound verbs with intransitive *al-* and transitive *adereg-*. *sbr ale* = got suddenly broken, *sbr aderege* = he broke it suddenly)

13. $\begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix} + \begin{bmatrix} e & e & 0 \end{bmatrix}^T \Rightarrow$ *seber al-/adereg-*
(slow actions: the opposite of the previous(12). Verb derivations 12 and 13 have adverbial functions)

Except for some phonological changes and hence letters, the vowel patterns for most verb stems are regular. Some of the patterns are [e,e,0], [a,e,e,0], [e,e,e,0], [a,a,e,0], [e,a,e,0], [a,a,e,0], [e,a,e,0], [a,e,a,e,0] and [e,e,a,e,0]. It is interesting to note the vowels for the stems consist mostly of only 'e','a' and few 'I's. The other vowels appear when these stems are combined with their suffixes.

4.3.4 Derived adverbs

Adverbs in Amharic are very few. Adverbial functions are often accomplished with noun phrases, prepositional phrases and subordinate clauses. One exception to this is the derivation of adverbs from a few adjectives by adding 'Na'.

Example *kfu*/bad + *Na* ⇒ *kfuNa*/badly.

4.3.5 Compound words

There are a considerable number of compound words for nouns, verbs, adjectives and adverbs.

Compound nouns

1. noun + **e** + noun

   Example *bEte mengst*/palace
   bEte krstian/church
   liqe member/chairman

2. noun + noun ⇒ *ayer menged*/ airlines

3. noun + verb ⇒ *alem akef* / international

4. verb + verb ⇒ *arso ader*/farmer

Most non-verb words do not end in the vowel **e**, but when they do, they are part of a compound noun. This is an important indicator for identifying compound words.

Compound adjectives

noun + **e** + adjectives ⇒ *Igere qelal*/fast

Just like the **e** in compound nouns above, the presence of **e** at the end of words is a key indication that the current and the following words form compound words.

Compound verbs

Compound verbs are formed with the verb stems *al-* and *adereg-* as shown in the verb derivation rules 12 and 13.

Example 1 *sbr ale::* (= It got broken suddenly.)

Example 2 *seber aderge::* (= He broke it slowly.)

Another group of compound verbs consists of three words. The first two words are either opposites of each other or duplicates and the last word is the verb *al-* or *adereg*.

Example 1 *bq Tlq ale::* (= He appeared and disappeared.)

Example 2 *weTa weTa ale::* (= He went out repetitively.)

The main indicator words for the presence of compound verbs are *al-* or *adereg-*. If the previous two words are duplicates or antonyms, then they are parts of the compound. A dictionary of antonyms can help us to find out if the previous two words are opposites or the cheapest way would be to see if the previous two words form collocations. If we are sure that not any of the above conditions are true, then we decide that only the left hand word is part of the compound.

Compound adverbs

Compound adverbs are quite few in number and are formed by repeating an adjective.

Example *tnx tnx wha iTeTal::*
(word to word translation: little little water he drinks)

Identifying compound adverbs involves seeing if bigrams of the same elements exist and the following word is not *al-* or *adereg-* (in which case it will be a compound verb).

4.3.6 Pronouns, prepositions and conjunctions

Pronouns, prepositions and conjunctions in Amharic can be individual words or be bound to other words as affixes. They belong to a closed class. No new words are derived from them. In the accusative and genitive, free personal pronouns take the affixes for nouns. The small number of independent pronouns and prepositions can be easily identified by using a dictionary. The bound prepositions are mostly proclitics (prefixes). Conjunctions are enclitics (suffixes). Common proclitics are *le-*(=for), *ke-*(=from), *be-*(=by or with) and *ye-*(=of). The enclitics are -*na*(=and) and -*m*(=also). Looking at the first two letters of a word may be enough to identify the proclitics and enclitics. However, there is a trap. Some words already have these letters as constituting letters.

Example 1 *ketema* (= city)
it starts with *ke-*, but there is no preposition in the word

Example 2 *buna* (= coffee)
it finishes with -*na*, but there is no conjunction in the word

The solution to this problem is to take out the proclitics and/or enclitics and see if the remaining letters form a word that is an entry in a dictionary. If it does, then the clitics are prepositions or conjunctions. If that is not the case, then the word does not have clitics.

4.4 NLP Challenges in Amharic

Amharic poses its own challenges to natural language processing at all levels of linguistic studies: phonology, morphology, syntax, semantics and discourse. With respect to part of speech tagging, the challenges result mainly from the complexity of the morphology , lack of resources and the nature of the writing system.

POS taggers, especially those based on rules, need to do morphological analysis with support of a dictionary. The complete and efficient way to perform these operations is to use finite state methods. The two challenges in modeling the morphology of most natural languages(becomes three for Semitic languages including Amharic) are related to:

- morphotactics
- the phonological/orthographical alternations
- interdigitation (for Semitic languages)

Morphotactics is the study of how morphemes combine together to make well-formed words. The variations /alternations are the discrepancies between the underlying or morphophonemic strings and their surface realization (Beesley, 1998).

A finite state machine is a model of computation that consists of a set of states, a start state, an input alphabet, an accept state and a transition function that maps input symbols and current states to a next state. Computation begins in the start state with an input string. It changes to new states depending on the transition function. If next state is the accept state, the machine has accepted the input symbols. There are many variants of this basic model. Of interest with respect to morphology generation is a machine having outputs associated with transitions. This machine is called Finite State Transducer (FST) and it is this machine that will help us generate information on input symbols (example: all possible tags for a word).

Regular expressions/relations are convenient and concise methods of representing words and their variations. They can then be compiled into FSA/FST. A detailed analysis of these methods as applied to Amharic and Tigrigna are outlined in Gasser (2009*b*).

Outside morphological issues, there is a particular ambiguity issue with respect

to proper nouns in Amharic. Proper nouns that are easily recognized in English by the case of the initial letter cannot be recognized in Amharic as the Ethiopic writing system does not include capital letters. What makes this even more problematic for NLP is the fact that names of most Ethiopian people or even locations have meaning and can be nouns, verbs, adjectives, adverbs and even phrases with all their inflectional variations. Examples 1 and 2 show ambiguities related to proper nouns.

Example 1: adis abeban ywedal::

The last word in an Amharic sentence is a verb which comes right before the 2 colons. In this example, the verb is **ywedal**, which means **he likes**. This sentence can have the following three interpretations in the most unusual contexts.

1. He likes **Addis Ababa**. (**Addis Ababa** is the capital city of Ethiopia)
2. **Addis** likes **Abeba**. (**Addis** and **Abeba** are names of people)
3. He likes a new flower. (**adis** = new, **abeba** = a flower)

Example 2: habtE Tefa::

1. **Habtay** is missing.
   (Habtay is missing. (**Habtay** is a name of a person)
2. I lost my wealth.
   (**habt** = wealth, **E** is possession marker for first person singular I)

In addition to the class of ambiguities related to proper nouns, there is another class of ambiguities related to gemination. The same Amharic written words can be pronounced differently in different contexts as a result of gemination. Gemination happens when a spoken consonant is pronounced for an audibly longer period of time than a short consonant. Even though the Ethiopic writing system does not have symbols for long or short vowels, native speakers do not have difficulty in identifying when the same letter should be pronounced longer or shorter. The importance of gemination distinction cannot be overemphasized in speech synthesis, word sense disambiguation and other applications. Examples 1 and 2 show its importance.

1. **ale** is an Amharic verb which means **there is** or **he said** depending on the length of pronunciation of the second letter **l** (longer or shorter, respectively).

2. **wana** can be an adjective (**main**) or a noun (**swimming**) when **n** is pronounced longer or shorter, respectively.

For part of speech tagging, the absence of gemination markers increases the number of ambiguous words but in another sense, the problem of the same word having different parts of speech is not a new kind of problem.

Chapter 5

Research Methods

- Introduction
- The ELRC Tagset
- Limitation of the Tagset
- The POS Tagged News Corpus
- Preparing the Dataset
- The Tagging Process
- Training and Test Data

5.1 Introduction

In the previous chapters, we discussed the state-of-the-art stochastic tagging methods and Amharic morphology. In this chapter, we will discuss the tagset, the annotated corpus and the issues surrounding the application of the tagging methods on the corpus. The issues include cleaning the corpus, tokenization, feature extraction, training and testing procedures.

5.2 The ELRC Tagset

A part of speech tag is a symbol or an abbreviation that is used to give linguistic information as to the class of a given word in a given language. Knowing a

particular word as a noun or a verb tells us how that word can be used and the morphological changes it may be subjected to. The nature, universality (across languages) and the number of tags are always debatable but their existence in some form is incontestable. Any given language can benefit a lot by categorizing the words of the language into some predefined classes (tags).

The kind of information that each tag carries and the total number of these tags (tagset) depend on the purpose and the given language. For English, for example, there are generally eight word classes as in traditional grammars (noun, verb, adjective, adverb, determiner, preposition, conjunction, interjection). Other more refined and carefully designed models have much larger numbers: 45 for the Penn Treebank (Marcus et al., 1993), 87 for the Brown corpus (Francis, 1980), 64 for the C5 tagset and 146 for the C7 tagset (Leech et al., 1994).

The design of a tagset and the availability of a corpus that uses that tagset determines its importance and its usage. For Amharic, there is a medium-sized POS tagged corpus consisting of 1,065 news articles (210,000 tokens) using a 31 tagset (Demeke and Getachew, 2006).

The tagset design was constrained from the beginning by lack of finance, expertise and time (Demeke and Getachew, 2006). Under such constraints, the primary of objective of the tagset design is to give each word as much basic grammatical information as possible.

The ELRC tagset is based on 11 basic tags, most of which have further been refined to provide more linguistic information, thus increasing the tagset to 31. The tags for nouns are **VN** (**V**erbal **N**oun), **NP** (**N**oun with **P**reposition), **NC** (**N**oun with **C**onjunction), **NPC** (**N**oun with **P**reposition and **C**onjunction) and **N** (for any other **N**oun). There are similar patterns for **V**erbs, **ADJ**ectives, **PRON**ouns and **NUM**erals. Additional tags under the verbs category are **AUX** (for **AUX**iliary) and **VREL** (for **REL**ative **V**erbs). The latter tags lose the distinction between **AUX** and **VREL** when they are attached with prepositions and/or conjunctions. Numerals are divided into cardinals and ordinals represented by the **NUMCR** and **NUMOR** tags. The numeral distinction between cardinals and ordinals is lost too when either is attached with prepositions and/or conjunctions. The rest of the tags are **PREP** for prepositions, **CONJ** for conjunctions, **ADV** for adverbs, **PUNC** for punctuation, **INT** for interjection and **UNC** for unclassified (difficult to classify). Table 5.1 shows a complete list of the tags in the ELRC POS tagset.

Table 5.1: The ELRC POS Tagset

Basic Tag	**Tag**	**Definition**
Noun	VN	Verbal noun
	NP	Noun with preposition
	NC	Noun with conjunction
	NPC	Noun preposition and conjunction
	N	Any other noun
Verb	AUX	Auxiliary verb
	VREL	Relative verb
	VP	Verb with preposition
	VC	Verb with conjunction
	VPC	Verb preposition and conjunction
	V	Any other verb
Adjective	ADJP	Adjective with preposition
	ADJC	Adjective with conjunction
	ADJPC	Adjective with preposition and conjunction
	ADJ	Any other adjective
Pronoun	PRONP	Pronoun with preposition
	PRONC	Pronoun with conjunction
	PRONPC	Pronoun with preposition and conjunction
	PRON	Any other pronoun
Numeral	NUMCR	Cardinal numbers
	NUMOR	Ordinal numbers
	NUMP	Numeral with preposition
	NUMC	Numeral with conjunction
	NUMPC	Numeral with preposition and conjunction
Preposition	PREP	Prepositions
Conjunction	CONJ	Conjunctions
Adverb	ADV	Adverbs
Interjection	INT	Interjections
Punctuation	PUNC	Punctuation
Unclassified	UNC	Unclassified

5.3 Limitations of the Tagset

For a morphologically complex language like Amharic, 31 tags do not give much information to reliably develop applications of machine translation, information retrieval, information extraction and speech synthesis/recognition. Some tags that may be critical depending on the target application are missing. Thirty-one tags may not seem far off from 45 tags of the Penn POS tagset, which gives practically useful information for English. However, in reality the ELRC tagset has only 18 unique POS tags if tagging is done at the level of morphemes instead of words as they appear in a text. Preposition and conjunction affixes that appear with nouns also appear with verbs, adjectives, pronouns and numerals. Therefore, instead of having multiple information in tags such as found in NPC (Noun, Preposition and Conjunction), there will be one tag with one basic information for each morpheme. The the same 31 tagset can then be expressed with the 18 simplified tagset.

The new tagset will look like this P,C,ADJ,ADV,AUX,VREL,V,PRON, CONJ, INT,N,VN,NUMOR,NUMCR,NUM,PREP,PUNC,UNC. From this tagset, it is clear to see the kinds of information missing. Nouns in Amharic have gender and number and they may also come attached with determiners as shown in figure 4.1. Those distinctions are not, however, reflected in the ELRC tags for nouns. In fact, a less important distinction between nouns is made; the POS tag for noun VN indicates that the word is the noun form (starts with $m(e)$) of a verb. Similarly, the ELRC tags for adjectives and verbs do not reflect the variations in gender and number.

Another critical POS tag that is missing from the ELRC tagset is the tag for proper nouns. Identification of names of people and places, which is critical in information extraction, is considered as a noun in the ELRC tagset. The presence of a proper noun tag is even more important in the context of Amharic, where the idea of letter case distinction does not exist and where most Ethiopian names are just normal words in the language. Thus, most proper nouns that are easily recognized in English by the case of the initial letter cannot be recognized in Amharic. These names can be nouns, verbs, adjectives, adverbs and even phrases with all their inflectional variations. (See section 4.4 for illustration of ambiguities related to proper nouns)

5.4 The POS Tagged News Corpus

The part of speech tagged corpus that is used for experiments and evaluations has come from a website dedicated to providing resources to Amharic NLP researchers (http://nlp.amharic.org/resources/corpora-collections/).

Publicly available since June 2008, this corpus has reportedly 210,000[1] tagged tokens of 1,065 news articles collected from Walta Information Center (WIC) from 1998-2002. It was developed by Ethiopian Languages Research Center of Addis Ababa University. The corpus is available in Fidel and SERA (Firdyiwek and Yaqob, 1997). Fidel is the name of the alphabet in which Amharic is written; SERA is the ASCII encoding or transliteration for Amharic alphabets.

The document structure of the corpus is defined in DTD (Document Type Definition) and is shown below:

```
<! DOCTYPE amnews94 [
<! ELEMENT amnews94(document+) >
<! ELEMENT document (filename, title, dateline, body) >
<! ELEMENT filename (#PCDATA) >
<! ELEMENT title (fidel, sera) >
<! ELEMENT fidel (#PCDATA) >
<! ELEMENT sera (#PCDATA) >
<! ELEMENT dateline EMPTY >
<! ELEMENT body (fidel, sera) >

<! ATTLIST dateline place CDATA #IMPLIED >
<! ATTLIST dateline month CDATA #IMPLIED >
<! ATTLIST dateline date CDATA #IMPLIED >
] >
```

The above DTD text defines that 'amnews94' is the root element of one or more documents. Each document has four data elements, namely file name, title, date and body whose values are parsed character data (represented by #PCDATA). The title and the body of each document are represented both in Ethiopic Script (Fidel) and SERA. The 'dateline' with attributes for place, month and date refer to the place, month and year of the reported news. The values for the attributes of the 'dateline' are optional character data (represented as CDATA #IMPLIED).

[1] Actual counting of the tokens gives a number < 210,000

Example of the tagged corpus for a single news article (in SERA) is shown below.

```
<document>
<filename> mes07a2.htm </filename>
<title>
<fidel>
... (title in Ethiopic script)
</fidel>
<sera>
beborena <NP> yeohdEd <NP> tehadso <N>
wyyt <N> tejemere <V> ::<PUNC>
</sera>
</title>
<dateline place="negelE" month="meskerem" date="7/1994/(WIC)/" />
<body>
<fidel>
... (body in Ethiopic script)
</fidel>
<sera>
beborena <NP> zonna <N> 13 <NUMCR> weredawoc <N> lemigeNu <VP>
yemengst <NP> serateNoc <N> yetezegaje <VREL> yeohdEd <NP>
tehadso <N> wyyt <N> zarE  <ADV> mejemerun <VN> yezonu <NP>
mestedadr <N> mkr <N> bEt <N> astaweqe <V> :: <PUNC> yemkr <NP>
bEtu <N> Sehefi <N> ato <ADJ> mehemed <N> jlo <N> IndegeleSut <VP>
leamst <NUMP> qenat <N> bemiqoyew <VP> bezihu <PRONP> tehadso <N>
yeabyotawi <ADJP> dEmokrasiyawi <ADJ> tyaqE <N> beityoPya <NP>
, <PUNC> yeabyotawi <NP> dEmokrasi <N> yelmat <NP> merhowoc <N>
, <PUNC> stratEjiwocna <NC> yesratu <NP> adegawoc <N> bemilu <NP>
rIsoc <N> lay <PREP> wyyt <N> ykahEdal <V> :: <PUNC> yeamelekaketn
<NP> trat <N> lemamTat <NP> bemikahEdew <VP> yetehadso <NP> wyyt
<N> kezon <NP> memriyawocna <NC> kewereda <NP> shfet <N> bEtoc <N>
yetewTaTu <VREL> ke2 xi 500 <NUMP> belay <NP> yemengst <NP>
serateNoc <N> ysatefalu <V> teblo <V> IndemiTebeq <VP> Sehefiw <N>
lewalta <NP> InformExn <N> maIkel <N> gelSewal <V> :: <PUNC>
</sera>
<copyright> copyright 1998 - 2002 Walta Information Center
</copyright>
</body>
</document>
```

The project of developing the tagged corpus started in September 2005 and lasted about four months. A number of organizations and individuals participated in the project. Walta Information Center (WIC) generously made the news documents available for research and Argaw and Asker (2005) preprocessed it and provided an electronic copy to ELRC, where it was tagged. Nine people, most of them from the Center, were involved in the actual manual tagging of the 1065 news documents. One technical assistant and four other administrative support staff were also involved at various levels during the project.

The corpus has limitations with respect to scope(representativeness) and quality. Both these limitations are almost inherent to all linguistic resources. There is no corpus that is 100% representative of a language and there are also limits as to the extent of the quality (human inter-annotator agreement). So the differences between language resources become that of a degree or extent. Some annotated data are bigger and come from different genres. Others may be smaller and come from one or few varieties of text. In addition, annotated data may be of high quality by following strict standards of annotation. Examination of the WIC corpus reveals that it leaves more to be desired on the two criteria. The WIC corpus has only one category (genre) - news, which limits its coverage of the different aspects of the language. It also has quite a few errors and inconsistencies that need to be cleaned up before further processing.

5.5 Preparing the Dataset

Even though 210k tagged tokens were reported for the WIC corpus, the actual number without cleaning the corpus is 200545, a difference of 9455 (about 5%). Part of the reason for this discrepancy is caused by tagging errors. The errors are such that some tokens have multiple tags and other tokens do not have tags at all, which makes us think that they constitute multi-unit tokens. Some punctuation marks (quotation marks and forward slash) are considered as part of some tokens (eg: "bodigardna" < NC >). This kind of error accounts for almost half of the errors. Other errors, probably associated with typing, include some tags without angle brackets <TAG> and so can be mistaken for tokens. Eight headlines and one sentence are not tagged at all or they are just tagged as multi-word units. However, in reality the tag is the correct tag of only the last word in the headline or sentence).

In addition to the aforementioned errors, there are serious inconsistencies with

respect to what constitutes a word and what tags should be assigned for a word under the same contexts. Some inconsistencies could have easily been avoided if clear annotation guidelines were followed. For example, words in collocations, which are sometimes treated as one unit, other times as separate words, could have been treated as one or the other. The inconsistencies with the same words being tagged differently in similar contexts is a common problem in annotating a corpus. However, this kind of inconsistency could have also been minimized just to the most difficult ones if strict annotation guidelines were followed.

Therefore, given the above errors and inconsistencies, it is safe to assume that the inter-annotator agreement, the measuring metric for the agreement between annotators annotating the same text, can be assumed to be low for the WIC corpus.

Any POS tagging method cannot be expected to have less error rates than the fraction of errors or discrepancies introduced by the annotators. Since our objective is to improve performance, the best strategy is to start with a cleaned version of the corpus before designing the tagging method. With this view, an effort has been made to correct as many errors and inconsistencies as possible.

Correction of the simple errors mentioned earlier resulted in an increase of the total number of token-tag pairs from 200545 to 200766. (an increase of 0.11%). Correcting the inconsistencies proved more sophisticated and laborious. The first inconsistency problem is related to tokenization and POS tagging of time, number and name expressions. In some cases, the words are considered as independent tokens. In other cases under similar conditions, they are tagged as multi-word tokens. The following examples show that.

```
1. ... abrikaw <N> 85 neTb 7 miliyen <NUMCR> br <N>
    yeteTara <VREL> trf <N> ageNe <V> :: <PUNC> ...
2. ... yeskWar <NP> mrtoc <N> 85 <NUMCR> miliyen <N>
    733 <NUMCR> xi <N> br <N> ...
3. ... <N> be3 <NUMP> neTb <N> 9 <NUMCR> miliyen <N> br <N> ...
```

From the three extracts from the WIC corpus, we can see that numbers have not been treated in a principled manner. In one extreme case, they have been treated as one multi-word token. In the other extreme, each constituting unit is considered as separate tokens. In other cases, they have been mixed between the two. Though less common, this case is true also with named entities likes names of people and places.

Table 5.2: Distribution of the Number of Tokens in a Single "Word"

# tokens	2	3	4	5	6	7
# instances	4622	485	170	11	2	1

The percentage of multi-word tokens in the WIC corpus is less 3%. There are around 5291(2.6%) tokens that have at least two constituting words separated by space. Words separated by hyphen are not considered as multi-word tokens. Some multi-word tokens had hyphens and space (332). For our purpose, the space has been deleted.

The number of words that make up the multi-word units ranges from two to seven. The distribution of the number of words in multi-word units is skewed towards two. As can be seen in table 5.2 , 87% of the multi-word tokens consist of two words. More than three quarters of these (76%) are noun-associated tags (N, NP and NPC) and 11% are number associated tags. The rest 13% (with more than 2 constituting words) are almost equally numbers and nouns. Here are a few examples that show the nature of the multi-word tokens.

1. ... 1 miliyon1 meto 80 xi 786 <NUMCR> ...
2. ... 6 miliyen 1 meto xi <NUMCR> ...
3. ... 1 miliyen 3 meto xi <NUMCR> ...
4. ... tu si ze sen rayz <N> ...
5. ... 1 miliyen 576 xi 189 <NUMCR> ...
6. ... abdula bin abdul muse al turki <N> ...
7. ... instityut of internaxnal kapasiti biyolding for afrika <N> ...
8. ... sle Ec ay vi Edsm <NPC> ...

It is interesting to note that example 7 is a long institution name in English encoded in SERA. It represents 'Institute of International Capacity Building for Africa'. In the corpus, it is POS tagged as N. This and many others like that should have been tokenized and each word treated as foreign word (FW) or unclassified. Otherwise, the tokenization problem will be more complex and will require named entity recognition techniques.

Outside manual correction, there are a number of alternatives to solving that problem. One solution is to throw out the multi-word tokens. However, this will have a bad effect as it will introduce noise in the syntax (context) of the language. Leaving out the sentences that contain them is also not a good idea as the corpus is already small and so it will make it even smaller (increase data sparsity). The good solution is to tokenize them on space and give the constituting words tags

that together they had in the first place. This has the advantage of decreasing sparsity (improving the language model). This can be done easily automatically. One trap that we should avoid falling into is that when prepositions and conjunctions are attached with the beginning or last words in the multi-word tokens, then the middle words should have tags with the preposition and/or conjunction tags stripped off because in Amharic, prepositions and conjunctions are attached with the beginning or last words of the multi-word tokens. Applying the suggested method increases token-tag pairs from 200766 to 206929 (an increase of 3.07%). It must be noted, however, that this method will not solve all problems as there are tokens of a unit already attached together for no reason (eg: 2xi represents 2 thousands and should have been written as 2 xi).

The second inconsistency problem is related to tokens receiving multiple tags under the same conditions. An attempt has been done to identify and correct them. One technique that we have used it to list all the tokens and the frequency of its association with each tag it is assigned. A closer examination of this list for a given word reveals that some tags are wrongly assigned. The following examples show how many times a particular tag has been assigned to a given word.

```
"    [('PUNC', 502), ('N', 8), ('NP', 3),
     ('ADJP', 2), ('AUX', 1), ('VREL', 1)]

,    [('PUNC', 3543), ('N', 3), ('NP', 2), ('NPC', 2)]

10   [('NUMCR', 121), ('NUMP', 2), ('NUMC', 1),
     ('NUMPC', 1), ('PREP', 1), ('PUNC', 1)]

13   [('NUMCR', 45), ('N', 1), ('NP', 1), ('NUMOR', 1)]

bemehonum    [('NP', 118), ('NPC', 72), ('CONJ', 61), ('PRONP', 8),
             ('VP', 6), ('VN', 4), ('VPC', 3)]

Indihum      [('PRONP', 503), ('PRONPC', 21), ('PREP', 4),
             ('PRON', 3), ('ADV', 2), ('PRONC', 2)]
```

The punctuation mark (") has been tagged as <PUNC> correctly 97% of the times. In the rest 3%, it has been assigned the wrong tags. Similarly, the punctuation mark (,) is correctly tagged as PUNC in 99.8% times, but it is also tagged incorrectly in few other instances. Such errors are not limited to few cases, but in fact, most frequent multi-tag tokens have some extra tags assigned incorrectly

Table 5.3: Ambiguity Distribution in the WIC Corpus

# tags per token	# tokens	Percentage
1	153211	74.04
2	34107	16.48
3	10544	5.10
4	4122	1.99
5	2057	0.99
6	1393	0.67
7	941	0.45
8	427	0.21
9	127	0.06

infrequently.

An effort has been made to correct some of the errors. About 552 tokens have been correctly retagged as prepositions and 893 tokens as nouns, verbs and their variants. Similarly, about 980 numbers and punctuation marks have also been correctly tagged. For multi-tag tokens, token-tag pair with frequency of appearance of one has been replaced by the tag with the highest frequency of at least 10 ($\approx$ double the average frequency of each word in the WIC corpus). With this method, 1209 tokens have been retagged with the tag of the highest frequency.

Before using the corrected data to train and test algorithms, let's see how hard tagging is in Amharic. On average, how many words have different tags?

Statistics of the WIC corpus
Number of sentences = 8067, each sentence contains about 25.7 words
Number of unique word-tag pairs = 37486
Type token ratio = 5.5, each word type occurs on average 5.5 times
The number of word types = 32480
Ambiguity ratio = count(unique token-tag pairs)/ count(unique tokens) = 1.2

Table 5.3 shows the number of the same tokens having multiple tags ranging from one through nine. From the table, it can be seen that 74% of the tokens have only one tag, a little less than double of what it was before correction (38%). 23% of the tokens have from 2 to 4 tags, a reduction by half (50%). Only 2% of the tokens have 5 or more tags. Before correction, it was 8%.

The tag distribution is skewed towards noun and noun variants. Graph 5.1 shows the counts of each tag type in the WIC corpus. N, NP, VN ,NC and NPC make up 58% of the tags for the tokens. Verb related tags make up 18% of the tokens.

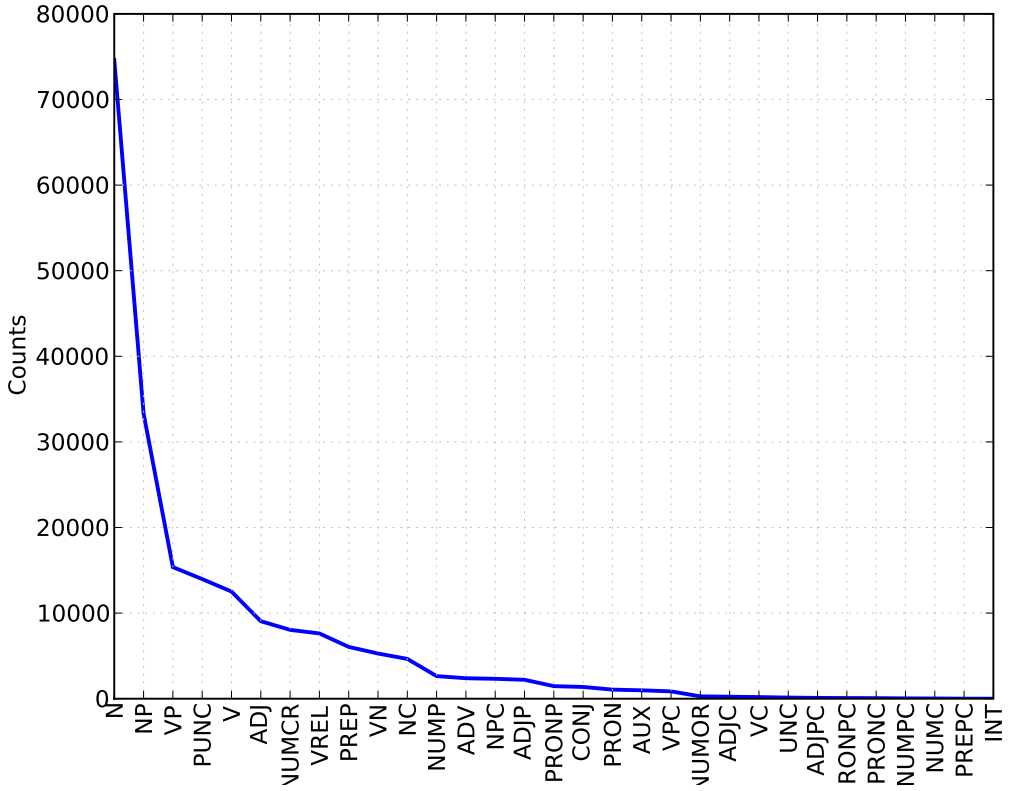

Figure 5.1: Tag Distribution in the WIC Corpus

5.6 The Tagging Process

For the experiments in this dissertation, we use customizable open source implementations of some selected machine learning algorithms, notably CRF++ (Kudo, 2007), LIBSVM (Chang and Lin, 2001) and NLTK (Loper and Bird, 2002). CRF++ is a C++ open source implementation of Conditional Random Fields (CRFs) for segmenting/labeling sequential data. LIBSVM is a C++ and Java implementations of support vector based classification and regression algorithms. It is interfaced with many languages. LIBSVM comes with a Python interface (via SWIG), which we used for our experiments. NLTK, Natural Language Toolkit, is a suite of open source program modules (with documentation) covering symbolic and statistical natural language processing algorithms. It is easy to learn, use and modify. TnT and Brill implementations in NLTK have been used for POS tagging experiments in this dissertation.

Each of these tools, albeit critical, is only one aspect of the stochastic tagging process, which usually involves, either explicitly or implicitly, the following tasks.

Step 1: Tokenization It is a necessary preprocessing step in all tagging systems and it is used to identify the basic units in the language before a set of possible tags can be assigned. The natural text is divided into units. Here, we have used the term units instead of words in order to avoid confusion. In English, the units are usually the words which are separated from each other by space. For Amharic, it can be more complicated than that. The choice of units for Amharic will determine the tagset and tokenization algorithm.

Step 2: Feature Extraction Unlike in rule-based systems where rules and a lexicon are used to assign the correct tags, in probabilistic tagging methods, features are extracted from the annotated data to learn a tagging model. The kind and number of features are chosen carefully to achieve high accuracies. We will discuss the features used in our POS tagging experiments for Amharic.

Step 3: Disambiguation Words may receive multiple tags but only one is correct. The disambiguation step attempts to find the correct tag for every token. We will apply a number of machine learning algorithms whose brief theories have been discussed in chapter 3, namely conditional random fields, support vector machines, TnT (HMM-based) and Brill tagging.

5.6.1 Tokenization

The process of breaking up a text into its constituent meaningful units is known as tokenization or segmentation. Tokenization can occur at a number of different levels: a text could be broken up into paragraphs, sentences, words/morphemes, syllables or phonemes. The algorithms used for tokenization will vary depending on the level of tokenization, the language given and the purpose. For part of speech tagging, sentence and word or morpheme tokenizations are relevant.

Segmentations of Amharic text into sentences and words are not hard problems. Unlike English where identifying the period does not help much in marking the end of a sentence, two colons are sufficient to find the end of a sentence in Amharic.

Like English, Amharic sentences can be segmented into words by using the space character. However, the meaning of words for English and Amharic are not the same. English words as seen separated by space usually have one POS associated with each. By contrast, Amharic words as seen separated by space represent concatenated morphemes, with each morpheme capable of having its own POS.

In general, Amharic words consist of zero or more prefixes, one or more space separated stems/words and zero or more suffixes. The prefixes and suffixes are usually members of a closed class which include prepositions and conjunctions. These affixes are finite in number. The following regular expression summarizes it all.

$$\text{Amharic word} = \text{prefix}^{*}[\text{STEM}|\text{WORD}]^{+}\text{suffix}^{*}$$

Given the form of Amharic words as shown above, three approaches can be adopted in segmenting the words, some of which are the intermediates of the other two extremes.

The first approach is taking the space separated elements as words consisting of only one big 'morpheme'. This decision will determine the nature of POS tags and the tagging algorithms. If words that have prepositions and/or conjunctions are considered as the units of tokenization, then the tagset will be complex and will not conform to conventional tags like NN for nouns, JJ for adjective, etc. The tags will become, instead, more complex so that they can represent 'words' using such tags as NP or NPC (Noun with preposition or noun with preposition and conjunction). This has also the effect of the tags being more determined by the nature of the given language than by universal abstract tags. Another disadvantage of this approach is that a usage of dictionary will severely be restricted as all words in a text cannot possibly be in the dictionary of the language. So that

means if we implement a rule-based system, the need for morphological analysis will be difficult to avoid. The advantage of this approach is its simplicity; it is simple to implement. All that is needed to tokenize a text is the identification of the space as the delimiter of tokens.

The second approach is separating the words into their smallest constituting morphemes, which are generally agreed to be the smallest meaningful units. This approach too has significant effects on the design of a tagset and the tokenization and tagging algorithms. The tagset will constitute more tags that are less dependent on the given language. So the tagset of a given language can be compared with that of another language, which by itself can be important for language universals studies. The tagging algorithms for this tagset can make effective use of a dictionary or tagged corpus. Data sparsity will be less of a problem than in the first approach. The main disadvantage of this approach is that the tokenization by itself is a hard problem. Identifying the morphemes that constitute the words by itself becomes a tagging problem. There will be many more ambiguities. Does 'Windows' have one morpheme or two morphemes? The answer does not come with an easy algorithm. Because 'Windows' can be a word referring to Microsoft Windows or it could be just the plural of window at the start of a sentence.

The third approach is choosing somewhere in between the two extremes. This means for some selected morphemes, there will be tokenization and for others we leave them as part of the word. As can be imagined, this too determines the tagset design and the tagging algorithms.

The level of difficulty in identifying the morphemes is not the same in all approaches. The second and third approaches will have to deal with many more ambiguities of identifying the morphemes. All the three approaches should deal with the problem of identifying compound words or idioms in a principled manner.

The kind of approach chosen determines the design and development of the tagset. Taking the first approach, Ethiopian Languages Research Center (ELRC) of Addis Ababa University have designed a tagset of 31[2] elements (Demeke and Getachew, 2006). The same approach is used here except that multi-word tokens are consistently treated as separate units for the sake of simplicity.

[2]30 is reported, but actual counting gives 31

5.6.2 Feature Extraction

After tokenization comes the need for extracting the word and context features of a given unit in a format suitable for classification algorithms. The input to the tagging algorithms are represented in terms of values for attributes and the attributes are used to encode information about morphological, syntactic and semantic clues that are usually used by linguists to determine the category of a word. The informative attributes for POS tagging a text of a given language are the following:

Morphological: Words of different POS tags have different affixes associated with them. For example, plural nouns in Amharic usually end in **oc**. This knowledge helps us guess the word class of a word that ends in **oc**. The vowel patterns in the words as shown in chapter 4 also are indicative of the POS tags.

Syntactic: Unlike English, Amharic has much of the information about syntax in the words themselves with words usually placed in SOV order. Even so, there is considerable information to be gained from positions of words in a sentence. For example, the main verb in Amharic is always at the end of the sentence unless the text is a poem. Adjectives come before nouns. Adverbs come before verbs.

Semantics: Even though semantics is hard to formalize, words by virtue of their meaning and functions in the language are given different POS tags. For example, prepositions or closed class words in general do not undergo morphological changes and their positions in a sentence can be ambiguous. So, in such cases the semantics can be used to determine the lexical category of words.

Here are the features that capture most of the morphology, syntax and semantics of a given word extracted from a window of 5 words from a given Amharic text.

- the current word, the previous/following word, the word before/after the previous/following word {String}
- prefixes and suffixes of length $\leq$ five {String}
- vowel patterns{String}
- radicals (consonants){String}

- is punctuation(word){True, False}
- has punctuation(word){True, False}
- is alphabetic(word) {True, False}
- is alphanumeric(word){True, False}
- is digit(word){True, False}
- has digit(word){True, False}
- has **e** suffix(previous word){True, False}

The novel features are the vowel patterns and radicals.

5.6.3 Disambiguation as Classification

Given the features of a given word, the next step is to assign it the correct tag. Classification in machine learning is the task of assigning the correct class label for a given input. In this sense, part of speech tagging can be seen as a classification problem where the classes are the different POS tags and the inputs are the words represented as a set of feature values. This way of looking at the problem enables us to use some of the state-of-the-art classification algorithms.

There are a number of machine learning algorithms but they all fall into one of these: supervised, unsupervised and semi-supervised algorithms. This categorization is made based on the availability and usage of labeled or unlabelled data. Any machine learning algorithm requires some kind of data for training. The training examples used in the learning task consist of a set of data records, which are described by a feature vector $X = x_1, x_2, x_3, \ldots x_n$ where n is the number of features or attributes. The dataset has also a special target attribute O, which can have values that are either discrete or continuous.

Given a dataset D, the objective of learning is to produce a classification/prediction function to relate values of features in X and values in O. The learned function/classifier can then be used to predict O values of future data. If the O values are known during training, it is called supervised learning. If the O values are unknown and the learning algorithm needs to automatically generate them, it is called unsupervised learning. The third learning category relies on a small labeled data and a large unlabeled data and is called semi-supervised learning.

Based on whether the output O values taking on continuous or discrete values, supervised learning is divided into regression and classification. For part of speech tagging, the O values are discrete as there are only a finite number of them.

The dataset D is divided into training data and test data. The training data is used by a learning algorithm to learn a model. The test data is used to assess the model accuracy. In the following series of experiments, we have performed 10-fold cross validation.

5.7 Training and Test Data

The annotated data available for our experiments is the WIC corpus, which has about $\approx 207k$ tokens. How much of it and which portion of it should be used for training and testing are important questions to answer as the variations in these tasks determine the performance of algorithms. Trained on the given portion of the corpus and tested on the remaining portion, two or more supervised machine learning algorithms will usually perform differently for many reasons. Some of the reasons are attributed to randomness and others to the power of the algorithm. It is important to distinguish the sources of variations so that we know better which algorithms actually perform better. Here are some possible sources of variation that must be noted (Dietterich, 1998).

The first source of variation is randomness of the test data. Two learning algorithms may perform differently on a given randomly-selected test data even though, on average, they both perform identically if they are tested many times using each time a randomly-selected data. The second source of variation comes from the variation in the selection of the training data. The given algorithms may differ in performance on a given randomly-selected training data even though, on average, they achieve the same accuracy. The third source of variation is dependent on the inner workings of algorithms. Some learning algorithms are initialized to random state or default values. The fourth source of variation is associated with the errors in the test data itself. If the test data has x fractions of mislabeled data, then any learning algorithm will not achieve an error rate of less than x fraction. Probably, these are not the only sources of information. There could be others including the sensitivity of the learning algorithms to noise, ie. errors in the training data.

In this dissertation, to make our results more consistent and the comparisons between algorithms more reliable, k-fold cross-validation and statistical tests

Table 5.4: 10-fold Cross Validation Data

	# Tokens		# Test Tokens	
Fold	**Training**	**Testing**	**# Known**	**# Unknown**
1	186406	20523	17927	2596
2	185832	21097	18581	2516
3	186724	20205	18043	2162
4	186154	20775	18458	2317
5	186500	20429	18081	2348
6	186719	20210	18108	2102
7	185788	21141	18795	2346
8	185372	21557	19132	2425
9	186615	20314	18085	2229
10	186251	20678	18444	2234
Average	**186236.1**	**20692.9**	**18365.4**	**2327.5**

are applied. Specifically, a 10-fold cross-validation is applied. In 10-fold cross-validation, the whole corpus is partitioned into 10 samples and each sample is kept for testing while the rest 9 (joined together) are used for training. In other words, training is done 10 times each time one sample left out for testing. The ten results are then averaged to give a final result. The advantage of this method is that each sample is used for both training and testing in the repeated processes of training and testing.

A good statistical test should conclude that the two algorithms are different if only if their average performances are different without being fooled by the above sources of variation. Statistical calculations can answer this question: If the algorithms really have the same average accuracies, what is the probability of observing such a large difference or larger between observed average accuracies for the 10 experiments? The answer to this question is called the P value. The P value is the probability of observing a difference as large or larger than observed if the null hypothesis were true. The null hypothesis simply states that there is no difference in accuracy between the algorithms.

The ($\approx 207k$) sized corpus consists of 8067 sentences. The corpus is divided into training and test data. The training data is the 90% portion of the data and the remaining 10% is the test data. Table 5.4 shows the number of tokens in the training and test sets. The numbers of tokens in each fold is not the same because the partition is made at a sentence level. Except for the last fold, which has 804 sentences, each fold has 807 sentences. Each fold is also divided into known tokens and unknown tokens. About 11.25% of the test tokens are unseen in the training data.

Chapter 6

Results

- Introduction
- Evaluation Methods in NLP
- Evaluation Metrics
- POS Tagging Results
- Results Analysis

6.1 Introduction

Natural language systems are designed and developed to perform specific tasks as required and expected by users or other systems. A machine translation system is expected to give a correct translation for a given input. An information retrieval system (search engine) is expected to retrieve correctly ranked relevant documents. Similarly, a part of speech tagger is expected to assign a correct tag to a given instance of a word. In general, for a given input, the NLP system is expected to give a correct output. What constitutes correct output and how we can measure it is, however, not an easy task and so is an active area of research in natural language processing. For example, given that two human translators do not translate the same French text into the same English text, how can a translation produced by a machine be measured for correctness? The answer is not trivial. Similarly, how can we measure the correctness of a POS tagger, which is naturally an easier problem than machine translation? Apart from output correctness, there are other issues to raise about NLP systems: how easy are

they to use by non-experts, how well do they plug into other components, can they be ported and maintained by someone who did not participate in the system development?

Therefore, raising one or more questions of accuracy, user-friendliness, efficiency, modularity, portability, robustness and maintainability is important depending on the purpose. Evaluation is the process of measuring one or more of the above qualities of an algorithm or a system. It has an important role in natural language processing for both system developers and technology users. With evaluation, system developers are much better equipped with the knowledge of what components to improve in the system in order to achieve the desired goals. It helps researchers communicate their results and compare them with previous research work. For users, evaluation provides them with the necessary information they need to easily compare alternative systems and choose the one that meets their requirements.

In this chapter, we will discuss the kind of evaluation methods used in reporting and analyzing POS tagging results.

6.2 Evaluation Methods in NLP

Given the importance of evaluation for both system developers and end users, it is necessary to clearly formulate the evaluation methods. Depending on the evaluation procedures, there are different kinds of evaluation techniques (Palmer and Finin, 1990). In the following subsections we will discuss the different approaches available in evaluation methods.

6.2.1 Intrinsic vs. extrinsic evaluation

NLP systems consist of a number of subsystems, each subsystem having its own input/output pairs. The performance improvements made in the whole system with changes in any one subsystem can be evaluated into two ways. The first one is to evaluate the subsystem as an isolated system using pre-defined input/output pairs or gold standard. This approach is called intrinsic evaluation. The second approach is to characterize the subsystem performance in terms of its function in a more complex system. This method is called extrinsic evaluation. The extrinsic performance of the system is then characterized in terms of its utility with respect

to the overall task of the complex system or the human user. For example, these two methods can be applied in part of speech tagging. A POS tagger can be evaluated as an isolated system or as a part of another bigger system, such as a parser. An intrinsic evaluation would run the POS tagger on some POS-tagged data, and compare the system output of the POS tagger to the gold standard (correct) output. An extrinsic evaluation would run the parser with different POS taggers and observe the performance changes in the parser with changes in the POS tagger.

6.2.2 Black-box vs. glass-box evaluation

Black-box evaluation is primarily focused on what a system does (Palmer and Finin, 1990). It measures system performance on a given task in terms of well-defined input/output pairs. The system performance is expressed in terms of parameters that refer to accuracy, user-friendliness, speed, reliability, efficiency, portability and maintainability. Most important of all these parameters is usually the accuracy. The accuracy refers to how well the system output matches with the gold standard. The reason the evaluation technique is called black box is that it does not require understanding the inner workings of the system for evaluation.

Glass-box evaluation, by contrast, examines the design and inner workings of the system and so is carried out by system developers. It evaluates the linguistic resources it uses (quality, coverage), the algorithms that are implemented, their efficiency, etc. In other words, glass-box evaluation examines the relevant linguistic theories and how well they are implemented. Given the complexity of the approach and NLP systems, it is usually difficult to measure performance only on the basis of glass-box evaluation. The advantage of this approach is that it is more informative with respect to error analysis and future modifications and improvements of the system.

A black-box evaluation of a subsystem of a system can be considered as part of the glass-box evaluation of the whole system. For example, if part of speech tagger is a component of a bigger system. We can develop a test set that has input/output pairs to measure the performance of the tagger (black-box evaluation). Since it is an evaluation of a component that cannot perform an application independently of others and since it will give information about the component's coverage that is independent of the coverage of any system in which it might be embedded, this can be seen as providing glass-box information for such an overall system.

6.2.3 Automatic vs. manual evaluation

Evaluation can be done automatically or manually. Automatic evaluation refers to comparing the output of an NLP system with the gold standard. With this approach, we can do evaluation as many times as we change the system as long as we have the gold standard. However, the cost of building the gold standard is high both in terms of time and human resource requirement. The gold standard is built over a long period of time by trained humans. One difficulty in building the gold standard is the agreement between annotators with respect to how they deal with complex annotation tasks. They do not usually agree as is, for example, evident in the part of speech tagged corpus used in this dissertation.

The alternative to automatic evaluation is manual evaluation, which is performed by human judges. These human judges are instructed to estimate one or more qualities of the output of a system based on a number of criteria. Here too, the judges will not give the same ratings to the same outputs. There is bound to be variations. Manual evaluation is expensive and cannot easily be repeated as fast as we can as in automatic evaluation.

6.3 Evaluation Metrics

In this dissertation, we will do mainly an intrinsic, black-box and automatic evaluation. We evaluate the different POS tagging algorithms as isolated systems (intrinsic). Within the isolated system, we are going to do black-box evaluation as we will only compare the outputs of the system for given inputs with the gold standard. This intrinsic black-box evaluation is done automatically using the gold standard instead of human judges.

The most commonly used evaluation metrics in POS tagging are accuracy, error rate, precision, recall and f-measure.

Accuracy is the ratio of the number of correct outputs to the total number of outputs for given inputs. Error rate is the ratio of errors to the total number of outputs.

Mathematically,

Accuracy = number of correct outputs/ number of total input-output pairs

Error rate = number of incorrect outputs/number of total input-output pairs

Precision and recall are concepts first widely used in evaluation of information retrieval systems. In information retrieval, precision is defined as the number of relevant documents retrieved by a search engine divided by the total number of documents retrieved by that search, and recall is defined as the number of relevant documents retrieved by a search divided by the total number of existing relevant documents including those which should have been retrieved.

Precision and recall can be used in evaluating POS tagging as well. In this case, precision is the number of items correctly labeled as belonging to the class of interest (**t**rue **p**ositives) divided by the total number of items labeled correctly or incorrectly as belonging to that class (**t**rue **p**ositives + **f**alse **p**ositives). Recall, by contrast, is the number of items correctly labeled items (**t**rue **p**ositives) divided by the total number of items that actually belong to the class which includes items not correctly identified as belonging to that class (**f**alse **n**egatives).

Mathematically,

$$P = \frac{tp}{tp + fp} \tag{6.1}$$

$$R = \frac{tp}{tp + fn} \tag{6.2}$$

Precision and recall are inversely related. When one is increased, the other decreases. For example, in part of speech tagging, if we are focusing on nouns and we say all words are nouns, then we get a 100% recall, whereas the precision decreases significantly as there are many words that are not nouns. Similarly, if we classify words that we know for sure are nouns as nouns and classify the rest as non-nouns, then precision is 100%, but recall decreases.

Usually, precision and recall are combined to give one value called the f-measure. The f-measure is the weighted harmonic mean of precision and recall. Mathematically,

$$F_1 = \frac{2PR}{P + R} \tag{6.3}$$

Equation 6.3 is the harmonic mean of precision and recall, where both measures are given equal importance.

Related to the concepts of precision and recall are type I and type II errors. Type I error (false alarm rate) is the error associated with classifying items as belonging

to a given class when they are not. For example, a verb may be classified as a noun, when actually it is not a noun. Type II error (miss rate) is the error associated with classifying an item that actually belongs to a given class as not belonging to that class. For example, a noun misclassified as a verb when it is a noun is a Type II error. Equation 6.4 shows the mathematical relationships between the two errors and precision and recall.

$$\begin{aligned} \text{Type I} &= 1 - \text{precision} \\ \text{Type II} &= 1 - \text{recall} \end{aligned} \tag{6.4}$$

Two or more algorithms can be compared on one or more of the above evaluation metrics. For part of speech tagging, accuracy is the most informative of them all but the other measures also have their importance as we will see in the following subsections.

6.4 POS Tagging Results

CRF++, LIBSVM, Brill and TnT[1] have all been applied for our POS tagging experiments using the WIC corpus. For good comparison, CRF and SVM are treated together as they use exactly the same features. Similarly, Brill and TnT are also treated together as they are similar in terms of their dependence on neighboring words/tags and their mechanisms in handling unknown words.

6.4.1 Baselines

The simplest tagger that can serve as a baseline in Amharic part of speech tagging is to tag all new tokens as **N**, which is the most frequent tag in the WIC corpus. This achieves an accuracy of about 36%. This is too low to be used as a baseline as most algorithms have much higher accuracies. Another baseline is assigning the most frequent tag of every word seen in the 90% of the training corpus and assigning **N** to unseen words. This achieves about 81% accuracy on the remaining data (10%). All the algorithms applied in this dissertation achieve much higher accuracies than 81%.

[1]Brill and TnT are available in NLTK

Table 6.1: POS Accuracy Results Achieved by CRF

Tested on	Known tokens	Unknown tokens	Overall
Fold1	91.00	78.35	89.40
Fold2	90.03	77.70	88.56
Fold3	91.10	77.70	89.67
Fold4	93.12	80.84	91.75
Fold5	94.18	84.03	93.01
Fold6	91.40	78.50	90.06
Fold7	94.49	85.64	93.51
Fold8	95.26	85.53	94.16
Fold9	94.12	83.85	92.99
Fold10	88.06	73.05	86.43
Average	**92.28**	**80.52**	**90.95**

Table 6.2: POS Accuracy Results Achieved by SVM

Tested on	Known tokens	Unknown tokens	Overall
Fold1	90.75	78.97	89.26
Fold2	89.74	77.50	88.28
Fold3	90.51	77.70	89.14
Fold4	92.25	80.70	90.97
Fold5	93.08	84.16	92.06
Fold6	91.17	78.54	89.86
Fold7	93.48	85.51	92.60
Fold8	94.41	86.10	93.47
Fold9	93.45	83.54	92.36
Fold10	87.87	73.23	86.29
Average	**91.67**	**80.59**	**90.43**

6.4.2 CRF++ and LIBSVM

Both CRF and SVM have been trained and tested on the same dataset using exactly the same features. Parameters have also been selected for both. The critical parameter in both cases is the penalty parameter C. A too small value for C causes underfitting and a too large value causes overfitting. In other words, a small value for C will allow a larger number of training errors, while a large value will minimize training errors. For CRF, $C = 0.05$ and for SVM, $C = 0.5$ have been experimentally found to give higher accuracies. These smaller C values have been chosen for a good reason. The WIC corpus has a number of training errors and so using a larger C can only make the algorithm learn the errors too. A smaller C value basically ignores some errors. An additional critical parameter for SVM is the kernel type. Here, the LINEAR[2] kernel has been found to give

[2]Other kernels tried did not improve performances much and required more parameter space searching.

higher accuracies.

On a 10-fold cross-validation, CRF achieves an average accuracy of 90.95%, while SVM achieves 90.43% under exactly the same conditions. The difference might seem too little, but a statistical significance test proves otherwise. This difference of 0.52 can happen by chance once in thousands, which is less than 0.05 (the conventional level of significance).

The accuracies of both algorithms for each fold are shown in tables 6.1 and 6.2. As can be seen from the tables, SVM achieves a slightly higher average accuracy of 80.59% than SVM (80.52%) on unknown tokens, which leads to the conclusion that SVM generalizes better. However, a statistical significance test shows that the difference is too little to reach that conclusion ($0.461 > p = 0.05$). On the other hand, CRF achieves relatively higher on known tokens which explains its slight overall higher accuracy.

6.4.3 Brill and TnT

As tables 6.4 and 6.5 show, Brill and TnT taggers achieve average accuracies of about 87%, which is 3% less than CRF and SVM. The tables also show the accuracies of each tagger on different folds for known and unknown tokens. On average, Brill achieves an accuracy of 87.41%, 0.32% higher than TnT (highly statistically significant) but both achieve the same average accuracy on unknown tokens. This is to be expected as they have been designed in these experiments to use the same techniques for handling unknown words. The same simple regular expression tagger that POS tags based on affixes has been used both in Brill as part of the initial state tagger and in TnT as part of the unknown words tagger.

The reason Brill performs better on average is because it has significant higher performance on tagging known tokens (91.90% against 91.54%). This is also to be expected given that TnT depends on using the statistics of previous two tags and the association of words and tags, while Brill uses much more information from the left and the right neighboring tags and words.

The Brill tagger has two important parameters: the maximum number of rules and the minimum score. The values for these parameters must be chosen carefully by experimenting. Table 6.3 shows the relative increases and decreases of these parameters and the resulting performance. As can be seen in the table, increasing and decreasing both parameters too much decreases performance. For

Table 6.3: Brill Tagger Minimum Score and Rules Trade-off

Minimum	**Max. # of Rules**			
score	50	100	150	200
3	87.39	87.38	87.38	87.38
4	87.40	87.40	87.40	87.40
5	87.40	87.41	87.41	87.41
6	87.41	87.41	87.41	87.41
9	87.39	87.39	87.39	87.39
12	87.39	87.39	87.39	87.39
15	87.39	87.39	87.39	87.39

Table 6.4: Best Brill Tagger Results: Min-Score = 6, Max-Rules = 50

Tested on	**Initial tagger**	**Known tokens**	**Unknown tokens**	**Overall**
Fold1	85.51	90.91	48.77	85.58
Fold2	85.00	89.82	50.60	85.14
Fold3	86.48	90.89	50.56	86.57
Fold4	87.64	92.38	50.32	87.69
Fold5	88.74	93.37	53.58	88.80
Fold6	87.29	91.15	54.14	87.30
Fold7	89.63	94.01	54.94	89.67
Fold8	89.85	94.55	53.03	89.88
Fold9	89.47	93.80	54.69	89.51
Fold10	83.92	88.13	49.15	83.92
Average	**87.35**	**91.90**	**51.98**	**87.41**

example, for minimum score of 3 and maximum number of 50 rules, the average accuracy is 87.39% and for the minimum score of 15 and maximum number of 200 rules, the same performance is obtained. Intermediate values usually have better performances. Table 6.4 is one best combination for the given possibilities. It achieves an accuracy of 87.41% with minimum score of 6 and a maximum number of 50 rules. Other combinations with the same performance may have more rules or lower minimum score, which makes the training slower and the tagger more complex.

One of the interesting features of Brill tagging is that we can see which rules are contributing the most to improving the tagging accuracies. Brill tagging as discussed in chapter 3 has transformation templates which examine the neighboring words and tags. One of the interesting rules it formed from the WIC corpus is related to tagging the Amharic word **adis** (= new), which is usually used as adjective and is tagged as such by the initial stage tagger. However, when it is followed by **abeba**[3], it should be tagged as noun. Brill has been able to learn that automatically. Here is an extract example of that rule and a few others that

[3]**adis abeba** is the capital city of Ethiopia

Table 6.5: Accuracy Results Achieved by HMM (TnT)

Tested on	Known tokens	Unknown tokens	Overall
Fold1	90.60	48.77	85.30
Fold2	89.56	50.64	84.92
Fold3	90.36	50.60	86.11
Fold4	92.11	50.37	87.45
Fold5	92.98	53.49	88.44
Fold6	90.80	54.09	86.98
Fold7	93.63	54.99	89.34
Fold8	94.14	53.03	89.51
Fold9	93.44	54.69	89.19
Fold10	87.82	49.15	83.64
Average	**91.54**	**51.98**	**87.09**

have been produced by the Brill tagger.

```
          B      |
  S   F   r   O  |     Score = Fixed - Broken
  c   i   o   t  |  R  Fixed = \# tags changed incorrect->correct
  o   x   k   h  |  u  Broken = \# tags changed correct->incorrect
  r   e   e   e  |  l  Other = \# tags changed incorrect->incorrect
  e   d   n   r  |  e
-----------------+-------------------------------------------------
 39  39   0   0  | ADJ -> N if the following word is 'abeba'
                 |
 17  17   0   1  | NUMCR -> N if the following word is 'aleqa'
                 |
 11  28  17   0  | NP -> ADJP if the following word is 'melk'
                 |
  8  18  10   0  | ADJ -> N if the following word is 'mesqel'
```

6.5 Results Analysis

In addition to the accuracy results just reported in the previous section, precision and recall can also be used to examine closely the performance of the algorithms with respect to each tag. Table 6.6 shows the precision, recall and f measure for the two best performing algorithms (CRF and SVM). The average precision, recall and f measures of SVM are slightly larger than the corresponding values for

CRF but a statistically significance test shows that the difference is not significant ($p = 0.75, 0.08, 0.44 > 0.05$).

Tags INT (for interjection) and PREPC (preposition with conjunction) are not predicted by both tagging models even wrongly, hence 0 values for recall and undefined for precision[4] and f measure[5]. In addition, CRF did not predict NUMC (number with conjunction) even by mistake. In both models, punctuation marks have been identified correctly 100% of the times. CRF predicted 76% of the adjectives correctly as adjectives and of those words it tagged as adjectives, 86% are correctly adjectives and 14% are non-adjectives. The corresponding values for SVM are 77% and 83%. Using the same table, similar comparisons can be made for other tags too.

Precision and recall values of table 6.6 may be important for comparing the tagging models based on their performance for each tag but it is less useful for understanding the most performance decreasing errors. The same table shows that PRONC (pronoun with conjunction) is not identified in 99% of the cases and it seems huge. However, its effect to performance decrement is little compared to nouns which are identified 97% of the times. The highest performance decreasing errors can be seen from a table of confusion usually called confusion matrix.

Tables 6.7 and 6.8 show confusion matrices for both CRF and SVM. Confusions between nouns and other tags account for most of the errors in both tagging models. More than 44% of the errors in CRF resulted from taking non-nouns and their variants (ie: NP,NC, and NPC) as noun families. The corresponding percentage for SVM is a little less (39.05%). From these confusions, the bigger portions are taken by confusions between nouns and adjective families, which account for more than 19% in CRF and 18% for SVM.

In both tagging models, non-noun families are taken to be noun families more than the other way round. For example, in CRF, 7.7% of the error rates resulted from confusing **ADJ**s for **N**s, whereas 4.51% resulted from confusing **N**s for **ADJ**s. The corresponding values for SVM are 6.59% and 4.89%. This should not come as a surprise if we closely examine the morphology of the words. The same affixes are shared by noun families and most of the non-noun families.

A noun phrase that consists of only the head noun gets affixes such as prepositions, definite article, and the case marker. However, if a noun phrase contains prenominal constituents such as adjectives, numerals, and other nouns, then the

[4]the precision formula will involve division by zero

[5]the f measure will also involve division by zero if recall is zero

Table 6.6: Precision, Recall and F measure Results for CRF and SVM

	CRF			SVM		
Tags	**Recall**	**Precision**	**F**	**Recall**	**Precision**	**F**
ADJ	0.76	0.86	0.81	0.77	0.83	0.80
ADJC	0.55	0.72	0.62	0.60	0.66	0.63
ADJP	0.44	0.68	0.53	0.47	0.61	0.53
ADJPC	0.25	0.71	0.37	0.31	0.44	0.36
ADV	0.65	0.81	0.72	0.65	0.79	0.71
AUX	0.82	0.94	0.88	0.81	0.92	0.86
CONJ	0.83	0.90	0.87	0.83	0.89	0.86
INT	0	-	-	0	-	-
N	0.97	0.95	0.96	0.96	0.96	0.96
NC	0.91	0.85	0.88	0.91	0.85	0.88
NP	0.94	0.89	0.91	0.92	0.89	0.91
NPC	0.83	0.74	0.78	0.81	0.74	0.77
NUMC	0	-	-	0.50	0.76	0.60
NUMCR	0.98	0.99	0.99	0.99	0.99	0.99
NUMOR	0.89	0.91	0.90	0.94	0.91	0.93
NUMP	0.96	0.95	0.96	0.97	0.96	0.96
NUMPC	0.07	0.23	0.11	0.30	0.44	0.35
PREP	0.94	0.97	0.95	0.94	0.96	0.95
PREPC	0	-	-	0	-	-
PRON	0.66	0.70	0.68	0.66	0.69	0.67
PRONC	0.01	0.04	0.02	0.11	0.21	0.14
PRONP	0.82	0.72	0.77	0.83	0.72	0.77
PRONPC	0.01	0.33	0.02	0.05	0.22	0.08
PUNC	1.00	1.00	1.00	1.00	1.00	1.00
UNC	0.08	0.28	0.12	0.08	0.19	0.11
V	0.93	0.92	0.93	0.92	0.91	0.92
VC	0.35	0.63	0.45	0.38	0.60	0.46
VN	0.90	0.88	0.89	0.89	0.87	0.88
VP	0.76	0.84	0.80	0.75	0.80	0.77
VPC	0.56	0.67	0.61	0.59	0.62	0.60
VREL	0.86	0.77	0.81	0.80	0.74	0.77
Average	**0.60**	**0.67**	**0.62**	**0.64**	**0.68**	**0.65**

Table 6.7: Confusion Matrix for CRF in Percentage

	ADJ	N	NP	NC	V	VP	VREL	Others	Row-sum
ADJ	0	7.7	1.09	0.08	0.14	0.36	0.49	2	11.86
ADJP	0.23	0.08	4.88	0	0	0.51	0.3	0.55	6.55
ADV	0.25	0.94	1.36	0.07	0.5	0.58	0.07	0.66	4.43
CONJ	0	0.26	0.38	0.02	0.1	0.05	0	0.41	1.22
N	4.51	0	3.06	1.66	0.69	0.32	0.09	2.61	12.94
NC	0	1.32	0.15	0	0.02	0.01	0.01	0.87	2.38
NP	0.19	1.71	0	0.2	0.08	3.42	1.22	4.44	11.26
NPC	0	0.01	1.23	0.41	0	0.08	0.01	0.46	2.2
PRONP	0.02	0.05	0.83	0	0	0.04	0	0.47	1.41
PREP	0.06	1	0.26	0.05	0.03	0.02	0.04	0.41	1.87
PRON	0.13	0.3	0.19	0	0	0.1	0.03	1.17	1.92
V	0.13	1.82	0.14	0.04	0	1.56	0.44	0.86	4.99
VN	0.01	2.19	0.21	0.23	0.08	0.1	0.1	0.02	2.94
VP	0.04	0.56	6.06	0.07	2.64	0	7.91	2.36	19.64
VPC	0	0.02	0.05	0.35	0	0.73	0.06	0.79	2
VREL	0.05	0.09	1.14	0.02	0.5	3.75	0	0.5	6.05
Others	0.12	0.7	0.5	0.81	0.48	0.32	0.22	2.45	5.6
Col. sum	5.74	18.75	21.53	4.01	5.26	11.95	10.99	21.03	99.26

Table 6.8: Confusion Matrix for SVM in Percentage

	ADJ	N	NC	NP	V	VP	VREL	Others	Row-sum
ADJ	0	6.59	0.05	0.98	0.16	0.33	0.44	2.04	10.59
ADJC	0.03	0.03	0.3	0.01	0	0	0	0.11	0.48
ADJP	0.42	0.07	0.01	4.05	0	0.52	0.29	0.63	5.99
ADV	0.31	0.86	0.06	1.18	0.49	0.51	0.07	0.79	4.27
N	4.89	0	1.76	2.91	0.8	0.32	0.07	2.97	13.72
NP	0.5	1.89	0.2	0	0.1	3.98	1.32	5.04	13.03
CONJ	0	0.19	0.03	0.39	0.1	0.07	0	0.42	1.2
NC	0.01	1.09	0	0.15	0.02	0.03	0.01	0.96	2.27
NPC	0.01	0.02	0.41	1.06	0.01	0.06	0.01	0.7	2.28
PREP	0.03	0.9	0.04	0.2	0.03	0.02	0.04	0.48	1.74
PRON	0.14	0.21	0	0.16	0.01	0.08	0.03	1.23	1.86
PRONP	0.08	0.02	0	0.51	0	0.03	0.01	0.65	1.3
V	0.19	1.38	0.03	0.12	0	1.95	0.56	0.94	5.17
VN	0.02	1.84	0.24	0.22	0.09	0.37	0.13	0.06	2.97
VP	0.1	0.47	0.08	5.87	2.69	0	7.84	2.64	19.69
VPC	0	0.01	0.31	0.07	0.01	0.65	0.04	0.72	1.81
VREL	0.13	0.09	0.03	0.86	0.53	5.47	0	0.65	7.76
Others	0.13	0.45	0.31	0.34	0.43	0.34	0.23	2.4	4.63
Col. sum	6.99	16.11	3.86	19.08	5.47	14.73	11.09	23.43	100.76

stated affixes appear on the prenominal constituents. This phenomenon blurs the morphological distinctions that would otherwise have been useful for distinguishing nouns against their constituents. That is why, for example, **ADJ**s are mistaken for **N**s more than the other way round in both CRF and SVM.

The largest error percentage resulted from confusion between **VP**s (verb with preposition) and **VREL** (verb relative). In CRF, mistaking **VPs** for **VREL** accounts for 7.91% of the total errors. The corresponding value for SVM is 7.84%. Closer examination of the results shows that the POS taggers did not actually predict wrong tags at least in some cases. The problem is that the predictions were made against wrongly assigned tags in the test set. In fact, some of the confusions between some pairs can be shown to be errors in the test set. For example, *yehonu* has been tagged as both **VP** and **VREL** in the same test under similar conditions, making either prediction wrong for the other.

Chapter 7

Conclusions

- Introduction
- Summary of Results
- Limitations
- Recommendations

7.1 Introduction

In this dissertation, theoretical and practical POS tagging issues have been discussed with the view to improving part of speech tagging performance for Amharic, which was never above 90%. Knowledge of Amharic morphology, the given annotated data and the tagging algorithms have been examined and shown to play critical roles in the final performance result. With the experiments carried out on WIC corpus($\approx 207k$), POS tagging accuracies for Amharic have crossed above the 90% limit for the first time.

The improvement in performance is attributed to a combination of three factors. First, the POS tagged corpus (WIC) has been cleaned up to minimize the pre-existing tagging errors and inconsistencies. Second, the vowel patterns and the roots, which are characteristics of Semitic languages, have been used to serve as important elements of the feature set. Third, state-of-the-art of machine learning algorithms have been used and parameter tuning has been done whenever necessary and as much as possible.

This work is not different from previous work on the first factor as much as it is with the other two factors. The tagging errors and inconsistencies in the WIC corpus have been acknowledged by all previous researchers and their experiments were done on their "cleaned" version of it. In this sense, this work can only be different from them in the kind and degree of the clean-up. It is, however, definitely different on the other factors.

Even though most of the features used are not different to those used in most machine learning-based tagging methods, two other unique features have been included - the vowel patterns and the radicals. The vowel patterns are the letters without the consonants and the radicals are the consonants without the vowels. These additional features have reduced to some degree the impact of the data sparsity problem (the problem of not observing enough data), thereby contributing to the performance improvement.

The last factor that contributed to performance improvement is the choice of the values for the parameters in the selected algorithms. Previous experiments that applied similar tools used default parameter values. The experiments carried out in this dissertation, however, used selected parameters, different from the default values, and have been observed to have significant impact on the final performance results. Of particular importance is the penalty parameter C that controls the underfitting/overfitting phenomena of machine learning algorithms. Specifically, smaller C values have been chosen in order to minimize the effect of training errors in the learned models.

This work is also different from previous Amharic POS tagging experiments in the methods applied. Brill tagging has been applied for the first time and has resulted in encouraging results. The current patch templates in Brill are based on the words and tags surrounding the focus word. For Semitic languages, more benefit can be obtained by extending the templates to look at the affixes of the neighboring words.

7.2 Summary of Results

The highest POS tagging accuracies have been achieved by both conditional random fields and support vector machines, followed by Brill and TnT. The CRF tagger achieved an average accuracy of 90.95% on a 10-fold cross-validation while under the same conditions, SVM achieved an average of 90.43%. The difference is statistically significant ($p = 0.00052 < 0.05$). Brill and TnT achieved compa-

Table 7.1: Best POS Tagging Accuracies for 3 Semitic Languages

Language	Tag	Training	Test	Method	Result (in %)
Arabic	24	132k	12k	SVM	97.6 (Habash and Rambow, 2005)
Arabic	24	4119 s	4k s	SVM	95.5 (Diab et al., 2004)
Hebrew	21	39k	9k	HMM	89.59 (Bar-Haim et al., 2005)
Amharic	30	180k	20k	SVM	88.30 (Gambäck et al., 2009)
Amharic	31	186k	21k	CRF	90.95
Amharic	31	186k	21k	CRF	90.43
Amharic	31	186k	21k	Brill	87.41
Amharic	31	186k	21k	TnT	87.09

rable accuracies. Brill achieved an average overall accuracy of 87.41%, which is statistically higher ($p = 0.00355 < 0.05$) than 87.09% for TnT.

Table 7.1 shows the aforementioned results in the context of results for two related languages, namely Arabic and Hebrew. As can be seen from the table, Amharic is not badly positioned with respect to the size of the tagged corpus. In fact, it is the biggest but its quality is probably the worst. Even so, the accuracy obtained for a tagset of 31 is encouraging. The results reported in the table for Arabic and Hebrew are for a tag size of 24, which should favor higher accuracy results.

Error analysis using confusion matrices for both CRF and SVM shows that between 39% and 45% of the errors resulted from confusions of non-noun families for noun-families. Closer examination of their morphology shows that both families share the same affixes. A noun phrase that consists of only the head noun gets affixes of prepositions, conjunctions, etc. However, if the noun phrase contains prenominal constituents such as adjectives, the affixes appear on the constituents, thereby blurring the morphological distinctions necessary to distinguish them.

7.3 Limitations

As explained above, POS tagging performances for Amharic have been improved because of contributions from three factors. The relative contributions of each factor has, however, not been shown. Of particular importance for research is the second factor which is related to the vowel patterns and radicals in Semitic languages. These features have been included in the featuresets learned by the tagging models. It would have more interesting if performance results were also available for featuresets that excludes them on known and unknown words. Results with and without the vowel patterns/radicals would have clearly shown their

significance.

Another limitation of this dissertation is related to the question: how much more annotated data contributes how much more performance. In other words, the relationship between corpus size and performance has not been demonstrated. Interestingly, this could be related to the vowel patterns and radicals. In what data range sizes is the effect of the vowel patterns or radicals more visible? Such and other data size related issues could have been further explored.

The last limitation is inadequacy of the POS tagging accuracies. Even though, the results obtained in our experiments are higher than previous results. It is still far behind Arabic and English, where accuracies are above 97%[1].

7.4 Recommendations

POS tagging experiments for Amharic have been so far based on supervised stochastic methods using annotated data and have not used a morphological analyzer primarily because there was not any until recently. Now since a morphological analyzer is available (Gasser, 2009*a*), it will be interesting to see how the morphological analyzer can be integrated in such tagging methods. It will also be interesting to explore a rule-based tagger based on the same or a similar morphological analyzer.

Vowel patterns and consonants in Semitic words have special syntax and semantics values. Their significance in improving the quality and performance of basic tasks and applications such as POS tagging, parsing and machine translation should be explored for Semitic languages and maybe for other languages too.

In addition to exploring other techniques in POS tagging for Amharic. It is also important to develop the linguistic resources. In fact, there will not be major advances in Amharic POS tagging using stochastic methods unless the existing corpus is cleaned further or a new one developed.

In development of new linguistic resources, it is necessary to follow standards with two main objectives: first, minimization of errors and inconsistencies similar to the ones observed in WIC; second, maximization of flexibility and adaptability of the resources for applications and other processing tasks including parsing, which should be the next research topic after significant improvements in POS tagging.

[1]See table 2.1 in chapter 2 for the state-of-the-art results in POS tagging for English

Bibliography

Adafre, S. F. (2005), Part of speech tagging for amharic using conditional random fields, *in* 'Semitic '05: Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages', Association for Computational Linguistics, Morristown, NJ, USA, pp. 47–54.

Adler, M. (2001), Hidden markov model for hebrew part-of-speech tagging (in hebrew), Master's thesis, Ben Gurion University, Israel.

Aires, R., Aluísio, S., Kuhn, D., Marcio, L. and Oliveira Jr, O. (2000), 'Combining multiple classifiers to improve part of speech tagging: A case study for brazilian portuguese', *Núcleo* **3**, 1.

Aklilu, A. (1987), *Amharic-English Dictionary*, Kuraz Publishing Agency.

Argaw, A. A. and Asker, L. (2005), Web mining for an amharic - english bilingual corpus, *in* 'WEBIST', pp. 239–246.

Bahl, L. R. and Mercer, R. L. (1976), Part of speech assignment by a statistical decision algorithm, *in* 'In Proceedings' IEEE International Symposium on Information Theory', pp. 88–89.

Bar-Haim, R., Sima'an, K. and Winter, Y. (2005), Choosing an optimal architecture for segmentation and pos-tagging of modern hebrew, *in* 'Semitic '05: Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages', Association for Computational Linguistics, Morristown, NJ, USA, pp. 39–46.

Bar-haim, R., Sima'an, K. and Winter, Y. (2008), 'Part-of-speech tagging of modern hebrew text', *Nat. Lang. Eng.* **14**(2), 223–251.

Baum, L. (1972), 'An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes', *Inequalities* **3**(1), 1–8.

Beesley, K. (1998), Arabic morphology using only finite-state operations, *in* 'Proceedings of the Workshop on Computational Approaches to Semitic languages', Association for Computational Linguistics, pp. 50–57.

Birocheau (2003), Etiquetage morphologique et contribution la dsambigusation automatique des ambiguts morphologiques sur un lexique anglais, PhD thesis, Thse de doctorat sous la dir. de Sylviane Cardey, Centre Tesnire, Universit de Franche-Comt, Besanon.

Boser, B., Guyon, I. and Vapnik, V. (1992), A training algorithm for optimal margin classifiers, *in* 'Proceedings of the fifth annual workshop on Computational learning theory', ACM, pp. 144–152.

Brants, T. (2000), 'Tnt - a statistical part-of-speech tagger'.

Brill, E. (1992), 'A simple rule-based part of speech tagger'.

Brill, E. (1995), 'Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging', *Computational Linguistics* **21**, 543–565.

Brill, E. and Wu, J. (1998), Classifier combination for improved lexical disambiguation, *in* 'ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS', Vol. 36, Association for Computational Linguistics, pp. 191–195.

Buckwalter, T. (2002), *Buckwalter Arabic Morphological Analyzer Version 1.0*, Linguistic Data Consortium, Philadelphia.
URL: *http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2002L49*

Cardey, S. and Greenfield, P. (2003), Disambiguating and tagging using systemic grammar, *in* 'Proceedings of the 8th International Symposium on Social Communication', pp. 559–564.

Carmel, D. and Maarek, Y. S. (1999), Morphological disambiguation for hebrew search systems, *in* 'In Proceeding of NGITS-99', Springer, pp. 312–326.

Chang, C.-C. and Lin, C.-J. (2001), *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.

Chen, S. and Goodman, J. (1999), 'An empirical study of smoothing techniques for language modeling', *Computer Speech and Language* **13**(4), 359–394.

Church, K. and Gale, W. (1991), 'A comparison of the enhanced good-turing and deleted estimation methods for estimating probabilities of english bigrams', *Computer Speech & Language* **5**(1), 19–54.

Church, K. W. (1988), A stochastic parts program and noun phrase parser for unrestricted text, *in* 'In Proceedings of the Second Conference on Applied Natural Language Processing', pp. 136–143.

CIA (2010), 'The world fact book - ethiopia', [Accessed in May 2010].
URL: *https://www.cia.gov/library/publications/the-world-factbook/geos/et.html*

Cortes, C. and Vapnik, V. (1995), 'Support-vector networks', *Machine learning* **20**(3), 273–297.

Daelemans, W., Van Den Bosch, A., Zavrel, J., Veenstra, J., Buchholz, S. and Busser, B. (1998), 'Rapid development of nlp modules with memory-based learning', *Proceedings of ELSNET in Wonderland* pp. 105–113.

Daelemans, W., Zavrel, J., Berck, P. and Gillis, S. (1996), Mbt: A memory-based part of speech tagger generator, *in* 'Proceedings of the Fourth Workshop on Very Large Corpora', pp. 14–27.

De Pauw, G., de Schryver, G. and Wagacha, P. (2006), Data-driven part-of-speech tagging of kiswahili, *in* 'Text, Speech and Dialogue', Springer, pp. 197–204.

Demeke, G. and Getachew, M. (2006), 'Manual annotation of amharic news items with part-of-speech tags and its challenges', *Ethiopian Languages Research Center Working Papers* **2**, 1–16.

Derose, S. J. (1988), 'Grammatical category disambiguation by statistical optimization', *Computational Linguistics* **14**, 31–39.

Diab, M., Hacioglu, K. and Jurafsky, D. (2004), Automatic tagging of arabic text: from raw text to base phrase chunks, *in* 'In 5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL04', pp. 149–152.

Dietterich, T. (1998), 'Approximate statistical tests for comparing supervised classification learning algorithms', *Neural computation* **10**(7), 1895–1923.

Firdyiwek, Y. and Yaqob, D. (1997), 'The system for ethiopic representation in ascii', *URL: citeseer. ist. psu. edu/56365. html* .

Francis, W. (1980), 'A tagged corpus–problems and prospects', *Studies in English linguistics for Randolph Quirk* pp. 192–209.

Gambäck, B., Olsson, F., Argaw, A. A. and Asker, L. (2009), Methods for amharic part-of-speech tagging, *in* 'AfLaT '09: Proceedings of the First Workshop on

Language Technologies for African Languages', Association for Computational Linguistics, Morristown, NJ, USA, pp. 104–111.

Garside, R. (1987), 'The claws word-tagging system', *The computational analysis of English: a corpus-based approach* pp. 30–41.

Garside, R. and Smith, N. (1997), 'A hybrid grammatical tagger: Claws4', *Corpus annotation: Linguistic information from computer text corpora* pp. 102–121.

Gasser, M. (2009*a*), *HornMorpho 1.0: morphological analysis and generation of Amharic verbs and nouns and Tigrinya verbs.*

Gasser, M. (2009*b*), Semitic morphological analysis and generation using finite state transducers with feature structures, *in* 'EACL '09: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics', Association for Computational Linguistics, Morristown, NJ, USA, pp. 309–317.

Getachew, M. (2001), Automatic part of speech tagging for amharic: An experiment using stochastic hidden markov (hmm) approach, Master's thesis, Addis Ababa University.

Giménez, J. and Marquez, L. (2004), Svmtool: A general pos tagger generator based on support vector machines, *in* 'Proceedings of the 4th International Conference on Language Resources and Evaluation', Citeseer, pp. 43–46.

Good, I. (1953), 'The population frequencies of species and the estimation of population parameters', *Biometrika* **40**(3-4), 237.

Greene, B. and Rubin, G. (1971), 'Automatic grammatical tagging of english', *Providence, RI: Department of Linguistics, Brown University* .

Habash, N. (2005), 'Arabic morphological representations for machine translation', *Arabic Computational Morphology* pp. 263–285.

Habash, N. and Rambow, O. (2005), Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop, *in* 'Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics', Association for Computational Linguistics, p. 580.

Hajič, J. (2000), Morphological tagging: Data vs. dictionaries, *in* 'Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference', Morgan Kaufmann Publishers Inc., pp. 94–101.

Halteren, H., Zavrel, J. and Daelemans, W. (2001), 'Improving accuracy in word class tagging through the combination of machine learning systems', *Computational linguistics* **27**(2), 199–229.

Johnson, W. (1932), 'I.–probability: The deductive and inductive problems', *Mind* **41**(164), 409.

Jurafsky, D., Martin, J. and Kehler, A. (2000), *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*, MIT Press.

Katz, S. (1987), 'Estimation of probabilities from sparse data for the language model component of a speech recognizer', *IEEE Transactions on Acoustics, Speech and Signal Processing* **35**(3), 400–401.

Khoja, S. (2001), Apt: Arabic part-of-speech tagger, *in* 'Proceedings of the Student Workshop at NAACL-2001', Citeseer.

Kudo, T. (2007), 'Crf++: Yet another crf toolkit'. Software available at http://crfpp. sourceforge.net.

Lafferty, J. (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data, Morgan Kaufmann, pp. 282–289.

Leech, G., Garside, R. and Bryant, M. (1994), 'Claws4: The tagging of the british national corpus'.

Levinger, M. (1992), Morphological disambiguation (in hebrew), Master's thesis, Computer Science Department, Technion, Haifa, Israel.

Levinger, M., Ornan, U. and Itai, A. (1995), 'Morphological disambiguation in hebrew using a priori probabilities', *Computational Linguistics* **21**, 383–404.

Lidstone, G. (1920), 'Note on the general case of the bayes-laplace formula for inductive or a posteriori probabilities', *Transactions of the Faculty of Actuaries* **8**(182-192), 80.

Loftsson, H. (2006), 'Tagging icelandic text: An experiment with integrations and combinations of taggers', *Language Resources and Evaluation* **40**(2), 175–181.

Loper, E. and Bird, S. (2002), 'Nltk: The natural language toolkit', *CoRR* **cs.CL/0205028**.

MA, A., Braverman, E. and LI, R. (1964), 'Theoretical foundations of the potential function method in pattern recognition learning', *Automation and remote control* **25**, 821–837.

Mansour, S. (2008), Combining character and morpheme based models for part-of-speech tagging of semitic languages., Master's thesis, Technion, Haifa, Israel.

Marcus, M. P., Marcinkiewicz, M. A. and Santorini, B. (1993), 'Building a large annotated corpus of english: the penn treebank', *Comput. Linguist.* **19**(2), 313–330.

Marshall, I. (1983), 'Choice of grammatical word-class without global syntactic analysis: tagging words in the lob corpus', *Computers and the Humanities* **17**(3), 139–150.

McCallum, A. (2002), 'Mallet: A machine learning for language toolkit'.

Morgadinho (2004), Analyse pour un systme d'tiquetage morphologique et de dsambigusation morphosyntaxique:Labelgram espanol, PhD thesis, Thse de doctorat sous la dir. de Sylviane Cardey, Centre Tesnire, Universit de Franche-Comt, Besanon.

Palmer, M. S. and Finin, T. (1990), 'Workshop on the evaluation of natural language processing systems', *Computational Linguistics* **16**, 175–181.

Rabiner, L. R. (1989), A tutorial on hidden markov models and selected applications in speech recognition, *in* 'Proceedings of the IEEE', pp. 257–286.

Ratnaparkhi, A. (1996), 'A maximum entropy model for part-of-speech tagging'.

Segal, E. (2000), Hebrew morphological analyzer for hebrew undotted texts., Master's thesis, Computer Science Department, Technion, Haifa, Israel.

Shacham, D. (2007), Morphological disambiguation of Hebrew, PhD thesis, Citeseer.

Shen, L., Satta, G. and Joshi, A. (2007), Guided learning for bidirectional sequence classification, *in* 'ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS', Vol. 45, p. 760.

Simaan, K., Itai, A., Winter, Y., Altman, A. and Nativ, N. (2001), 'Building a tree-bank of modern hebrew text', *Traitement Automatique des Langues* **42**(2).

Sjöbergh, J. (2003), Combining pos-taggers for improved accuracy on swedish text, *in* 'Proceedings of NoDaLiDa 2003', Citeseer.

Spoustová, D., Hajič, J., Votrubec, J., Krbec, P. and Květoň, P. (2007), The best of two worlds: Cooperation of statistical and rule-based taggers for czech, *in* 'Proceedings of the Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies', Association for Computational Linguistics, pp. 67–74.

Stolcke, A. (2002), Srilm-an extensible language modeling toolkit, *in* 'Seventh International Conference on Spoken Language Processing', Vol. 3, Citeseer, pp. 901–904.

Stolz, W. S., Tannenbaum, P. H. and Carstensen, T. V. (1965), 'A stochastic approach to the grammatical coding of english', *Communications of the ACM* pp. 399–405.

Tachbelie, M. and Menzel, W. (2009), 'Amharic part-of-speech tagger for factored language modeling'.

Toutanova, K., Klein, D., Manning, C. and Singer, Y. (2003), 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-Volume 1', Association for Computational Linguistics, pp. 173–180.

Tsuruoka, Y. (2005), Bidirectional inference with the easiest-first strategy for tagging sequence data, *in* 'Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing', Association for Computational Linguistics Morristown, NJ, USA, pp. 467–474.

Tsuruoka, Y., Tateishi, Y., Kim, J., Ohta, T., McNaught, J., Ananiadou, S. and Tsujii, J. (2005), 'Developing a robust part-of-speech tagger for biomedical text', *Advances in Informatics* pp. 382–392.

Viterbi, A. (1967), 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm', *IEEE transactions on Information Theory* **13**(2), 260–269.

Voutilainen, A. (1995), A syntax-based part-of-speech analyser, *in* 'Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics', Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 157–164.

Wikipedia (2010), 'Amharic', [Accessed in June].
URL: *http://en.wikipedia.org/wiki/Amharic*

Ymam, B. (2007), *Amharic Grammar (In Amharic)*, ISBN 978-99944-999-8-4, second edition edn, Eleni Press, Addis Ababa.

Zavrel, J. and Daelemans, W. (1999), Recent advances in memory-based part-of-speech tagging, *in* 'VI Simposio Internacional de Comunicacion Social', Citeseer, pp. 590–597.