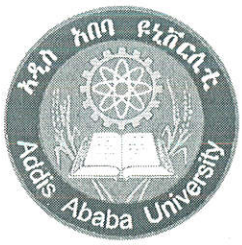


ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

THE APPLICATION OF DECISION TREE FOR PART OF SPEECH (POS) TAGGING FOR AMHARIC

A THESIS SUBMITTED TO SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY

GEBEYEHU KEBEDE

September, 2009
A.A.U

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

THE APPLICATION OF DECISION TREE FOR PART OF SPEECH (POS) TAGGING FOR AMHARIC

BY

GEBEYEHU KEBEDE

Approved by the Examining Board

Lemma Lessa — 20/11/2009

Chairman, Examining Committee — Signature

ERMIAS ABEBE — Signature

Advisor

Million M. — Signature

Examiner

ACKNOWLEDGMENT

Above all I would like to praise the Almighty God for being in my side in all my endeavors and gave me endurance at times of many challenges and rainy days.

Next I would like to express my heartfelt gratitude and appreciation to my advisor Ato Ermias Abebe for his invaluable guidance, advice and outstanding comments right from the inception of the research proposal up to the last point of this thesis and for providing me his precious time whenever I was in need of his help. Had it not been for his genuine help, this research work would have not been accomplished.

My heartfelt gratitude and special thanks goes to my wife Fitsumwork Haile who gave me all rounded support in this journey. Indeed she deserves special thanks beyond what words can express.

I am enormously indebted to my friends Biru Asmare, Solomon Mekonnen, Dawit Mulugeta, Nebeyou Azanaw, Yohannes Mulugeta, Worku Kelemwork, Wakgari Dibaba, and Abdu Seid who gave me their support and encouragement and contributed a lot at different stages of writing the thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

ACRONYM

BNC	British National Corpus
CART	Classification and Regression Tree
CF	Confidence Factor
CHAID	Chi-Squared Automatic Interaction Detection
CV	Cross Validation
E. C	Ethiopian Calendar
ELRC	Ethiopian Languages Research Center
ID3	Induction Decision Tree version three
MDL	Minimum Description Length
NLP	Natural Language Processing
POS	Part of Speech
TDIDT	Top Down Induction of Decision Trees

Abstract

Automatic understanding of natural languages requires a set of language processing tools. POS tagger, which assigns the proper parts of speech (like noun, verb, adjective, etc) to words in a sentence, is one of these tools. This study investigates the possibility of applying decision tree based POS tagger for Amharic. The tagger was developed using j48 decision tree classifier algorithm, which is Weka's implementation of C4.5 algorithm.

In the process, a corpus developed by ELRC annotation team was used to get the required data for training and testing the models. The dataset is comprised of 1065 news documents; 210,000 words. A sample of some 800 sentences are selected and used for model development and evaluation. The dataset was preprocessed in line with the requirements of the Weka's data mining tool. In order to support decision tree classification models, a table that contains the contextual and orthographic information is constructed semi-automatically and used as training and testing dataset.

The right and left neighboring words tags for each word are used as contextual information. Moreover, orthographic information about the word like the first and last character, the prefix and suffix, existence of numeric digit within the word and so on are included in the table to provide useful information to the word to be tagged.

Performance tests were conducted at various stages using 10-fold cross validation test option. Experimental results show that, only two successive left and right words tag provide useful contextual information; contextual information beyond two doesn't provide useful information rather noise. In the end, an over all, including ambiguous and unknown words, 84.9% correctness (or accuracy) was obtained using 10-fold cross validation test option. Even though, the accuracy of this study is encouraging further study to improve the accuracy so as to reach at implementation level is recommended.

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

Natural language is a specialized language developed in a usual way as a method of communication between people [8]. It enables human beings to learn and share knowledge in spoken or written form. Natural language in written form serves as a long term record of knowledge transfer from one generation to the next.

The rapid increase of the storage and processing capability of computers and the availability of large electronic data enabled human beings to process natural languages using one or more algorithms. It is concerned with the designing and building of software so that computers analyze, understand, and generate natural languages. But this is not an easy task; understanding language means, among other things, knowing what concepts a word or phrase stands for and knowing how to link those concepts together in a meaningful way. The challenges come from the highly ambiguous nature of natural language [27]. For example, the presence of ambiguous words (like the English words train, race, etc), which have more than one tag, makes the designing of POS tagger a challenging task.

POS tagging, which is one of the natural language processing applications, is the process of assigning a part of speech like noun, verb, pronoun, preposition, adverb, adjective, etc class markers to each word in a sentence [35]. The input to a tagging algorithm is a string

of words of a natural language sentence and a specified tagset and the output is a single unambiguous POS as much as possible or more than one tag that needs further investigation to disambiguate it.

Generally POS taggers can be characterized as rule based or corpus based [14]. Rule based POS tagging models use a set of hand crafted rules and a dictionary to tag words in a sentence [10]. Most Rule based models use a two phase architecture [20]. In the first phase all words in the sentence received all the possible tags using dictionary. Following this, the second phase uses rule templates to get an unambiguous single tag by removing tags that violet the rule template.

Rule based POS taggers are more accurate and better describe linguistic phenomena [22] as they are written from a linguistic point of view and explicitly describe linguistic phenomena. Moreover, rule based POS tagger models may contain many and complex kinds of information; this allows the construction of accurate system.

On the contrary, developing the necessary rules for POS tagging for a particular language is not easy especially for morphologically rich languages like Amharic. Rule based POS tagging needs from few hundreds to thousands of rules by linguistic experts, and it may require years of labor to develop [24]. Transportability, robustness and coverage are also the major limitations of rule based taggers [22]. Transporting the model to other languages means starting the whole process again to model the language. Rule based models do not consider frequency of information and thus have a limited robustness and coverage [ibid].

On the other hand, corpus based taggers use one or more algorithms that learn and build models from the training dataset and apply their experience to new instances. For example, Hidden Markov, Neural network, memory based, and decision tree are some of the existing corpus based tagging algorithms for POS tagging [42].

Decision tree classifier learns the classification rule from the training dataset and applies it to classify the new instance with little human involvement (for training and testing set preparation if it is supervised learning). This minimizes the cost needed to develop classification rules unlike that of rule based POS tagging. Moreover, the rules developed by decision tree classifiers are human understandable and subject to further improvement unlike that of other statistical taggers. That is, it can be verified whether or not they capture true underlying linguistic phenomena [14]. Saying it in different words, the rules generated by decision tree classifiers can be checked and evaluated linguistically to improve the accuracy of the classifier.

1.2 STATEMENT OF THE PROBLEM AND ITS JUSTIFICATION

Amharic is the working language of the federal government of Ethiopia [12]. It is also the working language of many other regional states of the country (Amhara, Addis Ababa, southern nation's, nationalities and peoples, Gambella, Dire Dawa and Benshanguel Gumuz regions). Manuscripts in Amharic are known from the 14th century and the language has been used as a general medium for literature, journalism, education, and so on [3]. A wide variety of Amharic literatures including books, religious writings, fiction, poetry, plays, and magazines are available both in printed and machine readable format.

The increased availability of electronic Amharic publications demands the powerful processing capability of computers for processing, storage and retrieval purposes. Although small in number, experiments on Amharic text processing are conducted on different areas. Some of the experiments done in this area are: Automatic morphological analyzer for Amharic (Tesfaye, 2002), Automatic sentence parsing for Amharic text (Abiyot, 2000; Atelach, 2002), information retrieval (Zelalem, 2001; Saba, 2001; Bethlehem, 2002; Yalemsew 2005), stemming (Nega, 1999), and automatic part of speech tagger for Amharic (Mesfin, 2001: Sisay, 2005; yenewondim, 2006).

Most NLP applications like Morphological analyzer, parser, stemmer, spelling checker, machine translation and information retrieval systems are highly dependant on appropriate tags of words in a sentence as one component of the whole process; i.e. POS tagging is not an end by itself; rather it is one component of other higher level NLP applications. As a consequence, the presence of robust POS tagger for Amharic eases the implementation of the above NLP applications for Amharic.

Mesfin [26] and Yenewondim [42] developed a prototype for POS tagging for Amharic using Hidden Markov model and neural network model respectively. Mesfin [26] used a one page text comprised of 390 words for his experiment. Testing with 29 words on 25-tagsets, Mesfin [ibid] reported 90% accuracy. Similarly, Yenewondim [42] reported an accuracy of 93.88% using 2429 words for training and 392 words for testing on 24-tagsets (using bigram model).

The hidden Markov model uses the lexical and transition probability to handle ambiguous and unknown words. On the other hand, neural network model uses lexical probabilities

as weights to the input layer and it uses a back propagation to readjust weights in the input layer to get a better single output (the tag of the target word). But both Hidden Markov model and neural network model treat language as a black box filled with probabilities and transition weights [14]. They use probability weights (the n-previous tag sequences) to assign the correct POS tag both for ambiguous and unknown words. Their highly dependency on statistical values made them far from linguist experts to evaluate and correct errors.

Unlike other statistical POS taggers, the contexts used in decision tree POS taggers to the problem of POS tagging are not only n-grams; other complex orthographic and morphological information can also be included to provide additional information to the word to be tagged.

Generally, developing decision tree POS tagger has the following merits as compared to other statistical taggers:

- Context information are not only n-grams (the n-previous tag sequences) like other statistical models; rather the n preceding and following tag sequences and other information (e.g. *Tag_1*[1]= <NN>, *Tag_1* ≠ <NN>, *Tag_2* ≠ <ADJ>, etc) may be also included to better model the language.

- Decision trees are flexible[2] [22] to include more morphological information like the suffix, prefix, first and last characters of the target word, etc. at any time before

[1] The description for *Tag_1*, *Tag_2*, etc is found in section 4.4
[2] A combination of different kinds of information in a rich language modeling is performed with in a flexible tagger to further improve the previously reported results [22].

and/or after the model is built in order to improve the accuracy of the model which is impossible in other statistical POS taggers.

- The rules generated from training dataset are subject for further improvement with linguistic analysis.

By seeing the potential advantages of decision tree for POS tagging and the recommendation by Yenewondim [42], this study explores the application of decision tree for Amharic POS tagging.

1.3 OBJECTIVES OF THE STUDY

1.3.1 General Objective

The general objective of this study is to investigate the applicability of decision tree for designing an automatic part of speech tagger for Amharic.

1.3.2 Specific Objectives

To achieve the general objective, the following specific objectives are addressed.

- To make a review of the literature so as to have conceptual understanding and share experiences of others in the area
- To study the morphological properties of Amharic words to identify the inflectional and derivational rules of word formation that are useful to develop training and testing datasets for designing a decision tree POS tagger.
- To examine the type of lexicon required for decision tree POS tagger, and design the lexicon accordingly

- To assess and select one algorithm from the existing different decision tree classification algorithms (ID3, C4.5, etc) based on accuracy and their capability to handle binary and categorical features
- To test/evaluate the performance of the selected POS tagger algorithm for Amharic
- To draw conclusions and recommendations based on experimental results

1.4 *METHODOLOGY OF THE STUDY*

1.4.1 Literature Review

Books, conference papers, research reports, journal articles and other published and unpublished materials (including those from Internet) are consulted for this study. Related works on Amharic word classes are also reviewed to understand the morphological property of Amharic word classes. Related works on POS tagging in general and POS tagging for Amharic in particular are also reviewed to understand the different approaches for POS tagging; to select suitable tagging algorithms; and to develop training and testing dataset for POS tagging for this study.

1.4.2 Corpus preparation for model building and model testing

The tagset used for this work is developed by ELRC. The reason why this corpus is selected is that: it's availability, accessibility, and it is annotated by linguistic experts (i.e. it is more reliable). The tagset used by ELRC annotation team is attached in appendix B. A sample of 19, 634 words, 10-fold cross validation is used to partition the data for training and testing, are taken and manually preprocessed to correct transcription and tagging inconsistencies. Then, the right and/or the left context of each word in the sample

corpus are extracted and converted in to context table. Moreover, the prefix and suffix information for each word in the sample corpus are included in the table manually to give additional information to the word that is going to be tagged.

1.4.3 Decision tree model building and testing

The Weka machine learning tool (weka is an open source machine learning tool developed by the University of waikato Hamilton, New zealand) is used for this study due to the familiarity of the researcher to it; its transportability, accessibility, and processing capability. Besides, the tool is language independent, i.e. it can be used for any languages, including Amharic, English, French, etc. The Weka software package has different algorithms for different purpose. However, in this study only the J48[3] classification algorithm is used for model building and model testing due to its capability to handle both binary and categorical features. A total of six experiments are done using the selected algorithm using 10-fold[4] cross validation test option. Finally, analysis and conclusions are made based on the percentage of correct tag assignment.

1.5 APPLICATION OF RESULTS

POS tagging is a basic task in most natural language processing systems. Natural language processing systems like sentence parsing, phrase recognition, machine translation, grammar and spelling checker, speech recognition, word sense disambiguation and information retrieval relay on POS tagger. Hence, the output of this research can be used as one component of the above NLP processing activities.

[3] The reason for the selection of the algorithm is given in section 5.2
[4] The reason for the selection of 10-fold cross validation test option is found in section 5.3

Moreover, the output rules from the decision tree POS tagger can be used as a starting point to researchers who plan to develop rule based POS taggers for Amharic.

Although this study is conducted on Amharic text corpus, its results could also be applied for any local languages (if tagged corpora[5], and prefix and suffix list is available) that use the Ethiopic script (Like Gurage, Harari, Tigrie, and Tigrniya).

1.6 SCOPE OF THE STUDY

This study develops a decision tree POS tagger for Amharic. For this purpose 10% sample sentences, containing 19634 words, are selected from the corpus developed by ELRC annotation team excluding XML like tags (e.g. the date the file is created, source, name of the document, etc) which are not useful for this experiment. The reason for not using the whole corpus for this study is that, it needs manual processing (which is time consuming, tedious, and costly) as there are inconsistencies related to transcription and tag usage.

The language is modeled by extracting the context, prefix, suffix, the first and last character, and existence of digit characters features both automatically and manually (prefix and suffix features) for each word in the sample corpus. Other information like the infix and the stem of each word in the sample corpus is not included due to resource constraints (time and money).

[5] Corpora (singular corpus) in this context is labeled or tagged electronic text for experimental purpose

For the development of POS tagger, j48 decision tree classification algorithm which is weka's implementation of C4.5 algorithm is used. First, the extracted features (language modeling) are converted into a csv file format (.csv extension), which is one of Weka's valid file formats, and a total of six experiments are conducted to determine the most useful features to the problem of POS tagging for Amharic. The built model for this purpose is not compared with other statistical POS taggers using the same corpus due to time constraint.

1.7 *ORGANIZATION OF THE THESIS*

This paper is organized into six chapters including the current chapter. The second chapter discusses the existing approaches to POS tagging in general and the application of decision tree for POS tagging problem in detail. The third chapter deals with the morphological processes of Amharic. The Amharic word categories are also discussed in detail in this chapter. Chapter four discusses the process of corpus preparation and the design of algorithms for the experiment. Chapter five presents the experimentation detail and the analysis made based on the findings. Finally, the last chapter deals with the conclusions drawn and the recommendations made based on the findings of the study.

CHAPTER TWO

POS TAGGING

2.1 INTRODUCTION

These days, computers are used to process natural language using one or more algorithms to enable a human computer interaction. It is an emerging field of NLP using a set of theories and a set of technologies to process natural language like that of human language processing.

POS tagging, which is one area of natural language processing, automatically assigns to each word in a sentence a part of speech (word class) from a predefined tagset. But, categorizing words in a sentence in to a particular POS is not an easy task due to two reasons [14]: the existence of ambiguous words and unknown words.

Ambiguous words are those words in the lexicon that have more than one tag category. The existence of such words in a sentence poses difficulty to categorize it in to one of its possible POS tags. For instance, the English word 'train' is ambiguous as it can be a noun or a transitive verb. In addition to the ambiguous words, unknown words, which are words in a sentence that are not found in the lexicon (or training set), pose difficulty during tagging. Unknown words assume all the possible tagsets during tagging. Choosing the most appropriate possible tag for unknown words in a sentence is also a big challenge.

To deal with both to ambiguous and unknown words during POS tagging, both the lexical, as well as the context information- i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph has to be considered [10]. Lexical and context information can be extracted using one or more algorithms (e.g. Vieterbi, C4.5, etc.) from the training corpus which is developed either manually (by linguistic expert) or automatically (using algorithms). Once the model is built from the training corpus, then it is used to assign unambiguous tags for unseen (or new) instances.

2.2 OVERVIEW OF POS TAGGING

POS tagging is the process of assigning each word in a sentence the proper part of speech tag in its particular context [23]. The input to the POS tagger is unannotated sequence of words in a sentence and the output is the sequence of words in a sentence with their appropriate part of speech tag. For instance the sentence *abebe meShef geza*[6] 'Abebe bought a book' is a sequence of unannotated words. But *abebe*/NN *meShef*/NN *geza*/V is the output of the tagger, which are sequences of words in a sentence with their appropriate tag separated by slash. The codes *NN* and *V* are tagsets in the lexicon, where *NN* stands for noun and *V* for verb. A tagset represents a collection of tags (which is symbolic representation of word categories) for all classes of words having distinct grammatical behavior. Eg. Penn Treebank tagset.

Automating the training and tagging process can be supervised or unsupervised [17]. Unsupervised tagging models do not require training corpus. Instead, they use

[6] Amharic characters or words are italicized in order to differentiate them from English characters or words. Moreover the equivalent English meaning of each Amharic word or phrase is shown using single quotes (' ')

sophisticated computational methods to automatically induce tagsets and based on those automatic groupings, to either calculate the probabilistic information needed by stochastic taggers or to induce the context rules needed by rule based systems [ibid].

Unlike that of unsupervised taggers, supervised taggers need pre-tagged (training) corpora to build the tagging model and to test the built model using new instances in order to check the accuracy of the built model before using it. For this purpose, there are lots of manually tagged corpora for other languages. For example, the Penn Treebank and the BNC are the well known English corpora available for researchers in the field of NLP. But researchers in Ethiopian languages like Amharic, Affan Oromo, Tigrigna, and so on suffer a lot due to the absence of sufficient corpora for experimentation. Recently, the ELRC annotation team manually annotated 210,000 words from 1065 Amharic news documents [16]. The corpus is available for researchers on website:http://nlp.amharic.org. It is encouraging for researchers in the field of natural language processing for Amharic.

Corpus preparation can be done either manually or automatically [42]. Manual annotation can be done by linguistic experts while automatic annotation can be performed by applying algorithms such as TreeTagger, Neural network, etc. Both manual and automated annotations have their own merits and demerits. Manual annotation is expensive, monotonous, inconsistent due to the high possibility of different annotators to annotate the same corpus differently, time consuming, difficult to correct errors due to inconsistency in the annotation process, and monotonous. But manual annotation will result in fewer errors as compared to automated annotation [ibid].

On the other hand, automated annotation of corpus is fast, easy, requires less involvement of people and it is easy to detect and correct errors as the annotation process is consistent. On the contrary, it lacks the flexibility that is found in human annotators.

2.3 EXISTING APPROACHES TO AUTOMATED POS TAGGING

There are different automated approaches to POS tagging. A simple rule based POS tagging [7]; Hidden Markov model POS tagging[26]; Neural network POS tagging [42]; Decision tree POS tagging [14;19; 22; 23; 34]; and so on are some of the automated POS taggers available today. The approaches can be broadly classified in to two: rule based and corpus based [14]. The details of each of these categories are discussed bellow.

2.3.1 Rule based POS tagging

Rule based POS taggers use hand crafted rules developed by linguistic experts from language paradigms for handling both ambiguous and unknown words. It is a-two stage architecture [20]. In the first stage, it uses lexicon (or dictionary) to tag each word. At this stage, each word received its dictionary tags without considering context. For example, if any English word X is used as a noun (**NN**) or an adjective (**ADJ**) in the dictionary, then at this stage X assumes two tags:

X /NN/ADJ.

Following this, context rules are applied to handle ambiguous and/or unknown words. The context frame rule (rule template) might say something like [17]:

1. If an ambiguous or unknown English word X is preceded by a determiner (**DET**) and followed by a noun (**NN**), tag it as an adjective (**ADJ**).

   DET - X - NN = X/ADJ

2. An ambiguous or unknown English word X is a noun (**NN**) rather than a verb (**V**) if it follows a determiner (**DET**).

   DET-X- =X/NN and so on.

The rules reduce or eliminate tags that are inconsistent with the context leaving correct tags intact [17; 23]; and should reduce the list of POS tags to a single POS tag per word as much as possible. In other words, using the context information ambiguous words are retagged according to the rule template and unknown words are also tagged following the rule template. For example, the ambiguous word **X** in the above example will be retagged as **X/ADJ** if it satisfy rule 1 or as **X/NN** if it satisfy rule 2. If it doesn't satisfy any of the rules, the tags assigned to it at phase one are kept intact (**X/NN/ADJ**).

In addition to contextual information, some rule based taggers also use morphological information to aid in the disambiguation process [17]. This includes the beginning one or two letters of the word, the last one or more letters of the word, the presence of capital letters (capitalization), punctuation mark etc. For instance, one such rule might be: if an ambiguous or unknown English word ends in **-ous**, label it as an adjective; because this is the most common tag for words endingin **-ous** in English.

2.3.2 Corpus based POS tagging approaches

Nowadays, there is a paradigm shift from rule based to corpus based POS tagging due to the ever expanding availability of large corpora; more powerful computing resources; and

a greater demand for natural language based applications [22; 31]. Moreover, unlike that of rule base POS tagging most corpus based POS tagging systems need limited human involvement and reduce the cost associated with it.

Corpus based taggers use large labeled (tagged) dataset for learning, and applies their experience (knowledge) to classify new (unseen) instances. The learning process from training set is based on the probability that a word occurs with a particular tag. Guilder [17] identified three levels of learning complexity for corpus based taggers.

The first level assumes only word/tag pair frequency and the one that encountered most frequently in the training set is assigned to ambiguous instance(s) of that word. Even though this approach yields a valid tag for a given word, it doesn't consider the neighboring word tags for the selection of the most probable tag for the target word. This leads to the possibility of inadmissible sequences of tags with neighboring words.

An alternative approach to the word frequency approach is to calculate the probability of a given sequence of tags occurring (transition probabilities)[7] to alleviate the problem of inadmissible sequences of tags. Transition probabilities are sometimes referred to as the *n-gram* approach. Transition probability approach refers to the fact that the best tag for a given word is determined by the probability that it occurs with the n previous tags [ibid]. The most common algorithm for implementing an n-gram approach is known as the Viterbi Algorithm [ibid]. It is a search algorithm which avoids the polynomial expansion of a breadth first search by trimming the search tree at each level using the best N

[7] The detail for transition probabilities are found in Jurafsky and Martin [20].

Maximum Likelihood Estimates (where n represents the number of tags of the following word).

The third level of complexity that can be introduced into a stochastic tagger combines the previous two approaches, using both tag sequence probabilities and word frequency measurements. This approach is known as a Hidden Markov Model. Hidden Markov model is based on the following assumptions: Each hidden tag state produces a word in the sentence and each word is:

1. Uncorrelated with all the other words and their tags and
2. Probabilities depend on the n- previous tags only

Corpus based taggers can be further classified as statistical and machine learning family [24]. Statistical taggers build statistical model of the language from the available training set and apply it to classify unseen data (or new instances).

Although statistical models involve either supervised or unsupervised learning, only those taggers that include more sophisticated information than n-gram model are categorized under machine learning tagging systems [ibid]. Transformation based learning; decision tree; artificial neural network and so on are under this category. The detail of decision tree is given in the next section since it is the method that is used in this study.

2.4 OVERVIEW OF TOP DOWN INDUCTION OF DECISION TREES (TDIDT)

Decision tree is a classifier with a flow-chart-like tree structure with three nodes (root node, internal node, and leaf node) and an arc (branch) that connects two nodes [18]. The root node has no incoming edges and zero or more outgoing edges. Internal nodes have exactly one incoming edges and two or more outgoing edges. Finally, the leaf node has one incoming edge and no outgoing edges and represents the class value.

Each non-terminal node of a decision tree is labeled with the identifier of one attribute, each edge (or branch) is labeled with the possible value for the attribute attached to the node the branch comes from, and the leaf node represents sets of examples of the same class. For example, in Figure 2.1, *tag_2, tag_1*, and *tag2* are attributes for branching and the tagset {NN, ADJ, PUNC, V, ...} are the set of possible values for attribute *tag_2, tag_1* and *tag2*. C_i represents the class to which the objects of the corresponding leaf node belong.

To classify a new object using the developed model, start from the root of the tree and descend following the path that satisfies the test criteria until the terminal node is reach. The class attached this terminal node (leaf) is the class assigned to the new object. For instance, a new object that has a value <N> for tag_2 attribute, and <V> for tag2 attribute will be assigned a class C_i.

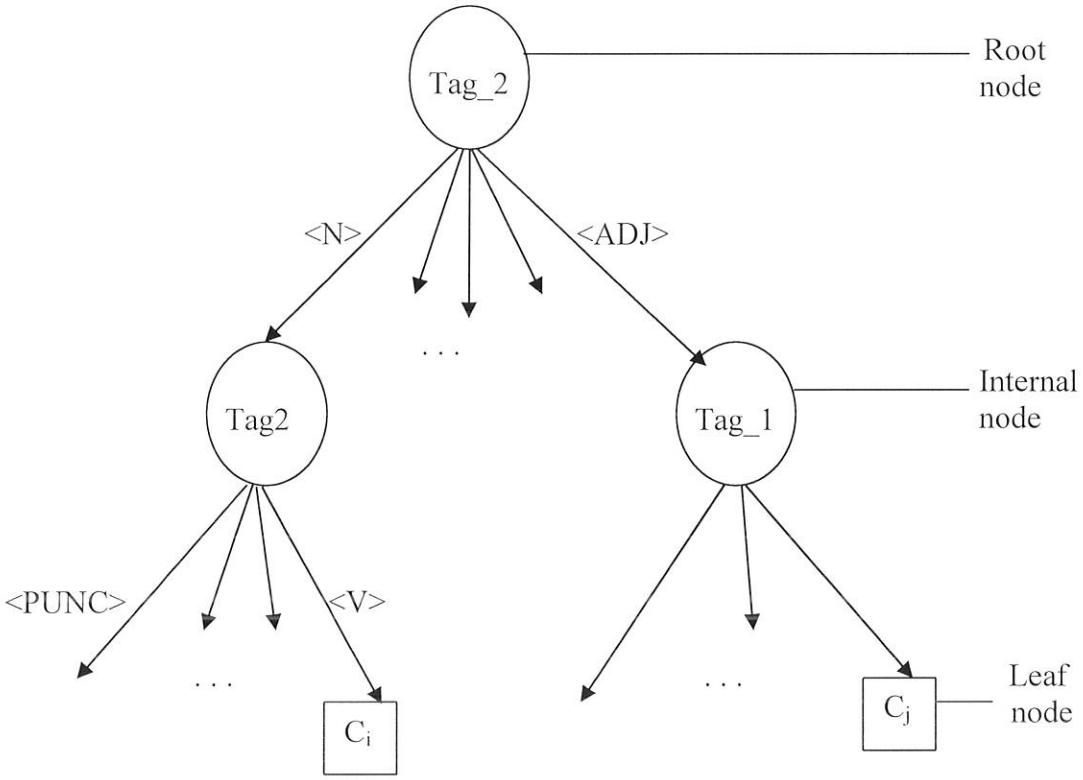


Figure 2.1 A decision tree diagram

Decision tree are used for classification in many areas, especially supervised machine learning [23]. Decision tree is used for Medical Diagnosis: Predicting tumor cells as benign or malignant; Fraud Detection: Classifying credit card transactions as legitimate or fraudulent; Classifying secondary structures of protein as alpha-helix, beta-sheet, or random coil; and Categorizing news stories as finance, weather, entertainment, sports, etc.

The application of decision tree for classification is a two step process: Model building and model using. Figure 2.2 shows the steps to use decision tree for classification. During

model building, the training set is used to construct by induction a classification rule such that it can be determined the class of any other object by applying this rule over the values of its attributes. TDIDT constitutes non incremental supervised learning algorithms from examples (training set) that construct decision trees in a top down way guided by the frequency of information in the example but not on the order of the examples [ibid]. The tree grows through an iterative splitting of data into discrete groups, where the goal is to maximize the "distance" between groups at each split. A number of different algorithms may be used for building decision trees [39] including ID3 (Induction Decision Tree ver. 3), CHAID (Chi-squared Automatic Interaction Detection), CART (Classification and Regression Trees), Quest, and C5.0 (the latest version of C4.5 by Ross Quinlan et al (1986)). Finally, previously unseen or unknown objects are classified by applying the developed model.

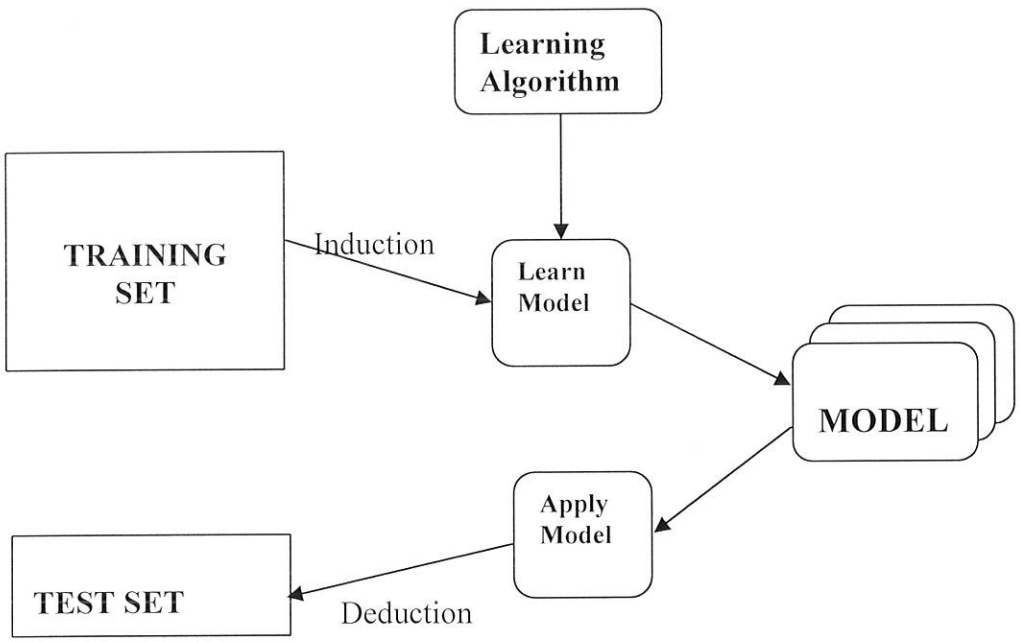


Figure 2.2 Classification model of decision tree

2.4.1 Application of Decision tree for POS tagging

Decision trees are used to assign a morphosyntactic tags to words in a sentence. Márquez and Rodrìguez [23] used decision tree for POS tagging for Spanish language. They annotated 17K word from a book (Spanish novel book) using 18 tagsets. From this dataset, examples are extracted for training and testing for each of the four different ambiguity levels:Article-pronoun ambiguity, the word 'que' meaning that, Noun-verb ambiguity, and unknown words. For each ambiguity level, some examples which belong to the corresponding ambiguity level are extracted and divided in to training and testing set. The training and testing set contains contextual, morphological and orthographic information which is peculiar to the language. Following this, experiment is carried out for each ambiguity level and performance result shows an increase in accuracy- from 82.16% (which is the result of the most likely tagger) to 98% (net increase 15.84 %) when decision tree is used for Article-pronoun ambiguity level. Similarly, the net increase for Noun-Verb ambiguity, 'que' and unknown words are 16.6%, 39.67% and 41.5% respectively. The system shows significant improvement on unknown and the word 'que' than the other ambiguity class levels. This shows that, if the language is modeled with careful linguistic analysis, decision trees can predict unknown and ambiguous words with reduced error rate. After tree pruning is made based on error rate threshold and minimum number of instances, and tuning the default parameters (e.g. the proper contexts to be considered, Eps1-the weight assigned to the attribute selection function, etc.) an overall accuracy of 93 % achieved. Moreover, the system outperforms when only two left and two right contexts are used-which is determined experimentally.

Hirshman [19] also used binary decision tree classifier (taggers) using the j48 algorithm on the Penn Treebank corpus. The training and testing sets for his study is taken from the Penn Treebank corpus of the Wall Street Journal articles which is annotated using 38 tagsets. He carried out different experiments by varying the size of the training set (1000, 5000, 10,000, 25,000, 50,000 and 100,000) and fixed number of test sets (50,000 words), the number of word specific subtrees grown, and the number of previous POS considered when making a tagging decision. In most cases, increasing the number of the training set shows improvement on accuracy. Moreover, bigram contexts (considering previous two contexts) results in better accuracy than other contexts. On the other hand, increasing the number of word specific subtrees doesn't improve the accuracy of the system significantly-in some cases the accuracy even decreases. The system achieved an accuracy of 83.57 % when bigam contexts with 250 word specific subtrees. The main challenge reported here is the requirement of large memory by decision tree POS tagger and it is unable to use large datasets for the experiment. This study also not considered right context which can be useful to disambiguate during tagging. Moreover, the method used for tree pruning to get optimal tree size is not reported in this work.

Decision trees are also used to calculate transition probabilities and shows better performance than the standard trigram taggers (trigram Hidden Markov Model) Schmid [34]. The TreeTagger used by Schmid [ibid] and the standard trigram tagger differ in the way that transition probabilities are estimated. The standard n-gram taggers use the Markov models based on maximum likelihood estimation (MLE) principle [ibid].

On the other hand, the TreeTagger estimates transition probabilities based on information gain principles which is presented in section 2.4.3. At each recursion step, the one that gives better information is attached to the current node of the decision tree and this node is expanded recursively following the same procedure (information gain measure calculation, followed by selection attributes to be attached to the node). Finally, leaf nodes are attached with tags that satisfy the test criteria with their support probabilities.

Schemid [ibid] trained and tested the TreeTagger and the standard trigram tagger using 2 million words and some 100, 000 words respectively and achieved an accuracy of 96.34 which is better than the standard tree tagger. That is, the trigram TreeTagger is better than the standard trigram tree tagger by 0.3%. The most interesting use of TreeTagger is its robustness in contrast to the standard tree tagger [ibid]. That is, small training corpus did not result in a sharp degradation of the accuracy, as it was observed for standard trigram taggers.

The use of decision tree for POS tagging is two fold. The first is that it reduces the cost incurred to develop classification rules as it learns the rule directly from large annotated corpora. The second is, the rules it generated are human understandable and they are subject to further improvement. Apart from its advantages, memory is the main problem reported by some researchers. Decision tree parses all of the instances in main memory to do computation-which needs high memory.

2.4.2 Decision Trees and Attribute Selection

The basic algorithm for decision tree induction is a greedy algorithm (for choosing test attributes) that constructs decision trees in a top-down recursive divide-and-conquer manner [18]. Decision tree algorithms learn from independent instances where instances are represented by attribute-value pairs. It searches through the attributes of the training instances and extracts the attribute that best separates the given examples. For the selection of the attribute with the most inhomogeneous class distribution the algorithm uses the concept of information gain [ibid], which is explained in section 2.4.3. The reason why information gain is selected for the selection of best attributes than other measures (e.g. Gini index, gain ratio, etc) is that C4.5 algorithm, which is used for this study, uses information gain ratio [30] to select best attributes for branching. Such a measure is referred to as an attribute selection measure or a measure of the goodness of split [18]. The attribute with the highest information gain (or greatest entropy reduction) is chosen as the test attribute for the current node. This attribute minimizes the information needed to classify the samples in the resulting partitions and reflects the least randomness or "impurity" in these partitions. Such an information-theoretic approach minimizes the expected number of tests needed to classify an object and guarantees that simple (but not necessarily the simplest) tree is found. Here, the central focus of the decision tree growing algorithm is selecting which attribute to test at each node in the tree.

2.4.3 Information Gain

Information gain measure is used to select the test attribute at each node in the tree [18]. The attribute with the highest information gain or greatest entropy reduction is chosen to

test attribute for the current node. The expected information (or the split information) needed to classify a given sample S, which consists of S data samples residing in *m* distinct classes C_i where $1 \leq i \leq m$ is given by [ibid]:

$$I(S_1, S_2, \ldots, S_m) = -\sum_{i=1}^{m} P_i \log_2(P_i)$$

Where P_i is the probability that an arbitrary sample belongs to class C_i and estimated by S_i/S and S_i is the number of samples of S in class C_i.

The entropy and information gain has to be calculated for each attribute to select the best attribute for splitting. Hence, if an attribute A have N distinct values, $\{a_1, a_2, \ldots, a_n\}$ and A partitions S into N subsets $\{S_1, S_2, \ldots, S_n\}$, where S_j contains those samples in S that have values a_j of A. If A is selected as the test attribute (i.e., the best attribute for splitting), then S_1, S_2, …, S_n would correspond to the branch grown from the node containing the set S. The entropy, or expected information based on the partitioning into subsets by an attribute A, is given by:

$$E(A) = -\sum_{i=1}^{n} \frac{S_{1j} + S_{2j} + \ldots + S_{mj}}{S} I(S_{ij}, S_{2j}, \ldots, S_{mj})$$

Where:S_{ij} to be the number of samples of class C_i in a subset S_j and $(S_{ij} + S_{2j} + \ldots + S_{mj})/S$ acts as the weights of the jth subset and is the ratio of number of samples in the subset to total sample in S.

The smaller the entropy value, the greater will be the purity of the subset partitions [ibid].

It should be noted that, for a given subset S_j.

$$I(S_{1j}, S_{2j}, \ldots, S_{mj}) = -\sum_{i=1}^{m} P_{ij} \log_2(P_{ij})$$

Where $P_{ij} = S_{ij} / S_j$ is the probability that a sample in S_j belongs to class C_i.

As a result, the encoding information that would be gained by branching on attribute A is:

$$\text{Gain}(A) = I(S_1, S_2, \ldots, S_m) - E(A)$$

In other words, Gain (A) is the expected reduction in entropy caused by knowing the value of the attribute A.

The algorithm computes the information gain of each attribute. The attribute with the highest information gain is considered as the most discriminating attribute of the set under consideration. In other words, the smaller the entropy, the purer is the subset partition. So, an attribute that yields maximum information gain will be chosen for dataset partitioning. Then, a node is created and labeled with the chosen attribute; branches are formed for each value of the attribute, and the samples are partitioned accordingly. The same criteria will then be applied to each split sample.

The algorithm stops partitioning if either one or more of the following conditions are satisfied [39]:

1. All samples for a given node belong to the same class
2. There are no remaining attributes for further partitioning

3. If all the samples are exhausted. i.e., there are no samples left

2.4.4 Overfitting and tree pruning

Noise or outliers will result anomalies during decision tree construction. Tree pruning is used to address the problem of overfitting the data based on statistical measures. Overfiting of the training dataset will result if the features have many values [40]. If a feature with many values is selected as test attributes it uniquely identify each instance in the training dataset but it is inadequate to classify new instances (test set). For example, decision tree models can identify each instance from the training dataset based on their id (if id is a primary key feature) with little or no error. But it is unable to classify new instances which are not seen in the training set-as nothing is learned from the training set [ibid]. The corpus for this study has many values for target word, prefix, suffix, first character, and last character features. To avoid this problem, tree pruning is applied on the built model [18; 39]. There are two approaches to tree pruning [ibid]: pre-pruning and post-pruning.

Pre-pruning approach halts tree construction early-do not split a node if this would result in the goodness measure falling below a threshold. It compares the distribution of classes before and after the split and test statistically (Chi-squired test, information gain, and so on) to assess the goodness of split [ibid]. If partitioning the sample at the node will result in a split that falls below the pre-specified threshold, then further partitioning of the given subset is halted.

On the other hand, pos-pruning approach removes branches from a fully grown tree based on error estimation, significance testing, and MDL principle and the lowest unpruned node becomes a leaf and is labeled by the most frequent class among its former branches [18; 39; 40]. It uses either sub tree raising or sub tree replacement pruning operations to get an optimal tree for classification.

After tree pruning, an independent test set is used to estimate the accuracy of each tree and the decision tree that minimizes the expected error rate is preferred to use to classify new instances.

2.4.5 Extracting classification rules from Decision Trees

The knowledge represented in decision tree can be transformed in to IF-THEN rules which are easily understandable by human beings [18; 39]. One rule is created for each path from the root node to the leaf node by ANDing each attribute-value pairs in the rule IF part and assigning the leaf node value for the consequent (THEN part) . For example, from the decision tree shown in Figure 2.1, IF tag-2 = "NN" AND tag+2 = "V" THEN tag = "C_i" can be extracted being the class attribute tag.

2.4.6 Advantages of using Decision Trees for POS tagging

The application of decision tree for POS tagging has, generally, the following advantages:

- Decision trees make few passes through the data (no more than one pass for each level of the tree); as a consequence, models can be built very quickly, making them suitable for large datasets.

- Decision trees handle non-numeric data (nominal features) very well. This ability minimizes the amount of data transformations but it requires large memory. For example, for this experiment 1280 Mb of main memory is consumed.
- Unlike other statistical taggers (for example, HMM approach) possible contexts are not only the values of the attributes (unigrams, bigrams, trigrams, etc), but also other types of contexts. For example, in a bigram context if the attributes tag_2, tag_1 have possible values <NN>, <ADJ>, <DET>, and so on, then the possible contexts include tag_2=<NN>, tag_2 ≠ <NN>, tag_1≠<DET>, tag_2=<DET> etc which gives additional information to the word to be disambiguated.
- Information in tree can be converted in to easily human interpretable rules. This feature enables experts to evaluate and upgrade its performance.

2.4.7 Limitations of using Decision Trees for POS tagging

Apart from the strengths, decision trees use a "greedy" algorithm for the choice of an attribute for splitting. In other words, all future splits are dependent on the first split; which means the final solution could be very different if a different first split is made.

Furthermore, existing decision tree algorithms (for example ID3 and C4.5) have been well established for small datasets while in real world very large datasets (in millions of samples) are very common. But, more recent algorithms that include SLIQ and SPRINT, both of them are commercial softwares address the problem of scalability [18]. For this study the J48 algorithm, which is a freely available algorithm, is used as we described in

section 5.4 because of the limited resources (budget) for this study to cover the cost for SLIQ and SPRINT.

2.5 SUMMARY

Rule based and corpus based POS tagging systems are the two broad approaches to the problem of POS tagging. Rule based POS tagging systems are linguistic expert dependent while corpus based POS tagging systems are corpus dependent.

TDIDT algorithm is one of the corpus based approaches to classify new instances based on the rules it learned during training. Building the model using training datasets, tree pruning to avoid the problem of overfitting the data, testing the correctness of the pruned tree using independent test sets and finally using the model to classify new instance are the major steeps that must be done during model building and model using.

CHAPTER THREE

THE AMHARIC GRAMMAR AND THE TAGSET

3.1 THE AMHARIC WRITING SYSTEM

The Amharic writing system uses fidel or abugida, which was adopted from Ge'ez. Ge'ez, which belongs to the class of Semitic language family, is the language of literature in Ethiopia. At this time Ge'ez is used mainly in Ethiopian Orthodox Tewahedo Church for religious writing purpose [11].

The current Amharic writing system consists of a core of thirty-three characters (fidel), each representing a consonant and each of which occurs in a basic form and in six other forms or orders according to the vowel which is combined with the basic symbol [21]. Thus there are a total of 238 different characters. The first basic order and the other six orders represent the different sounds of a consonant-vowel combination (a characterization known as syllabic). This characteristic according to Bender [6] makes the Amharic writing system a syllabic writing system.

In a syllabic system, like Amharic, the number of characters (symbols) needed by the language is determined by the number of basic sounds used [ibid]. In addition to the 231 characters, there are nearly forty other characters that contain a special feature usually representing labialization, for example, ኳ(*kwa*) from ከ (*ke*) and ቈ(*qwa*) from ቀ (*qe*). For detailed information the most common characters for Amharic writing adapted from Yacob [41] and Dawkins [11] are listed in appendix A.

3.2 THE PUNCTUATIONS

The Amharic writing system uses some indigenous and foreign punctuation marks (signs) in addition to the Amharic characters [11]. However, only few of them are practically used, especially in computer-written text. The word-separator (*hulet neTb)*, two square dots arranged like colon (:), and sentence-separator (*arat neTb*), four square dots arranged in a square pattern (: :), are the basic punctuation marks in Amharic writing system that are used consistently. Today, the use of *Hulet Neteb* is not seen in modern typesetting. In typesetting its place is almost completely taken over by space.

Lists in Amharic text are separated by an equivalent of comma, '*netela serez*'(፣) followed by ASCII space and '*derib sereze*' (፤), which is the equivalent of semi-colon. In addition to these, the Amharic writing system has borrowed some punctuation marks from foreign languages (ibid). For instance, the usage of exclamation mark '!' and the question mark '?' in Amharic is identical with their usage in foreign languages.

3.3 AMHARIC MORPHOLOGY

Natural languages have complex systems to create words and word forms from smaller units in a systematic way. The part of linguistics which deals with the formation of words and their internal structure is referred as morphology. The basic assumption behind morphology is that the infinity of words of a language are produced from a finite collection of smaller units [38].

The basic building blocks in morphology are morphemes [4; 13; 38]. They are the smallest unit in language to which a meaning may be assigned or, alternatively, as the

minimal unit of grammatical analysis. For example, consider the following Amharic words:

1. *bEt* *bEt* 'house'
2. *temariwoc* *temari-woc* 'students'

The first word (*bEt*) is the smallest meaningful unit that can't be subdivided further. If we try to divide it further in to smaller units /*bE*/ and /*t*/, we get it's constitute sounds; but none of them has a meaning in isolation. In other words it is a word as well as a morpheme. By contrast, the second word *temariwoch* can be sub divided in to two morphemes: *temari* and /*-woc*/ each of them are meaningful Amharic morphemes. The suffix /*-woc*/ is attached to the word *temari* to show plural form. Such a process of making a morpheme or a word (or word components) is called morpheme realization [37]. Moreover, different forms of morpheme realizations are called a morph; while all different forms used to represent a morph are called allomorphs [ibid].

Like other languages, Amharic morphemes can be free or bound morphemes. A free morph form a word by its own; while a bound morph occurs only in combination with other forms. For example, *bEt* is a free morpheme and *–woc* is a bound morpheme.

3.3.1 Types of morphological Processes

Morphological processes can be categorized into inflectional and derivational [9; 38]. The two processes differ in the type of words they produce. Derivational processes create new words of different word class (for example generating a noun from a verb). While

inflectional process does not change the word class of the newly created word. The following are examples to clarify this concept:

1. *bEt* + *oc* = *bEtoch*

   noun bound morph noun

2. *gizE* + *awi* = *gizEyawi*

   'time' 'temporary'

   noun bound morph Adjective

Example 1 shows inflectional process of word formation since the newly formed word *bEtoch* and the base word *bEt* are under noun category. While in number 2, *gezE* (noun) is changed to an adjective (*gezEyawi*) by adding the suffix */-awi/*. This process of change from noun to adjective shows derivational morphological process. Detailed discussion for inflectional and derivational morphological processes is found in [1; 4; 11; 13; 29].

3.3.2 Constitutes of Amharic morph

The process involved in forming words from one or more morphemes is referred as word formation. Amharic words can be formed by affixation, reduplication, and Semitic stem interdigitation, [4; 11; 13; 36]. An affix is a bound morph that is realized as a sequence of phonemes [38]. Affixes can be prefix, suffix or infix. A prefix is an affix that is attached in front of a free morph. For example the Amharic words *ye-ityoPya* 'Ethiopian', *be-mekina* 'by car', *te-sebere* 'broken', and so on are formed by adding the bound morphemes: */ye-/*, */be-/*, and */te-/* on the free morphemes *ityoPya, mekina*, and *sebere* respectively.

A suffix is an infix attached at the end of a free morph. */-oc/, /-woc/, /-u/, /-wa/, /-itu/,* and so on are some of the Amharic suffixes that are added to a free morph to form a new word. For example *lj-oc* 'children', *geberE-woc* 'farmers', *lj-itu* 'children (female)' etc. are some of the Amharic words that are formed by suffixation.

An infix is an affix where the placement is defined in terms of some phonological condition(s) [ibid]. It is the process of forming new words by adding bound morphemes any where except the beginning and the end of a free morph. The example given for reduplication and Semetic stem interdigitation bellow can be also used for word formation by infixation process.

Amharic words can be also formed by reduplication. Reduplication copies some portion of a free morph to form another word form. The following are examples of reduplication in Amharic that are presented in [ibid].

ketefe 'choped' → *ke-ta-tefe* 'choped again and again'

qenese 'decreased' → *qe-na-nese* 'decreased a lot' and so on.

Unlike other languages, Semitic languages including Amharic use non-linear word formation by interdigitation of consonantal roots with vocalic patterns [4; 5; 13; 29; 33; 36]. Amharic root represents sequences of consonants-CCC[8] and are the bases for the derivations of most verbs as well as nouns and adjectives; while a stem is a consonant or consonant vowel sequences [1; 28; 33]. Pattern represents a set of vowels which are

[8] Although the most common root length for Amharic roots is three, there are different views in the range of roots. Detailed discussion on different views of Amharic roots is found in Saba [33].

inserted among the consonants of the root. The most commonly used roots are triconsonantal roots (which contains three consonant forms: CCC). For example, *sbr* is the root form of the verb *sebere* 'he has broken' containing three consonants. Some of the verbs derived from the root *sbr* are:

sebari 'the one who breaks'

sebara 'broken'

sebabare 'the one who has broken again and again'

sbari 'fragment' and so on.

Moreover, sing prefixation and suffixation on the above inflacted word forms, another word form can be formed as follows: *asebari* 'the one who make some one to break', *sebari-woch* 'they broke', etc. detailed discussion on Amharic word formation by Semitic stem interdigitation is found in Abiyot [1]; Nega and Willet [29]; and Saba [33].

3.4 WORD CATEGORIES IN AMHARIC

Like other languages, Amharic words are not ordered randomly. Sounds or characters are ordered in some pattern to form words, so are words to form sentences and sentences to form a certain paragraph or discourse [15]. But all the words in a certain language do not have the same pattern. They also do not behave similarly. Some words may inflect to identical grammatical categories and others may not inflect at all and their grammatical function may be different from the other.

Amharic linguists tried to study all words in Amharic by categorizing in to word classes to avoid the difficulty of studying all words one by one [4; 13]. Word classes are the

different groups under which the words of a certain language are grouped. Research works on the categorization of Amharic words can be seen in two groups: early work and recent work [ibid]. In the early works such as Mersi'hazen [25], Amharic words are categorized into eight classes or parts of speech. These are the noun, Verb, Adjective, Adverb, Preposition, Pronoun, Conjunction and Interjection classes. Baye [4] reduced the early classification of Amharic words into five by putting pronouns under the noun and conjunctions under preposition categories leaving interjections ungrouped in any of the five categories by reasoning out that interjection is a collection of words without syntactic functions.

To determine the class of a word, a single criterion is not enough. A combination of more than one criterion has to be used for word categorization. The recent categorization of Amharic words is based on a combination of three criteria: the meaning (or notional criterion), the form (or morphological criterion) and the position (or syntactical criterion) [4; 13; 15].

The notional criterion method categorizes words based on the meaning of the word. For instance, a noun is a word that has a referent; a verb is a word that denotes actions; an adjective is a word that modifies a noun; a preposition is a word that doesn't have a lexical meaning; and so on. If this criterion is used separately to classify words in to word classes, it doesn't adequately differentiate classes adequately. For example, the Amharic word *mehon* 'to be' and *madreg* 'to do' are nouns that do not have referent [42].

Besides to notional criterion, the shape or morphological criterion is used to categorize words based on their morphological behavior in terms of inflection and derivation. For example, *begoc* 'sheep' and *betoc* 'houses' take the plural marker suffix /*-oc*/. Hence, the Amharic words *beg* 'sheep' and *bet* 'house' are categorized under the same word category noun. But if we consider the Amharic word *bela* 'he ate', it doesn't take the suffix /*-oc*/. Hence, the category of the word *bela* is different from the category of the above words. Its category is under verb. That is, words that inflect for aspect and mood in Amharic can be considered as adverbs. On the other case, words that inflect for case, definiteness, number and gender can be considered as non-verb since this is not a property of verbs [15].

The third criterion, the syntactic criterion, is used to categorize words based on their role or usage in syntax. Here, the position of the word and the surrounding words in a sentence is important to categorize words. For instance, in Amharic, most of the time a verb comes at the end of the sentence.

Mesfin [26] adopted in his work the classification followed by the early scholars but treating nouns and pronouns in the same part of speech class as suggested by Baye [4]; with a justification that early categorization is more exhaustive and could allow the POS tagger to tag words exhaustively and easily.

The ELRC annotation team, which is selected for this study, used the basic eight classes: Noun, verb, pronoun, adjective, adverb, preposition, conjunction, and interjection and

other subclasses for the annotation of the Amharic news documents. For detailed information see Girma and Mesfin [16].

3.4.1 The Amharic Noun Class

Amharic nouns are words used to name or identify entities like a group of things, people, places, ideas, etc, or any subset (or member) of these classes. Nouns takes the suffix */-oc/* as a plural marker. For instance *sew* 'man' and *beg* 'sheep' are singular nouns. But when the suffix */-oc/* is added on each of them they become *sewoc* 'men' and *begoc* 'sheep' respectively to indicate the plural form of the noun *sew* and *beg*. But it doesn't mean that all words that take */-oc/* as suffix is a noun and words which doesn't add */-oc/* as a suffix are not nouns. For instance the words *gobz* 'clever' and *senef* 'lazy' take the suffix */-oc/*. But they are under the category of adjective not in noun. Amharic nouns can be preceded by adjectives to modify it. In the phrase *tlk beg* 'big sheep' the word *tlk* is used to express the noun *beg*. On the other hand, pronouns like *ene* 'I', *ante* 'you' etc doesn't add the suffix */-oc/* but they are under noun category. Amharic nouns can be either primitive or derived. Nouns or words in general are said to be in their primitive forms if they exist in their original form (e.g *gize* 'time') whereas they are referred to as derived if they originated (derived) into their present state from a different, and possibly completely different categories (e.g. *gizEyawi* 'temporary').

3.4.2 The Amharic verb class

The Amharic verbs are those words that usually come at the end of a complete grammatical declarative sentence. The word class 'verb' designates words that express actions, states, times and etc [15]. They also take suffixes (i.e. subject markers) like */-h/*

'you', /-x/ 'she', /-hu/ 'I' and so on, to agree with the subject of the sentence. Amharic verbs can be either primitive or derived like nouns. Unlike derived nouns, verbs can be derived from verbs only [ibid]. Verbs can't be derived from other words different from verb. For instance, the verb *esgedele* 'he made some one to kill some body' is derived from the verb *gedele* 'he killed some body'.

3.4.3 The Adjective Class

These are words that usually come before nouns and serve as modifiers of nouns they precede. But it doesn't mean that all words that precede nouns are adjectives. For clarification, consider the following sentences:

1. *tlq beg* 'big sheep'
2. *yh beg* 'this sheep'
3. *gobez temari* 'cleaver student'

In the above phrases, only *tlk* 'big' and *gobez* 'clever' are adjectives. But in sentence 2, even if the word *yh* 'this' precedes the noun *beg*, it is not an adjective. Like nouns, Amharic adjectives can be either primitive or derived. For example, the adjectives *TEnama* 'healthy', and *sekaram* 'Drunkard' are derived from the noun *TEna* 'health' and the verb *sekere* 'get drunk' respectively.

3.4.4 Prepositions in Amharic

The term 'preposition' refers to those words that will have meanings only when they are attached or used together with other words such as nouns, verbs, pronouns and adjectives. The prepositions category in Amharic comprises affixes such as /*le-*/ 'for', /*ke-*/ 'from', /*be-*

/ 'with', */ye-/* 'that' or 'of', */sle-/* 'for', */wede-/* 'to' and so on. They are used to form adverbial phrases appearing before nouns.

Amharic prepositions can take varied forms. They can separate word or they are prefixed to other words or they can appear as compound prepositions consisting of two parts: prepositional prefixes and post positions with a noun at the middle. For instance, in the phrase *sle sra* 'About work', the preposition sle appears as a simple preposition that stands alone as a separate word. Where as, the preposition *be-* 'by', in the phrase *bemekina meTa* 'he came by car', is prefixed with the word *mekina* 'car'. Detailed discussion about compound preposition is found in [4].

3.4.5 The Adverb Class in Amharic

In Amharic, adverbs that are found in their primitive form are very few [13] and the most common include: *gena* 'yet', *tolo* 'quickly', *kfuNa* 'severely', and so on. Thus, phrases that are combinations of prepositions and some other words like nouns (e.g. *beheyl* 'By force') or prepositions and adjectives (e.g. *bedehna* 'well') perform the functions of adverbs in the language. Adverbs refer to places, time, circumstances etc of the action mentioned by the verb. Besides, there are adverbs of position, also called to as noun adverbs that function either as a noun or as an adverb depending on the context they are used in. For example, in *wde wst geba* 'He entered inside', the word *wst* is used as a noun while in *bewst yseral* 'He works in' *wst* is used as an adverb [42].

3.4.6 Conjunctions in Amharic

This class of words includes both coordinating and subordinating conjunctions. Conjunctions are used to synchronize words, phrases, clauses and sentences. *Ina* 'and' and *weynm* 'or' are examples of Amharic conjunctions. List of Amharic conjunctions followed by a detailed discussion on them is found in Dawkins [11]. As it has been pointed so far at the beginning of this chapter, this study uses the early method of Amharic POS classification developed by ELRC in which conjunctions and prepositions are treated as independent POS categories. However, one problem with this trend of classification is the fact that sometimes the same forms (e.g. *sle*, *be*, and *ke*) forms are used both as prepositions and as conjunctions. This categorical ambiguity of such words can be solved by making careful analysis of the context in which the words are used.

3.4.7 Numerals

Numbers in Amharic can be represented in two ways: using symbols and using words. The symbols that are used to represent Amharic numbers consist of twenty (20) single characters. They represent numbers one to ten, multiples of ten (twenty to ninety), hundred and thousand. These characters are derived from Greek letters [6], and in order to make them look like the Amharic characters the symbols are modified by adding a horizontal stroke above and below. The Ethiopian number system doesn't have the concept of place values like Hindu Arabic numerals. Moreover, there is no symbol in Amharic number system to represent zero (0). The concept of commas and decimal points are not also included in the Amharic number system. A list of the Amharic cardinal numbers is found in [11; 21].

In addition to symbols, words are also used to represent numerals (of course ordinal numbers) in Amharic. i.e. ordinal numbers in Amharic are formed from their equivalent cardinal numbers and the suffix */-Na/*.

Example:	**Cardinal Gloss**	**Ordinal Gloss**
	and 'one'	*andeNa* 'first'

Like English, Amharic has also compound numerals and numerals that indicate distribution and partition of something whole. The following are examples to illustrate these.

Compound numerals: *and meto semanya hulet* 'one hundred and eighty two'

and meto semanya huleteNa 'one hundred eighty second'

Distributive numerals: *hulet hulet* 'two two'

Partition numerals: *gmax* 'half'; *siso* 'one third'; *rub* 'quarter'

Words or symbols that represent numbers are grouped under numeral category. The numbers under this category can be cardinal and ordinal numbers.

3.4.8 Interjections

Amharic has many words or phrases that are used to express emotions like sudden surprise, pleasure, annoyance and so on. Words that are used for such purposes are called interjections. For example, words like gox! 'Well done! tahnks!' , EndEta! 'Of course!', Enja! 'Don't know!' and so on are common interjections in the language.

Amharic interjections can stand alone or can appear any where in a sentence. i.e. they have no syntactic connection with any other words that may occur with them; they are discourse words [4; 15].

Example: *goS!*

goS! srayen Cereshu. 'Oh! I finished my work.'

srayen Cereshu goS! 'I finished my work oh!'

3.5 TAGSET FOR AMHARIC WORD CLASSES

The tagsets and the corpus developed by ELRC annotation team is used for this study as we discussed in section 1.4.2. The annotation team used eleven main tagsets (noun, verb, adjective, preposition, adverb, pronoun, conjunction, numeral, interjection, punctuation and unclassified) and different sub classes of noun, verb, adjective, pronouns, and numeral to annotate Amharic news documents.

3.6 SUMMARY

This chapter presents issues in Amharic that are related to this study based on recent and early works. Although researchers in the area classify Amharic words into five, seven, or eight parts of speech categories, for this study, the categories adopted by the annotation team is applied as discussed in section 3.5.

Generally speaking, Amharic words may be classified under one of the aforementioned word classes. Nevertheless, knowledge of the different word classes is not enough for the task of categorizing words of a given sentence into their proper classes. Moreover, some Amharic words ,like *abebe* 'a proper name' or 'blossomed', *bekele* 'a proper name or 'it grew', and so on, can be categorized in more than one word category (e.g. as noun and as verbs) depending on the context they are used. Recall also the case of some of the prefixes like */sle-/*,

/be-/, */ke-/* etc. that can be categorized as both conjunctions and prepositions. Hence, there should be a mechanism, more than one criterion with careful investigation, to disambiguate lexical classes of such words.

Understanding the Amharic writing, morphology and grammar helps to extract orthographic and morphological information for each word during training and testing dataset preparation to this study.

CHAPTER FOUR

CORPUS PREPARTATION AND ALGORITHM DESIGN

4.1 ARCHITECTURE OF THE SYSTEM

Figure 4.1 depicts the overall architecture of the system. The system starts from a raw text. Then the text is tokenized and manually annotated (for this experiment annotation is not made since the corpus is already annotated by the ELRIC annotation team). To smooth the inconsistencies related to tag usage for compound words and transcription, the sample corpus is manually preprocessed (more information is found in section 4.3). Following this, the features are extracted for each word in the sample corpus both automatically using one or more algorithms and manually to generate training and testing set for the classification algorithm (the detail is presented in section 4.4 and 4.5). Then the classification algorithm builds model from the training set and applies the built model to test new instances (i.e. it tags each word in the test set using the built model) and displays performance evaluation of the built model to users. The algorithm builds model based on information gain measure which is already discussed in section 2.4.

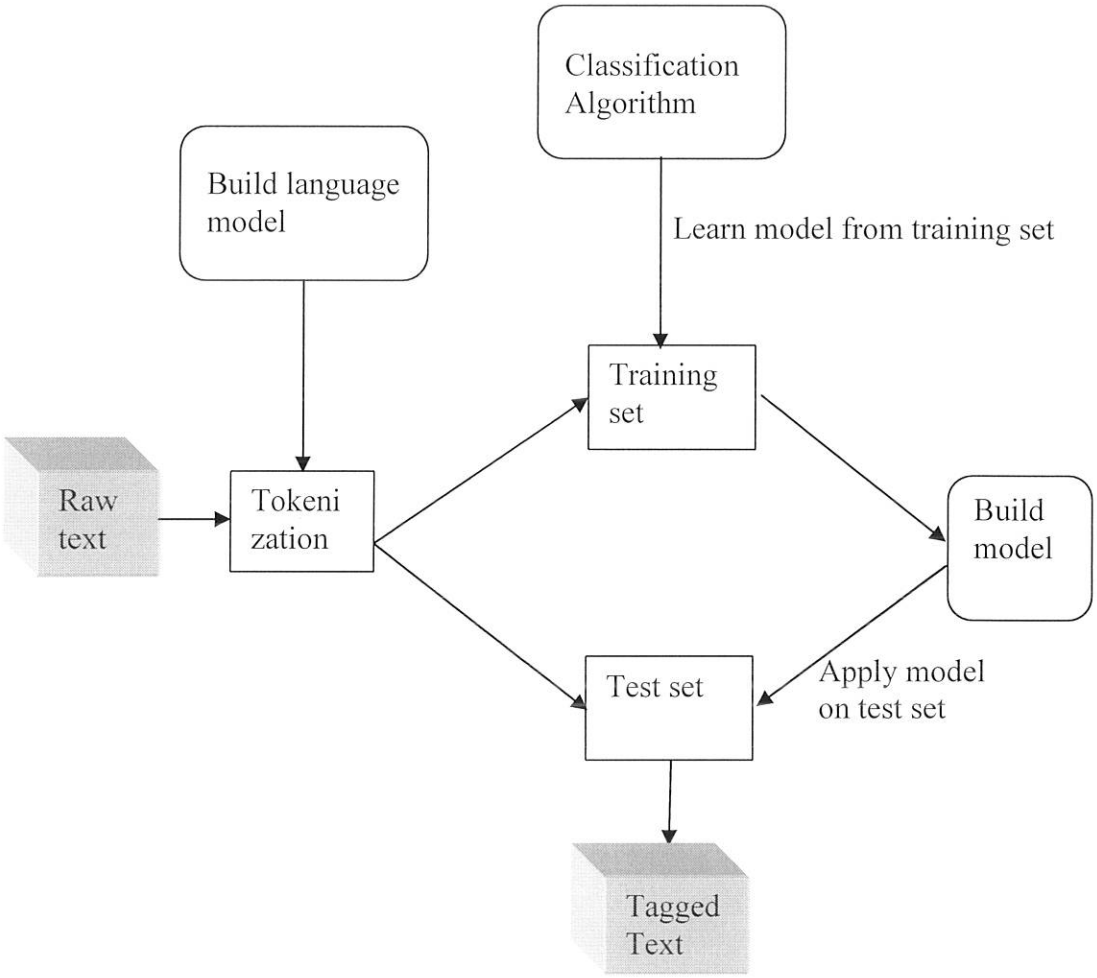


Figure 4.1 Architecture of the Amharic POS Tagger

4.1 SAMPLING TECHNIQUE

As indicated in section 1.4.2, the corpus developed by ELRC is used in this research. However, from high level inspection it is observed that the corpus has some inconsistencies that need manual preprocessing in order to use it in this study. For example, consider the following portions of sentences which are taken directly from the corpus:

1. *...mktl* <N> *ministr* <N> *ato* <ADJ> *tesfayE* <N> *sodano-* <PUNC> *yeserateNana* <NPC>...
2. *...mktl* <N> *ministr* <N> *ato* <ADJ> *belay* <N> *ljgu* <N> - <PUNC> *yegbrna* <NP> *mktl* <N> *ministr* <N>...
3. *... beteCemarim* <CONJ> *beager* <NP> *wsTna* <PREPC>...
4. *...Teqlay* <ADJ> *ministr* <ADJ> *meles* <NC> *zEnawina* <NC>...
5. *...prEzidant* <ADJ> negaso <N> *gidadana* <NC> *Teqlay ministr* <N> *meles* <N> *zEnawi* <N>... and so on.

In number 1, *sodono (* which is a proper name for Ethiopians) is attached with hyphen symbol (–) and received a <PUNC> tag while in number 2, the hyphen symbol (–) and the proper name *ljgu* considered as two separate words and received two different tags: <PUNC> and <N> respectively. The other inconsistency in the tagging process is observed in number 4 and 5. If we consider the Amharic word *Teqlay* 'prime', it is tagged as an adjective (<ADJ>) in number 4. But in number 5, *Teqlay* and *ministr* are considered as compound words and received a <N> tag. The two examples shows that the presence of inconsistence tag usage during the tagging process. Finally, in number 3 the Amharic word *wsTna* is tagged with <PREPC>. The problem here is that, the tag <PREPC> is not found in the list of the tagsets that are used for tagging the corpus (compare with the tagset used by the annotation team which is attached in appendix B). Reviewing the corpus will reduce such types of inconsistencies that are found in the corpus. But, for this study, preprocessing manually or automatically to smooth the inconsistencies observed in the corpus is mandatory to improve the accuracy of the experiment.

On the other hand, preprocessing the whole corpus manually for this study is not feasible as it is time consuming, laborious and costly. To alleviate the aforementioned problems vis-à-vis the existence of inconsistencies in the corpus and the constraints associated with manual preprocessing, eight hundred sample sentences (which are around twenty thousand words) are taken for this study.

Since all non XML sentences in the corpus are considered to be equally important for this study, systematic sampling method is used to select the samples from the whole corpus excluding XML tags that describe the title, date, document name and so on. This can be done by arranging the target population according to some ordering scheme and then selecting elements at regular intervals through that ordered list. It is important that the starting sentence is randomly chosen from the first sentence to the k^{th} sentence in the list to avoid bias in the selection of samples. In this case, k is the ratio of population size and sample size (K=population size/sample size) and represents the interval for selection. For this study a sample of 800 sentences is selected from 8079 non XML sentences. That is, the value of K is fixed to 8079/800=10 (rounding to integer values). After the value of K is fixed and the selection starts from a random start, 5 in this case, the next successive selections are every k^{th} sentence in the list until the list is exhausted. That is, the first selection is the 5^{th} sentence and following these the next successive selections are the 15^{th}, 25^{th}, 35^{th} and so on sentences until the end of the list of sentences in the population.

Below are the general steps that are followed through out the sampling process:

1. Select all non-XML tag sentences form the corpus
2. Determine the value of k
3. Generate a random number to fix the starting sentence for selection and
4. Apply consistent pattern for further selection until end of list is reached

4.3 MANUAL PREPROCESSING

To reduce the impact of the inconsistencies that are found in the corpus on the accuracy of the experiment, a careful manual preprocessing is done on the selected sample corpus. The manual preprocessing includes checking and correcting the inconsistencies related to transcription and tag assignments for each word in the sample corpus.

4.3.1 Checking and correcting transcription inconsistencies

There is some intermixed use of Latin alphabets to represent Amharic characters *('fidel')*. This is due to either typographical errors or Amharic character to Latin conversion error. Just to mention some, the English letters *T, t, c, C, na*, and *nW*a are observed to represent the Amharic characters ጥ, ጠ, ፅ, ቸ, ኗ, and ኘ respectively (see the representation for each character from appendix A to compare their usage in this sample). This phenomenon reduces the accuracy of the experiment due to the possibility of considering same words as different distinct two or more words and two or more different words as one word. For example, if the Amharic word ጠቅላይ (*Teqlay*) 'prime' is encoded as *teqlay (if t is used for* ጠ), it is completely changed to a different word ጥቅላይ which is not a meaningful Amharic word; and the two different Amharic words መታ(*meta*) 'he kicked...' and

መጣ(*meTa*) 'he came…' will be considered as same word if either ጀ is encoded by /*Ta* / or ጣ is encoded by /t*a*/.

Transcription or writing inconsistencies are not consistent- in some sentences a correct transcription or writing of a word is observed, while in other sentence the same word is transcribed differently. That is, it is difficult to smooth transcription or writing inconsistencies using algorithm(s); rather reviewing the corpus for such inconsistencies by the annotation team or any other interested party will reduce it. But for this study, each word is checked and corrected manually while the prefix and the suffix information are extracted.

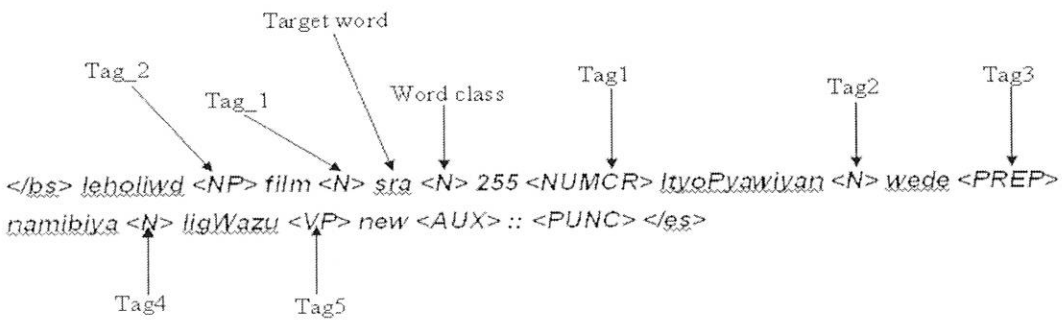
4.3.2 Checking and correcting tagging inconsistencies

Besides to encoding and typographic inconsistencies, the corpus has inconsistencies related to tag assignment to each word in the corpus as we have seen in section 4.2. Building a classification model on such inconsistent data degrades the accuracy of the built model. From thorough inspection, inconsistent tag usage for compound words is highly observed. To reduce the aforementioned inconsistencies, all compound words that received one tag in the sample corpus are thoroughly analyzed for their consistent tag assignment in other sentences they occur. If the compound words observed in other sentences are tagged separately, then the observed compound words are retagged separately. On the other hand, if they are tagged as compound words in their existence most of the time, then their tag assignment is kept intact. For example, consider the compound word found is *kilo meter* 'Kilometer' <*NN*>. If the two words (*kilo* and *meter*) are tagged as compound words in most other sentences, then the white space is

removed and their tag is kept intact. On the contrary, if the two words are tagged separately frequently, then the compound word is retagged as *kilo* <NN> and *meter* <NN>.

4.4 TRAINING AND TESTING DATASETS

After all the necessary preprocessing is done on the sample corpus training and testing datasets, 90% for training and the remaining 10 % is used for testing-using 10-fold cross validation test option, are prepared as shown in table 4.1. The table has ***Tag_i***, ***Tagi***, *where* $i \in \{1, 2, \ldots, 5\}$, ***Target word***, ***Has numeric character?***, ***Prefix***, ***Suffix*** , ***Beginning of sentence?***, ***First character***, ***Last character***, and ***Word class*** attributes. The description of each attributes used for this study is shown in table 4.2. The attributes ***Tag_i*** and ***Tagi***, refers to the i^{th} left and right neighboring word tags (context) from the target word respectively. If the i^{th} left word from the target word doesn't exist, for example if the target word is the beginning of a sentence, then the left i^{th} contexts(Tag_i) is assigned 'NA' symbol to mean the i^{th} left context is not applicable. Similarly, right contexts from the target word that are beyond the sentence boundary is filled 'NA'. For clarification, consider the following sentence which is taken from the sample corpus:



The symbols *</bs>* and *</es>* are beginning and end of sentence markers respectively. If we take ***sra*** *'work'* as target word, then ***Tag_1*** has a value *<N>* which is the tag of the previous word (the tag of ***film)***, ***Tag_2*** is filled with *<NP>*, which is the tag of the second previous word, and the other left context attributes (i.e. ***Tag_3***, ***Tag_4*** *and* ***Tag_5***) are filled with 'NA' to mean not applicable as described before. Similarly, the right contexts of the target word ***sra*** are ***tag1***, ***tag2***, ***tag3***, ***tag4***, *and* ***tag5*** with values *<NUMCR>*, *<N>*, *<PREP>*, *<N>*, and *<VP>* respectively.

In addition to context information, the table contains the ***prefix*** and the ***suffix*** information of the target word in column 14 and 15 respectively. The prefix and suffix of the target word will provide useful information to assign unambiguous POS tag for the word to be tagged especially for morphologically rich languages like Amharic. For example, if the Amharic word ends with */-oc/* most of the time the word is a noun.

For this study all the prefixes and the suffixes of each word in the corpus are extracted and included in the dataset manually if the word is composed of from more than one morpheme. But some words may not have a prefix or suffix or both. To represent this fact, a value "NP" and "NS" are filled under the prefix and suffix column respectively to denote the word has no prefix and/or no suffix respectively.

Tag_5	Tag_4	Tag_3	Tag_2	Tag_1	Target word	Tag1	Tag2	Tag3	Tag4	Tag5	Has numeric character?	Beginning of sentence?	Prefix	Suffix	First character	Last character	Word class
NA	NA	NA	NA	NA	leholiwd	<N>	<N>	<NUMCR>	<N>	<PREP>	no	yes	le	NS	le	d	<NP>
NA	NA	NA	NA	<NP>	film	<N>	<NUMCR>	<N>	<PREP>	<N>	no	no	NP	NS	fi	m	<N>
NA	NA	NA	<NP>	<N>	sra	<NUMCR>	<N>	<PREP>	<N>	<VP>	no	no	NP	NS	NFC	NLC	<N>
NA	NA	<NP>	<N>	<N>	255	<N>	<PREP>	<N>	<VP>	<AUX>	yes	no	NP	NS	NFC	NLC	<NUMCR>
NA	<NP>	<N>	<N>	<NUMCR>	ItyoPyawiyan	<PREP>	<N>	<VP>	<AUX>	<PUNC>	no	no	NP	awian	I	n	<N>
<NP>	<N>	<N>	<NUMCR>	<N>	wede	<N>	<VP>	<AUX>	<PUNC>	NA	no	no	NP	NS	we	de	<PREP>
<N>	<N>	<NUMCR>	<N>	<PREP>	namibiya	<VP>	<AUX>	<PUNC>	NA	NA	no	no	NP	NS	na	ya	<N>
<N>	<NUMCR>	<N>	<PREP>	<N>	ligWazu	<AUX>	<PUNC>	NA	NA	NA	no	no	li	u	li	zu	<VP>
<NUMCR>	<N>	<PREP>	<N>	<VP>	new	<PUNC>	NA	NA	NA	NA	no	no	NP	NS	NFC	NLC	<AUX>
<N>	<PREP>	<N>	<VP>	<AUX>	.	NA	NA	NA	NA	NA	no	no	NP	NS	NFC	NLC	<PUNC>

Table 4.1 Sample training and testing dataset

NO.	Attribute	Description	Value
1	*Tag_i*	used to hold the i^{th} previous word tag from the target word	The tagset which are used for annotating the corpus and 'NA' (e.g. <N>, <ADJ>, <AUX>, etc.)
2	*Tagi*	used to hold the i^{th} following word tag from the target word	Same as 1
3	*Target word*	Holds the target words for every sentence in the sample corpus	Any word in the sample corpus including punctuation marks
4	*Has numeric character?*	Used to hold numerical information for each word	{yes , no}
5	*Beginning of sentence?*	Holds information if the target word is the beginning of a sentence or not	{yes, no}
6	*Prefix*	holds the prefix of the target word if it has; otherwise 'NP'	English letters (A-Z, a-z, and NP)
7	*Suffix*	holds the suffix of the target word if it has; otherwise 'NS'	Same as 6
8	*First character*[9]	Holds the first character of the target word	(A-Z, a-z, NFC)
9	*Last character*	Holds the last character of the target word	(A-Z, a-z, NLC)
10	*Word class*	Holds the tag of the target word	Same as 1

Table 4.2 Description of attributes used for this study

4.5 ALGORITHM DESIGN

The algorithms discussed in this section are meant for the selection of sample corpus for this study from the population (entire corpus) and for the extraction of contextual and morphological information for each word in the sample corpus.

[9] The first and the last Amharic characters may be represented using more than two symbols (a single Amharic character may be transcribed using more than one Latin characters). For example *me* and *na* are the first and the last character of the Amharic word *mekina* 'car' respectively.

The sample generator algorithm shown in Figure 4.2 opens file (the population corpus) and passes it as an argument to the sentence selector algorithm shown in Figure 4.3 and waits until a value is returned back. After the values are returned from the sentence selector algorithm, the sample generator algorithm accepts the number of sample that must be taken from the entire corpus (population) from a user (step 2). Then it calculates the interval for selection (step 3), generates a random number (step 4), starts selection randomly (the m^{th} sentence, step 5), and it continues selecting sentences followed by writing the selected sentences to a secondary storage device (step 6) until the end of the file.

1. *read list of sentences*
2. *get sample size from user*
3. *calculate k = (# of sentences in the population) (sample size)*
4. *generate a random number m between 1 and k*
5. *select the m^{th} sentence*
6. *while end of list of sentences is not reached, do*
   select the K^{th} sentences in the corpus
   write the selected sentence to a file
7. *end while*

Figure 4.2 Sample generator Algorithm

The algorithm in Figure 4.3 selects the title and the body of the document from other non relevant XML tag types (to this experiment) like date of document creation, file name, etc. Finally, it passes the selected sentence to the sample generator algorithm as an argument and task is accomplished. Identifying the key features that uniquely differentiate the title and the body of the document from other non relevant sentences is a challenging task. For this case, the corpus is thoroughly analyzed and a key feature ('<'

symbol) uniquely identify the relevant and the non relevant sentences and handles all cases. That is, all non relevant sentences begin with '<' symbol and those sentences that doesn't begin with '<' symbol are selected and put in array buffer and passed to the sample generator algorithm shown in Figure 4.2.

```
1. Initialize an array of strings
2. i=1
3. open a corpus file
4. while not end of file do
       if sentence i is not xml tag types
           read sentence i from corpus
           put sentence i in to array buffer
           increment i by 1
5. end while
6. pass the content of the array buffer to the sample
   generator function
7. close file
```

Figure 4.1 Sentence selector algorithm

The context extractor algorithm shown in Figure 4.4 opens the sample corpus and extracts information like: left context, right context, and whether the word is the beginning of a sentence or not. The algorithm has two loops. The first loop (outer loop) starts at number 4 and the second loop (inner loop) starts at A. The outer loop continues until all the sentences in the sample corpus are visited and the inner loop continues until end of sentence marker is found for every sentence j. That is, for each word in sentence j, the inner loop extracts left and right contexts and beginning of sentence information. If end of sentence marker is found, the value of j is incremented by one and the outer loop reads the next sentence from list and the inner loop continues the usual process for this sentence also.

```
1. initialize array buffers of strings
2. j=1
3. open sample corpus file
4. while not end of file do
        read sentence j
        i=1
        while end of sentence marker is not reached do
            read word i from sentence j and put it in to an array buffer
            if word i is beginning_of_sentence j
                assign 'yes' to array
             if no
                assign 'no' to array
            read the tag of word i and assign it to an array buffer
            if the n previous and following words are with in the sentence
               read the n previous word tags from the target word (word i) and assign them to
                  array buffer
               read the n following word tags from the target word (word i) and assign to
                  array buffer
             if not
                assign 'NA' to array buffer
             increment i by one
          end inner while
          increment j by one
5.  end outer while
6.  first_last_cahrinfo=First_Last_character (Target_word)  //function calling
7.  numeric=numeric_character (Target_word)       //function calling
8.  write the content of the array to file
9.  close file
```

Figure 4.1 Context extractor algorithm

If all the sentences in the list (sample corpus) are exhausted, then two function calls are made: *First_Last_character* algorithm (Figure 4.5) and *numeric_character* algorithm (Figure 4.6) and waits until values are returned back. Finally, the function writes all the

information extracted in a csv comma separated file format for each word in a file with .csv extension; which is one of the valid file format of the Weka data mining tool.

The existence of some compound words that are separated by white space makes the implementation of this algorithm (Figure 4.4) difficult to extract left and right contexts. The words in the sentence are tokenized by white space and the pattern is expected to be w_1 <tag> w_2 <tag> .... But if a compound word separated by white space is found, the pattern will be affected: w_1 <tag> w_2 w_3 <tag> ... which makes difficult to recognize the left and right contexts of the target word. This problem is solved by removing the white space between compound words and the algorithm handles all cases.

```
First_Last_character (array[ ])     //function name
    initialize array buffers
    i=1
    do
      if word  i is not punctuation or word i is not digit
                put the first character in to array
                put the last character in to array
      if not
        assign 'NFC' to array buffer   // NFC is used to represent no first
            character
        assign 'NLC' to array buffer   // NLC is used to represent  no  last character
    until end of word is reached
    return the content of the array to calling function
```

Figure 4.5 First and last character extractor algorithm

The purpose of the algorithm shown in Figure 4.5 is to extract the first and the last character of each word in the sample corpus. The first and last characters provide

orthographic information about the word that is going to be tagged. It accepts its input from context extractor algorithm and it extracts the first and the last character of each word and assigns to array buffer provided that the word is not punctuation or digit. If the word is a punctuation or digit, 'NFC' and 'NLC' is assigned to array buffer to mean the word has no first and last character respectively. Finally, it returns the content of the array to the calling algorithm.

```
numeric_character (array[ ])      //function name
1. initialize an array buffer
2. i=1 //variable initialization
3. do, until end of list is reached (end of word is reached)
     if word i contains digit
        assign 'yes' to array
     if not
        assign 'no' to array
4. return the content of the array to the calling function
```

Figure 4.6 Numeric character information extractor algorithm

The purpose of this algorithm (Figure 4.6) is to provide the presence or absence of numeric characters within the word. This information is useful to disambiguate words that have numeric character(s). The function accepts list of all words from context extractor algorithm. Then for each word in the list it assigns 'yes' value to the array buffer if the word contains digit, otherwise 'no' value is assigned to it. Finally, it returns the content of the array to the calling algorithm (Figure 4.4) and task is accomplished.

4.6 TOOL SELECTION

Different tools are searched from literature and the internet. While searching, some are found to be proprietary and are beyond the budget of the study. As a result only open source tools are considered for comparison.

Tanagra, Rapid miner, and Weka, data mining tools are evaluated based on the task the tool is intended for; the algorithms it supports; the computer architecture and operating system which the software runs on; the maximum number of records that the software can handle; familiarity of the researcher with the tool and visualization capabilities.

The Tanagra data mining tool, which runs on window operating system, is tested using the dataset prepared for this study. But it is unable to do computation due to the presence of many distinct attribute values in the dataset (e.g. in this dataset *Target word* attribute has 7501 distinct values). This is the basic issue for this dataset; hence it is not necessary to consider other parameters to compare it with other tools. That is, it is not considered for this study.

The Rapid miner data mining tool is a pure java environment like the Weka data mining tool for knowledge discovery and runs in any platforms. It has more data mining operators (more than 400 operators or algorithms), where machine learning library WEKA is fully integrated, and it needs more time to master and use it for this study. Moreover, I haven't seen any difference in the use of decision tree classification algorithm, which is the algorithms that is used for this study, between the Rapid miner

Moreover, I haven't seen any difference in the use of decision tree classification algorithm, which is the algorithms that is used for this study, between the Rapid miner and the Weka data mining tool. As a consequence the Weka data mining tool is selected and used for this study.

The Weka data mining tool consists of a collection of machine learning algorithms for solving real-world data mining problems. It is written in Java, an object-oriented programming language that is freely available for all major computer platforms. It operates under Linux, Windows, and Macintosh operating systems [32]. Java also provides a uniform interface to many different learning algorithms, along with methods for pre- and post-processing and for evaluating the result of learning schemes on any given dataset.

In addition to the Weka data mining tool, python programming language is used for processing the corpus so that it is ready for the tool which is selected for this study. Python is used due to its string manipulation capability, the existence of many built in functions and the familiarity of the researcher to the language. The training and testing dataset is saved in a csv file format using Microsoft excel.

4.7 EXPERIMENTATION PROCEDURE

Not all the attributes that are used to model the language are equally important. Some may result in noise rather than giving useful information to the word to be tagged. Hence, to differentiate those attributes that contribute to goodness of split, different experiments are done. Based on the result of the experiment, only those attributes that provide useful information to the word to be tagged are considered iteratively. An experiment is also

conducted using attribute selection algorithm by inputting all the attributes and letting the algorithm to select the best attributes.

The experiment is conducted using two sets of attributes. These are: context attributes (**CA**) and other attributes (**OA**). The **CA** attributes represent the left and right neighboring tags and the **OA** attributes represent orthographic and morphological information attributes. The **CA** attributes include:

- left context attributes (**LCA**):*Tag_5, Tag_4, Tag_3, Tag_2, Tag-1*
- Right context attributes (**RCA**): *Tag1, Tag2, Tag3, Tag4, and Tag5*

The **OA** attributes include: *Target word, Has numeric character?, First character, Last character, Beginning of sentence?, Suffix,* and *Prefix attributes.*

In this study a total of six experiments are conducted. The first experiment is conducted to determine the appropriate split options using default (of course the appropriate training and test options is determined using experiment 2) test options: 10-fold cross validation and 66% split test options. The 10-fold CV iteratively partitions the data into two: 90% for model building and the remaining10% for validating. Then the classifier is evaluated 10 times and the average result is displayed into the output area. Similarly, the 66% test option uses 66% of the dataset (12958 instances) for model building and the remaining 34% (6676 instances) of the dataset for testing the built model.

Following the first experiment, a comparison is made between the selected default test option and the other randomly chosen test options (70%, 75%, 80%, 85%, and 90% test

options). One test option is selected based on accuracy and used for the next successive experiments to select the appropriate (best) attributes from others.

Finally, using the selected attributes, split and test options, two experiments are carried out to see the effect of minimum object and confidence factor on the accuracy of the model.

CHAPTER FIVE

EXPERIMENTATION

5.1 INTRODUCTION

This chapter is mainly concerned with the experimental details and the analysis following the output of the experiment. The experiment is carried out using the weka 3-7-0 data mining tool as we discussed in section 4.6. Weka package has a classifier sub package where decision tree classification algorithm is one component of it. The next section highlights the weka classifier as it is the selected algorithm to conduct this experiment.

5.2 WEKA CLASSIFIER

The Decision Tree classification, which is a sub package of the Weka classifier, supports sixteen learning and testing algorithms: *ADTree, BFTree, DecisionStamp, Id3, J48, J48graft, LADTree, LMT, MSP, NBTree, RandomForest, RandomTree, REPTree, Simplecast, and UserClassifier*. However, not all of these algorithms could be used in the experiment; as a consequence, one algorithm is selected and used to the problem of POS tagging based on the following criteria:

- The accuracy of the algorithm
- The time it takes to build the model
- The size of the tree it generates
- The capability of the algorithm to handle nominal attributes and
- The maximum number of records that the algorithm can handle

The *ADTree* algorithm was not tried because the algorithm cannot handle non-binary class (the experimental data has nominal classes). Similarly, MSP cannot handle nominal class. Moreover, ID3 algorithm is not considered as there is a J48 algorithm, which is extension of ID3 algorithm, with more functionality. To select one from the remaining thirteen algorithms, an experiment is done using small number of instances (3926 instances)[10] and keeping the other factors constant (default values).

NO.	Algorithm	Accuracy	Time (in sec.)	#of nodes
1	BFTree	61.06 %	2023.92	33
2	Decision Stump	50.87 %	0.05	NA
3	J48	**78.43 %**	**2.84**	**31280**
4	LADTree	68.46	2215.77	NA
5	RandomTree	61.27 %	0.64	388390
6	REPTree	63.07 %	10.27	7747

Table 5.1 Comparison of Weka's learning and testing Algorithms

Legend:
NA - stands for not applicable

The algorithms *FT, J48graft, LMT, NBTree, RandomForest, SimpleCart* and *UserClassifier* are not scalable; using this small amount of data (3926 instances), these algorithms need more than1408 Mb of main memory to do computation. As a result, a comparison is made only on the remaining six algorithms. We can observe from table 5.1 that the J48 algorithm yields better accuracy (78.427%) than all the other algorithms. Moreover, the time taken to build the model is acceptable (2.84 sec). As a consequence, the J48 algorithms is selected and used for this study.

[10] The need to use smaller number of values is just to minimize the computation time for each algorithm

5.3 EXPERIMENTATION DETAILS

The experiment process starts by inputting the dataset using the Weka's preprocessing sub package. The screen shot of the Weka preprocess section is shown in Figure 5.1 and the frequency and percentage of the tagsets observed in the sample corpus are represented in Table 5.2.

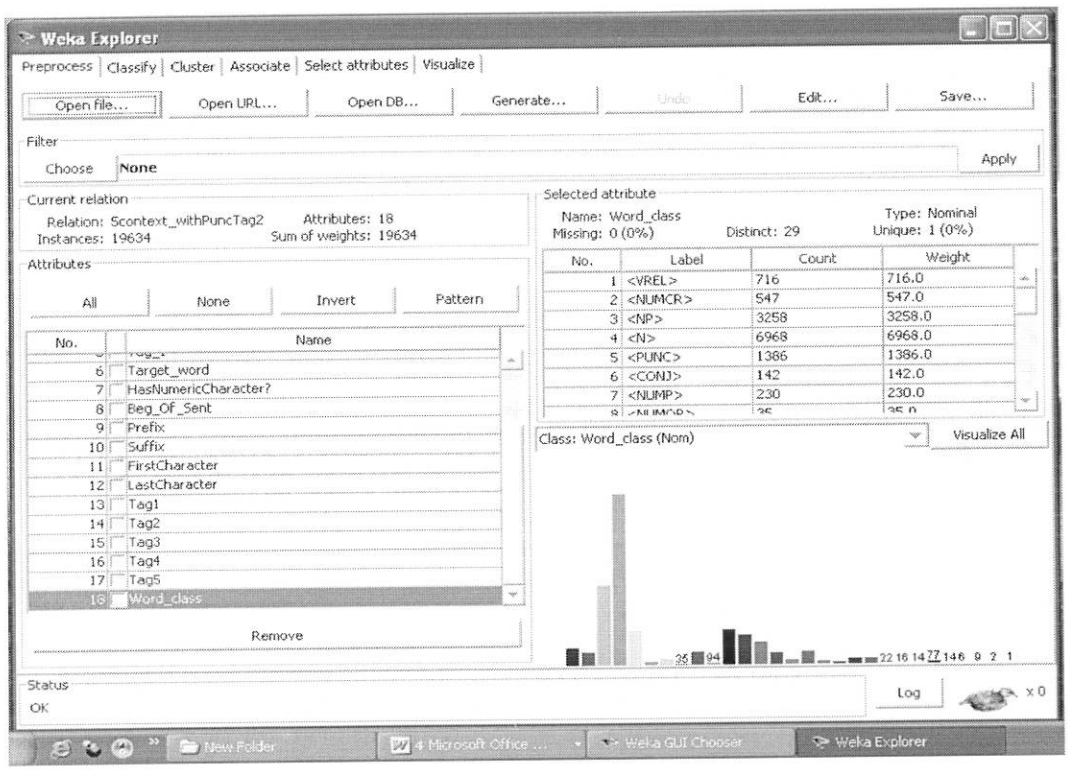


Figure 5.1 Screen shot of the Weka preprocess with opened dataset

We can see from Table 5.2, 6968 words, which account 35.5 % of the sample set, are tagged as noun (<N>) and only one word (which is almost zero percent of the sample set) is tagged as numeral attached with conjunction (<NUMC>). Moreover, interjections are not seen in the sample set at all. The reason is that their occurrence in the population is very small (only 3) and the probability to be selected in the sample corpus is almost zero.

NO.	Tag	Frequency	Percentage
1	<VREL>	716	0.036467353
2	<NUMCR>	547	0.027859835
3	<NP>	3258	0.165936641
4	<N>	6968	0.354894571
5	<PUNC>	1386	0.07059183
6	<CONJ>	142	0.007232352
7	<NUMP>	230	0.011714373
8	<NUMOR>	35	0.001782622
9	<PREP>	532	0.027095854
10	<VPC>	94	0.004787613
11	<VP>	1466	0.074666395
12	<V>	1215	0.061882449
13	<ADJ>	910	0.046348172
14	<NC>	490	0.024956708
15	<NPC>	215	0.010950392
16	<VN>	511	0.026026281
17	<PRONP>	168	0.008556586
18	<AUX>	109	0.005551594
19	<ADJP>	252	0.012834878
20	<ADV>	229	0.011663441
21	<VC>	22	0.001120505
22	<ADJPC>	16	0.000814913
23	<UNC>	14	0.000713049
24	<PRON>	77	0.003921768
25	<ADJC>	14	0.000713049
26	<PRONC>	6	0.000305592
27	<PRONPC>	9	0.000458389
28	<NUMPC>	2	0.000101864
29	<NUMC>	1	5.09321E-05
sum		**19634**	**1**

Table 5.2 Summary of tagsets observed in the sample dataset

After the data input and the entire necessary preprocessing task is done, a total of 6 experiments were carried out using the j48 algorithm as we described in section 4.7.

Moreover, the run parameters and the outputs of the respective experiments are presented in tables (Table 5.3, Table 5.4, Table 5.5, Table 5.6, Table 5.7 and Table 5.8).

Experiment 1: To determine the appropriate split option

The J48 algorithm has two split options for model building and testing: binary split 'False' (default) and binary split 'True'. The 'True' option results in only two child nodes: one is if test node is equal to X and the other is if the test node is not equal to X, where X is one of the test node attribute values. While 'False' option produces more than or equal to two children for every parent node (i.e. the parent node or the test node branches out to each of its possible values). For example, if the test attribute is *Tag1* with attribute values <N>, <ADJ>, and <V>, then *Tag1* branches out into three nodes: *Tag1*=<N>, *Tag1*=<ADJ> and *Tag1*=<V>. To decide the appropriate split option for this dataset, an experiment is conducted by varying the test option and keeping all other things constant (default values). The output of the experiment is presented in Table 5.3.

Run	# of instances	Attributes used	Test Option	Binary split	Accuracy	Time	# of nodes
1	19634	All (18)	66%split	False	81.61 %	9.47	41699
2	''	''	66% split	True	83.73 %	490.78	1709
3	''	''	10-fold cross validation (CV)	False	82.64 %	8.69	41699
4	''	''	**10-fold CV**	**True**	**84.16 %**	**488.63**	**1709**

Table 5.3 Effect of split options on performance of the algorithm

Even though the time taken to build the model in 'False' split option is short (compare the time taken to build the model for 'True' and 'False' option from table 5.3), it is less accurate as compared to the 'True' option. The reason for the decrease in accuracy for 'False' split option is that, it is more complex than the 'True' split options (24 times more nodes than the 'True' test options)-this leads the problem of overfitting the training data. As a consequence only binary split 'True' option is selected and consistently followed in the next successive experiments. Moreover, from the two default test options, the 10-fold cross validation test option results in better accuracy than the 66% test option. This is due to appropriateness of 10-fold cross validation test option for small datasets to get more training set [40]. It splits the dataset into 10 (each partition has approximately equal number of instances) and at each iteration (there are a total of 10 iterations), the nine partitions are used for training set and the remaining one partition for test set. Finally, the average accuracy rate of each runs is used as the overall performance of the algorithm. To check the fact that 10-fold cross validation is suitable for small dataset, experiment 2 is conducted and its result also confirmed this fact.

Experiment 2: On different test options

This experiment is conducted just to check the appropriateness of 10-fold CV test options with other test options (or validating the appropriateness of the 10-fold CV test option for small dataset). The experiment is carried out using test options in 5 intervals starting from 70% to 90% and the 10-fold CV test option. In this context, X% of split denotes X% of the total dataset is used for model building and the remaining (1-X%) of the dataset for model testing.

Test option	70%	75%	80%	85%	90%	10-fold (CV)
Accuracy	83.99 %	83.88 %	83.68 %	83.7 %	83.60 %	**84.16 %**
Time	450.56	455.2	456.88	454.72	452.63	**488.63**
Tree size	1709	1709	1709	1709	1709	**1709**

Table 5.4 Summary of results on different test options

The test results in table 5.4 shows that the 10-fold cross validation test option still performs better (84.16 %) than any other test options. Hence, for the next successive experiments only the 10-fold cross validation test option is used consistently.

Experiment 3: to determine appropriate context attributes

The aim of this experiment is to determine the appropriate window size of neighboring contexts (left and right contexts). The experiment is carried out by keeping all other things constant[11] (i.e. CF=0.25 (default), Binary split= 'True' (which is determined in experiment 1), MinObj=5 (the change from 2 to 5 is just to minimize the computation time)).

The input attributes to the algorithm are only context attributes. The context attributes are partitioned in to three: **LCA** (from run 1 to run 5), **RCA** (from run 6 to 10), and a combination of left and right contexts (from run 11 to run 19).

[11] Constant in this context is used to mean all the values listed in the bracket are the same for all runs (from run 1 to run 19 in Table 5.5)

Run	Context	10-fold CV test results		
		Accuracy	*Time (in sec)*	*#of nodes*
1	T_1[12]	40.77 %	0.59	19
2	T_2-T_1	41.25 %	3.53	77
2	T_3-T_2-T_1	**41.52 %**	**15.09**	**211**
4	T_4-T_3-T_2-T_1	41.45 %	31.16	451
5	T_5-T_4-T_3-T_2-T_1	40.82 %	64.89	663
6	T1	41.86 %	0.09	7
7	T1-T2	43.56%	1.55	33
8	T1-T2-T3	**44.02%**	**7.42**	**175**
9	T1-T2-T3-T4	44.03 %	23.06	307
10	T1-T2-T3-T4-T5	43.36%	31.64	417
11	T_1-T1	45.2 %	5.3	117
12	T_2-T_1-T1	46.36 %	17.13	253
13	T_2-T_1-T1-T2	**47.13 %**	**32.17**	**511**
14	T_3-T_2-T_1-T1-T2	47.08%	53.45	701
15	T_3-T_2-T_1-T1-T2-T3	46.72 %	85.38	1051
16	T_4-T_3-T_2-T_1-T1-T2-T3	46.52 %	103.09	1129
17	T_4-T_3-T_2-T_1-T1-T2-T3-T4	46.37 %	139.55	1147
18	T_5-T_4-T_3-T_2-T_1-T1-T2-T3-T4	45.81%	176.58	1403
19	T_5-T_4-T_3-T_2-T_1-T1-T2-T3-T4-T5	45.31 %	193.24	1277

Table 5.5 Summary of left and right context experiments

As we can see from table 5.5, a maximum accuracy of **47.13 %** (run 13) is observed when a combination of two left and two right contexts are used to disambiguate the target word. The reason for achieving only **47.13%** accuracy is that, only context information (with out the target word) are used. The accuracy is even less than this, if only left or right contexts are used to disambiguate the target word. This shows the need to include additional information like the target word, and orthographic and morphological information to improve the accuracy of the algorithm. As a consequence, the left and right two contexts are selected and used consistently in the next successive experiments.

[12] T_1, T_2, …, T4, T5 represent Tag_1, Tag_2, …, Tag4, Tag5 respectively just to save space.

The selected context attributes denoted by (**SCA**) include: *Tag_2, Tag_1, Tag1*, and *Tag2*.

Experiment 4: deciding best combination of SCA and OA attributes

This experiment is carried out to select best combination of the **SCA** and the **OA** attributes. The experiment is conducted first using the **SCA** and ***Target word*** (at run 1) as shown in table 5.6. The result shows a better accuracy (61.94 %) than using simply context attributes in experiment 3. That is, the presence of each words along with their contexts provide better information to the problem of POS tagging than using only context attributes.

There are two options to carry out experiment using the selected context attributes and the **OA** to see the effect of **OA** attributes on accuracy. The first option is by adding one attribute from the other attributes recursively with out replacement on selected context attributes and observing their effect on accuracy. If higher accuracy is registered with the addition of features from **OA** then, the features/attributes are considered to be important and those attributes that results in less accuracy are considered as not important to be considered for further experiment. The other alternative is reducing one attribute recursively from **OA** attributes with replacement from a combination of selected context attributes and **OA** attributes and observing their effect on accuracy. In this case, when the accuracy of the algorithm is lower than the accuracy registered using **SCA** and **OA** attributes with the reduction of one or more attribute(s) from **OA**, then the reduced attribute is said to be important and if the accuracy increases with the reduction of one or

more attributes, then the reduced attribute is said to be not important. For this experiment, simply the second option is applied.

From run 2 to 10, different combinations of attributes are inputted and their corresponding accuracy, time to build the model, and the size of the tree (# of nodes) are recorded using the same table. At run 2, **SCA** and all the **OA** attributes including the ***Target word*** attribute are used by the algorithm. From run 3 to 9 one attribute is reduced from the **OA** attributes recursively with replacement and keeping the **SCA** unchanged in all cases; At run 10 the **SCA** are removed; and finally at run 11, all the 18 attributes are inputted to the algorithm and let the algorithm select relevant attributes. The Weka's attribute selection algorithm, *cfsSubsetEvaluation* (default) with best first search, selects from an input of 18 attributes only 6 attributes (*Target_word, HasNumericCharacter?, Prefix, Suffix, FirstCharacter* and *Word_class*) (at run 11). But the accuracy of the classifier obtained using these attributes is substantially lower as compared to the manually selected attributes and not used for further experiment.

The result from table 5.6 shows that the reduction of ***Beginning of sentence?*** attribute from a combination of the **SCA** and the **OA** attribute sets results in a better accuracy (83.89 %, run 4) than in any other combinations. It also produces minimum number of nodes except from the attributes used in run 2, 8, 10 and 11. The time taken to build the model is also acceptable as compared with other runs. This phenomena show that the addition of ***Beginning of sentence?*** attribute in the attribute sets doesn't provide useful information. As a result it has to be removed form the attribute sets and the rest attributes

are used for the next two experiments to see the effect of minimum number of objects and confidence factor on accuracy, size of the tree and the time taken to build the model.

Run	Attributes used	10-fold CV test results		
		Accuracy	*Time(sec)*	*#of nodes*
1	**SCA** + Target word	61.94%	834.99	823
2	**SCA + OA**	83.85 %	37.11	625
3	Has numeric character? removed	83.23%	45.36	651
4	Beginning of sentence? removed	**83.89%**	**35.58**	**645**
5	Prefix removed	82.02 %	26.75	737
6	Suffix removed	82.75 %	27.5	673
7	First character removed	82.83%	56.42	653
8	Last character removed	83.52 %	42.88	555
9	Target word removed	79.53%	10.08	671
10	SCA removed	83.32 %	34.14	515
11	Using the algorithm to select attributes(16 inputted and 6 used)	82.23 %	34.11	405

Table 5.6 Summary of SA and OA experiments

Experiment 5: Effect of confidence factor on accuracy

Experiment 5 is carried out to identify the effect of confidence factor on the result of the algorithm. Confidence factors are used for tree pruning to get optimal tree to the dataset [40]. Generally, lower values for confidence factor incur more pruning and higher confidence factor value produce complex tree size. Over pruning (over simplified tree) are more general and may not be able to handle some of the cases- which may results in lower accuracy. On the other hand unpruned (or complex trees) can handle most of the cases in the training set but its performance on the new test set is low as described in section 2.4.4. As a consequence, experiment has to be done to determine the appropriate confidence factor which is suitable for this dataset. The experiment is conducted by keeping all other factors constant (same as to experiment 3) and varying the value of the

confidence factor. The output of each run for confidence factors: 0.05, 0.1, 0.15, 0.5, 0.25, 0.3, 0.35, and 0.4 are shown in table 5.7.

Run	Confidence factor(CF)	10-fold CV test results		
		Accuracy	*Time(sec)*	*#of nodes*
1	0.05	84.5 %	338.98	1113
2	0.1	84.73 %	328.16	1269
3	0.15	84.86 %	329.52	1303
4	0.2	**84.9 %**	**387.31**	**1375**
5	0.25	84.81 %	375.14	1471
6	0.3	84.64%	412.89	1635
7	0.35	84.59 %	483.81	1843
8	0.4	84.5 %	495.47	1921

Table 5.7 Effect of confidence factor on accuracy and tree size

Better accuracy for this dataset is observed at run 4 when the value of CF is 0.2. When the value of the CF is smaller than 0.2, the algorithm builds model in a shorter time with smaller nodes but the accuracy on the test set is low. On the other end, when the value of the CF is greater than 0.2, the performance of the algorithm is poor in all factors. Hence, a 0.2 CF value is fixed and experiment 6 is conducted by varying the number of minimum objects denoted by (MinOb).

Experiment 6: Effect of MinOb on accuracy

This experiment tries to see the effect of the minimum number of instances for a leaf node to be considered for splitting. Leaf nodes that has greater than or equal number of instances to the given minimum number are subject to split in to two child nodes and it

becomes an internal node (or parent node). Nine runs are made by varying the value of MinOb and the results are displayed in table 5.8.

Run	MinOb	10-fold CV test results			
		Accuracy	*Time in (sec)*	*#of leaves*	*#of nodes*
1	2	**84.9 %**	**387.31**	688	**1375**
2	3	84.5 %	122.3	433	865
3	4	84.24 %	69.49	361	721
4	5	83.91 %	30.19	289	577
5	6	83.72 %	28.64	253	505
6	7	83.48 %	31.75	226	451
7	8	83.22 %	20.78	218	435
8	9	82.9 %	17.14	198	395
9	10	82.6%	14.19	187	373

Table 5.8 summary of effect of minimum object on accuracy

The results in table 5.8 show a contradiction on accuracy and size of tree. Small MinOb value results in complex tree size but higher accuracy. The reason for this dilemma is the dependency of rules on the size of the tree in general and the number of leaf nodes in the tree in particular. For each leaf node, a rule is formed by ANDing the values of all internal test nodes until a leaf node is reached. Complex tree means many rules are generated from the training set and it handle cases which are less frequent. On the contrary, little number of rules is generated for smaller tree size and it is more general and is inadequate to classify infrequent cases in the test set.

On the other hand, more complex trees are difficult for searching and the search algorithm will be trapped at local maxima and couldn't reach a global maxima. Hence, a compromise has to be made between accuracy and size of the tree. For this case, a value 2 is selected for MinOb as the accuracy is better than the other MinOb values (of course

with maximum tree size). But the tree size is not much complex for POS tagging classification problems as it requires from few hundreds to thousands of expert developed rules for other language as we discussed in section 1.1. For this experiment, the algorithm learns and builds 688 rules (i.e. one rule per leaf node) and achieved an accuracy of **84.9%** as shown in Figure 5.2.

The accuracy achieved in this study is lower as compared to the previous works-Yenewondim (93.88%) and Mesfin (90%). The reason for this variation can be categorized in to two: the source of the dataset and the number of tagsets used.

The source of the dataset used for this study is news documents, which is different from the previous researchers (who used sample sentences from a single book). The document used for this study is developed by different authors (editors), i.e. writer independent, while the previous researchers used a book written by a single author. A dataset used for training and testing from a single author gives better accuracy than from multiple authors. The reason for this fact is that, if model is built and tested using sources from a single writer document, the possibility of unknown words in the test set is less as the writer follows consistent writing for same words that appear in different sentences. On the other hand, if the source is from multiple authors, then the probability of unknown words in the test set is high as different authors use different writings for the same word or concept; for example, *rsememhr* vs. *dayrEkter* for 'Director', *eyroplan* vs. *ewroplan* for 'Plane', *mekina* vs. *kamiwon*, *for 'car'* etc.

The other possible reason is, the number of tagsets used for this study (30 tagsets) which is higher than the tagset used by the previous researchers (both used less than 30 tagsets). The difference in number of tagsets usage has its own impact on accuracy. More number of tagsets means more number of class attribute values. That is, classifying each instance in the training set in to these (30) class values has its own impact on the resulting accuracy than classifying the same instance in to a lesser class attribute values.

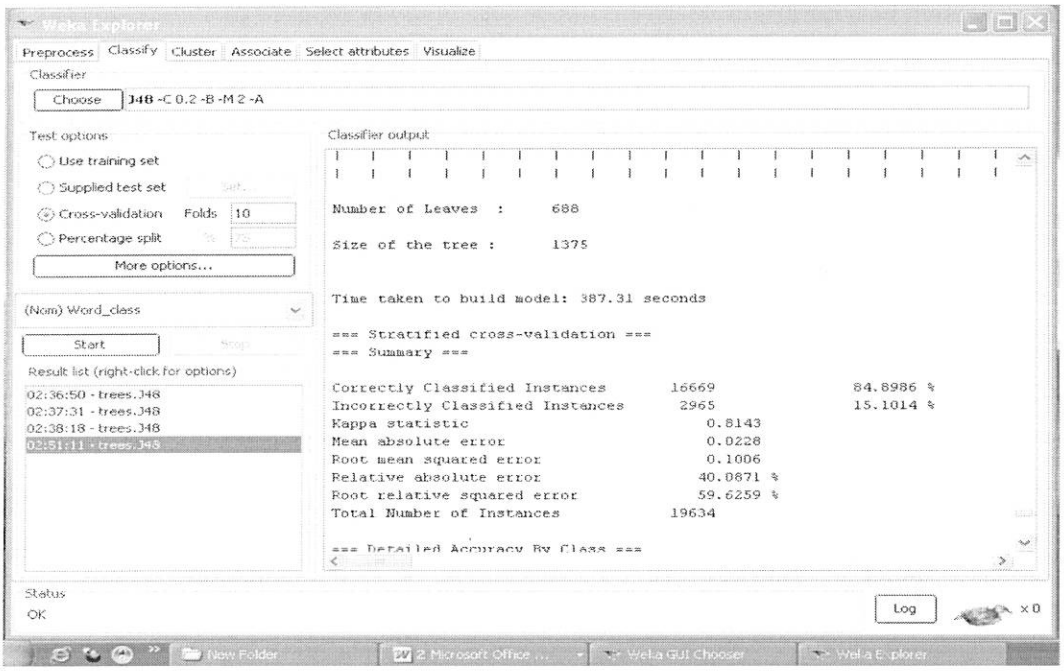


Figure 5.2 The Screenshot of the J48 Classifier

5.4 SUMMARY OF RESULTS

This study tries to identify the features that are important to tag Amharic texts. In this work, a combination of contextual, morphological and orthographic information-with linguistic analysis is used to tag Amharic texts which make it unique from the previous approaches of Mesfin [26] and Yenewondim [42], who used only lexical and contextual information. Experimental results show that, lexical information is the most useful

information to tag Amharic text (run 9 from experiment 4), followed by the prefix, suffix and context information. Moreover, the window size of the context attributes is fixed experimentally (two left and two right contexts) unlike other approaches that use either unigram, bigram, or trigram approaches that considers only the tag of the n-previous words tag. The model used for this study is also flexible to add important features at any time with linguistic analysis.

The main contribution of this study is the identification of some of the important features to tag Amharic texts-left and right contexts, prefix, suffix, the first one character, the last one character, existence of digits with in the word, and the word itself experimentally. As a consequence, this study can be taken as the starting point to model the language with more linguistic analysis in order to increase the accuracy of the algorithm.

Apart from its advantage in the identification of contributing features to the problem of POS tagging for this study, decision tree algorithm has constraints related to memory, which was the main challenge during experimentation. For example, for the dataset amounts to 9634 words or instances, it consumes 1280 MB of main memory to train and test the system.

CHAPTER SIX

CONCLUSION AND RECOMMENDATION

6.1 CONCLUSION

Availability of large sets of tagged corpora is essential in many natural language processing applications like: information retrieval, stemming, speech synthesis, machine translation, information retrieval and so on. For other languages like English, German, Spanish, etc there are large tagged corpora for experimental purpose. But researchers in Ethiopian languages, except the corpora developed by ELRC for Amharic, suffer a lot due to lack of sufficient tagged corpora. There are some attempts to POS tagging for Amharic. But we can say that all of them are for academic exercises and, as to the researcher knowledge, none of them are implemented to tag Amharic documents.

This study explores the application of decision tree for POS tagging for Amharic. The experiment shows that neighboring contexts beyond two (*Tag_2, Tag_1, Tag1, Tag2*) are useless. Saying it differently, the most useful context information to the problem of POS tagging using decision tree for Amharic is the two left contexts and the right two contexts in combination which is identical with the finding by Marquez and Rodrìguez [23] for Spanish.

The other important thing is that, using only the frequency of the target word with its neighboring contexts are not sufficient for POS tagging especially for morphologically rich languages like Amharic (run 1 in Table 5.6). Orthographic information like the

prefix and the suffix of the word, the first and last two characters of the target word, and the existence of numeral characters in the target word increase the accuracy of the built model. The accuracy achieved using 10-fold cross validation on 19634 instances (words) is **84.9 %,** which is promising but needs further improvement using additional parameters like the stem of the word, is the word a multi-word (compound word) or not, the infix information of the word.

6.2 RECOMMENDATION

The presence of tagged Amharic corpus by ELRC reduced the extra resources (time, many, labor, etc) needed for tagging (dataset preparation) for NLP in general and for POS tagging in particular for Amharic. But from thorough inspection, the corpus has inconsistencies in relation to tag assignment to compound words. In some cases compound words received one tag; while in other cases they are tagged separately. Moreover, the corpus is not balanced. For example currency exchange news item is observed frequently while other many news items are seen only once. Reviewing the corpus will smooth such types of inconsistencies that are found in it. On the other hand, the corpus is not representative as it is developed only from one source -news documents that cover the Ethiopian year 1994 (2001-2002 in Gregorian year). Sources from religious (books, periodicals), education (science, social science), General fictions (novel, short stories), and so on should be incorporated in the corpus in proportional quantity to make it a more representative of the language.

The presence of standard prefix and suffix list is important for many NLP tasks like POS tagging, information retrieval, morphological analyzer, etc. In other well studied

languages like English this is not a problem for researchers in the field of NLP. But the same is not true for Amharic. Researchers in the Amharic language developed and used prefix and suffix list only for their need-which results duplicate of efforts. As a consequence, developing prefix and suffix list to Amharic and other Ethiopian languages solve the problem.

POS tagging is still an active research area even for well studied languages like English, French, German, etc. The reason is that it is one of the basic components of other higher level NLP applications. As a consequence small errors in the tagging system propagate and degrade the accuracy of the higher level applications. To the best of my knowledge, there is no practical POS tagger developed for Amharic. All the attempts are for academic exercise and are not implemented for practical applications. As a consequence, there is a room for other researcher on the field of POS tagging for Amharic. The following are suggested as possible research areas:

1. POS tagging problem can also be solved using different ambiguity levels (classes). All words in the corpus can be grouped in to different ambiguity classes based on the set of their possible tags (noun-verb, noun-adjective, noun-verb-adjective, and so on ambiguity classes). Developing POS tagger for Amharic based on ambiguity class level is also another research area.

2. Investigate the potential of hybrid approaches for Amharic POS tagging. For example, Transformation based POS tagging using constraint rules combines statistical and rule based POS tagging and it is applied for English language.

Developing a hybrid POS tagger for Amharic and other Ethiopian languages is another potential researchable area in the field of NLP.

3. Compare and contrast the different approaches, e.g. HMM, decision tree, multi-layer perceptron, etc, for Amharic POS tagging and other local languages using the same corpus.

REFERENCE:

[1] Abiyot, B. (2000). Developing Automatic Word Parser for Amharic Verbs and Their Derivation. Master Thesis, Addis Ababa University.

[2] Atelach, A. (2002). Automatic Sentence Parsing for Amharic Text: An Experiment Using Probabilistic Context Free Grammars. Master Thesis, Addis Ababa University.

[3] Atelach, A., Askar, L., and Mesfin, G. (2003). Natural Language Processing for Amharic: Overview and Suggestions for a Way Forward , in Proceedings of TALN 2003 Workshop on Natural Language Processing of Minority Languages and Small Languages, Batz-sur-Mer, France. Retrieved on 25 may, 2009 from: http://www.sciences.univ-nantes.fr/info/recherche/taln2003/ articles/alemu.pdf

[4] Baye, Y. (1987 E.C). የአማርኛ ሰዋሰው. አዲስ አበባ. ት.መ.ማ.ድ።።

[5] Beesley, K. (1998). Consonant Spreading in Arabic Stems. Xerox Research Centre Europe Grenoble Laboratory 6, chemin de Maupertuis 38240 MEYLAN, France. Accessed on 16 Jun, 2009, from: http://www.aclweb.org/anthology/P/P98/P98-1018.pdf

[6] Bender, M., Sydeny, W., and Roger C. (1976). The Ethiopian Writing System. In Bender et el (Eds.) Languages of Ethiopia. London: Oxford University Press.

[7] Brill, E. (1992). A simple rule-based part of speech tagger. Proceedings of the third Annual Conference on Applied Natural Language Processing, ACL, Toronto, Italy. Retrieved on 5 March, 2009 from: http://*www.ldc.upenn.edu/acl/A/A92/A92-1021.pdf*

[8] Cambridge University (2005). Cambridge advanced learner's dictionary. 2nd ed., New York: Cambridge printing press.

[9] Crytal, D. (1997). The Cambridge Encyclopedia of Language. New York, Cambridge University press.

[10] Daelemans, W., Zavrel,J., Berck, P., and Gilis,S. (1996). MBT: A Memory-Based Part of Speech Tagger-Generator. Proceedings of Fourth Workshop on Very Large Corpora. Acessed on 15 October, 2008 from: www.citeulike.org/user/gboleda/article/1099358.

[11] Dawkins, C.H. (1969). The Fundamentals of Amharic. A.A Sudan interior mission.

[12] Federal Democratic Republic of Ethiopia (1995). The Ethiopian Federal Democratic Republic Constitution. Addis Ababa: Berhanena Selam Printing Enterprise.

[13] Getahun, A. (1989 E. C). ዘመናዊ የአማርኛ ሰዋሰው በቀላል አቀራረብ። አዲስ አበባ፣ ንግድ ማተሚያ ድርጅት።

[14] Giorgos, O., Dimitris, K., Thanasis, P., and Dimitris, C. (nd). Decision Trees and NLP: A Case Study in POS Tagging. Greece, University of Patras. accessed on 15 October, 2008 from: http://dke.cti.gr/pubs/confs/acai99a.pdf.

[15] Girma, A. (2006). Issues on lexicography: Understanding word classes. Ethiopian language research center. Addis Ababa; Addis Ababa University printing press.

[16] Girma, A. and Mesfin, G. (2006). Manual Annotation of Amharic News Items with Part-of-Speech Tags and its Challenges, ELRC Working Papers Vol. 2; No.1. Retrieved on 18 March, 2009 from: http://nlp.amharic.org/research/papers/by-year/2006/tagging-girmaandmesfin.pdf

[17] Guilder, L. (1995). Automated POS tagging: A brief Overview. Accessed on 8 May, 2009 from: http://ccl.pku.edu.cn/doubtfire/NLP/Lexical_Analysis /Word_Segmentation_Tagging/POS_TaggingOverview/POS%20Tagging%20Overv ew.Html

[18] Han, J., and Kamber, M. (2001). Data Mining: concepts and Techniques. San Fransisco; Morgan kufman Publishers.

[19] Hirshman, B. (nd). Training Set Properties and Decision-TreeTaggers: A Closer Look. Carnegie Mellon University Pittsburgh. Accessed on 26 April, 2009 from: http:// www.cs.cmu.edu/~epxing/Class/10701-06f/project reports/hirshman.pdf

[20] Jurafsky, D., and Martin, J. (2005). Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition. 2nd ed., NJ: Prentice Hall

[21] Leslau, W. (1965). An Amharic textbook of everyday usage. University of California. Los Angeles.

[22] Marquez, L (1999). Part of speech tagging: A machine learning approach based on decision trees. Phd thesis, Universitat Politècnica. Retrieved on 1 August, 2009 from: *citeseer.ist.psu.edu/329989.html*

[23] Marquez, L., and Rodn'guez, H. (1995). Towards Learning a Constraint Grammar from Annotated Corpora Using Decision Trees. ESPRIT BRA_7315, WP #15. Accessed on 15 September, 2008 from: http://*www.springerlink. com/index/57237389271804r8.pdf*

[24] Marquez, L., Padro, L. and Rodriguez, H. (2000). A Machine Learning Approach to POS tagging. Journal of machine learning. Vol. 39, No. 1. Netherlands: Springer. Accessed on 12 November, 2008 from: http://www.springerlink.com/content/p8w6123175766757/fulltext.pdf

[25] Mersehazen, W. (1934 E. C) **ያማርኛ ሰዋሰው። አዲስ አበባ፤ብርሀንና ሰላም ማተሚያ ድርጅት።**

[26] Mesfin, G. (2001). Automatic Part of Speech Tagging for Amharic Language: An Experiment Using Stochastic Hidden Markov (HMM) Approach. Master Thesis, Addis Ababa University.

[27] Microsoft Research (nd). Natural Language Processing. Microsoft Corporation. Retreived on 12October, 2008 from:http://research.microsoft. com/nlp/

[28] Nega, A., and Willet, P. (2003). The effectiveness of stemming for information retrieval in Amharic. Electronic library and information systems Vol. 37, Number 4 · Acessed on 3 June, 2009 from: http://eprints.whiterose.ac. uk/145/1/willettp1.pdf

[29] Nega, A., and Willet, P. (2002). Stemming of Amharic words for information retrieval. Literary and linguistic computing, Vol. 17, No. 1 UK, Oxford University press. Retrieved on 3 June, 2009 from: http://llc.oxfordjournals.org/cgi/reprint/17/1/1

[30] Quinlan J. (1986). Induction of Decision Trees. Machine Learning, Kluwer Academic Publishers, Boston. Retrieved on 8 December, 2009 from http://www.cs.toronto.edu/~roweis/csc2515-2006/readings/quinlan.pdf

[31] Roberts, A. (2003). Machine Learning in Natural Language Processing. Accessed on 3 July, 2009 from: http://www.andy-roberts.net/misc/latex/sessions/bibtex/bib_example_nat.pdf

[32] Rogers, J. (2001). Data Mining Using the EM Clustering Algorithm on Places Rated Almanac Data. Accessed on 16 July, 2009 from:http://www.galaxy.gmu.edu/ stats/syllabi/inft979/RogersPaper.pdf

[33] Saba, A. (2007).Bilingual Word and Chunk Alignment: A Hybrid System for Amharic and English. Phd thesis, Universität Bielefeld. Accessed on 2 August, 2009 from: http://deposit.ddb.de/cgi-bin/dokserv?idn=985328878&dok_var=d1&dok_ext=pdf&filename=985328878.pdf

[34] Schmid, H. (1994). Probabilistic Part-Of-Speech tagging using decision Trees. In the proceedings of the Conference on New Methods in Language Processing, Manchester, UK. Accessed on 12 April, 2009, From *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.28.1139*

[35] Shanmugam, K. (nd). POS Tagging. AU-KBC research center. Anna University. Retrieved on 15 October, 2008, from:http://www.au-kbc.org/ research_areas/nlp/projects/postagger.html

[36] Sisay, F. (2005). Part of Speech tagging for Amharic using Conditional Random Fields. Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages,

http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=BEF0BD598B74E4410407C9C875E5DB4E?doi=10.1.1.59.7119&rep=rep1andtype=pdf

[37] Tesfaye, B. (2002). Automatic Morphological Analyser: An Experiment Using Unsupervised and Autosegmental Approach, Masters Thesis, Addis Ababa University.

[38] Trost, H. (2000) Computational Morphology, MIT Press, Cambridge. Retrieved on 3 June, 2009 from: http://www.coli.uni-saarland.de/~schulte/Teaching/ESSLLI-06/Referenzen/Morphology/trost-2003.pdf

[39] Witten, I. and Frank, E. (2000). Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco.

[40] Witten, I. and Frank, E. (2005). Data Mining: Practical machine learning tools and techniques, 2nd Edition, Morgan Kaufmann, San Francisco.

[41] Yacob, D. (1996). System for Ethiopic Representation in ASCII (SERA). Retrieved on 18 May, 2009, from: http://www.abyssiniacybergateway.net/fidel/

[42] Yenewondim, B. (2006). Application of multilayer perceptron neural network for tagging parts of speech for Amharic. Master Thesis, Addis Ababa University.

APPENDIX A: The Amharic alphabet ('fidel') adopted from Yacob [41] and Dawkins [11]

	Ordinary characters							Diphthong ('diqala') characters				
1	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
	he	hu	hi	ha	hE	h	ho					
2	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ			ሏ		
	le	lu	li	la	lE	l	lo			lWa		
3	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ			ሗ		
	He	Hu	Hi	Ha	HE	H	Ho			HWa		
4	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ			ሟ		
	me	mu	mi	ma	mE	m	mo			mWa		
5	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ			ሧ		
	`se	`su	`si	`sa	`sE	`s	`so			`sWa		
6	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ			ሯ		
	re	ru	ri	ra	rE	r	ro			rWa		
7	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ			ሷ		
	se	su	si	sa	sE	s	so			sWa		
8	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ			ሿ		
	xe	xu	xi	xa	xE	x	xo			xWa		
9	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ	ቍ	ቋ	ቌ	ቊ
	qe	qu	qi	qa	qE	q	qo	qWe	qWu	qWa	qWE	qWi
10	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ			ቧ		
	be	bu	bi	ba	bE	b	bo			bWa		
11	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ			ቯ		
	ve	vu	vi	va	vE	v	vo			vWa		
12	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ			ቷ		
	te	tu	ti	ta	tE	t	to			tWa		
13	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ			ቿ		
	ce	cu	ci	ca	cE	c	co			cWa		
14	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኈ	ኍ	ኋ	ኌ	ኊ
	`he	`hu	`hi	`ha	`hE	`h	`ho	hWe	hWu	hWa	hWE	hWi
15	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ			ኗ		
	ne	nu	ni	na	nE	n	no			nWa		
16	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ			ኟ		
	Ne	Nu	Ni	Na	NE	N	No			NWa		
17	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ					
	e	u	i	a	E	I	o					
18	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	ኵ	ኳ	ኴ	ኲ
	ke	ku	ki	ka	kE	k	ko	kWe	kWi	kWa	kWE	kWu
19	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ			ዃ		
	`ke	`ku	`ki	`ka	`kE	`k	`ko			ea		
20	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ					
	we	wu	wi	wa	wE	w	wo					
21	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ					
	`e	`u	`i	`a	`E	`I	`o					
22	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ			ዟ		
	ze	zu	zi	za	zE	z	zo			zWa		
23	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ			ዧ		
	Ze	Zu	Zi	Za	ZE	Z	Zo			ZWa		

24	ዩ	ዩ	ዩ	ያ	ዬ	ይ	ዮ					
	ye	yu	yi	ya	yE	y	yo					
25	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ			ዷ		
	de	du	di	da	dE	d	do			dWa		
26	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ			ጇ		
	je	ju	ji	ja	jE	j	jo			jWa		
27	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጐ	ጉ	ጕ	ጓ	ጔ
	ge	gu	gi	ga	gE	g	go	gWe	gWu	gWi	gWa	gWE
28	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ				ጧ	
	Te	Tu	Ti	Ta	TE	T	To				TWa	
29	ፈኈ	ፈኈ	ፈኈ	ፋኌ	ፈኈ	ፋኍ	ፎ኎				ፏ	
	Ce	Cu	Ci	Ca	CE	C	Co				CWa	
30	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ					
	Pe	Pu	Pi	Pa	PE	P	Po					
31	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ				ፇ	
	Se	Su	Si	Sa	SE	S	So				SWa	
32	ፀ	ፀ	ፂ	ፃ	ፄ	ፅ	ፆ					
	`Se	`Su	`Si	`Sa	`SE	`S	`So					
33	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ				ፏ	
	fe	fu	fi	fa	fE	f	fo				fWa	
34	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ				ፗ	
	pe	pu	pi	pa	pE	p	po				pWa	

APPENDIX B: Tagset used by ELRC annotation team

Basic class	Definition of the tag	Code of the tag
Noun	Verbal/ infinitival Noun, formed from any verb form such as active, passive, and repetitive, by attaching the prefix m(ä)-	VN
	Any noun including verbal noun attached with a preposition	NP
	Any noun including verbal noun attached with conjunction	NC
	Any noun including verbal noun with a proclitic preposition and an enclitic conjunction	NPC
	Any other noun; simple or derived	N
Pronoun	Pronoun attached with preposition	PRONP
	Pronoun attached with conjunction	PRONC
	Pronoun with a proclitic preposition and an enclitic conjunction	PRONPC
	Any other Pronoun	PRON
Verb	Auxiliary verb	AUX
	Relative verb	VREL
	Any Verb including relative verbs and auxiliaries attached with preposition	VP
	Any Verb including relative verbs and auxiliaries attached with conjunction	VC
	Any Verb including relative verbs and auxiliaries with a proclitic preposition and an enclitic conjunction	VPC
	Verb (all other) Adjective attached with preposition	V
Adjective	Adjective attached with preposition	ADJP
	Adjective attached with conjunctions	ADJC
	Adjective with a proclitic preposition and an enclitic conjunction	ADJPC
	Any other Adjective	ADJ
Preposition	Preposition	PREP
Conjunction	Conjunction	CONJ
Adverb	Adverb	ADV
Numeral	Cardinal	NUMCR
	Ordinal	NUMOR
	Numeral (cardinal or ordinal) attached with preposition	NUMP
	Numeral (cardinal or ordinal) attached with conjunction	NUMC
	Numeral (cardinal or ordinal) with a proclitic preposition and an enclitic conjunction	NUMPC
Interjection	Interjections	INT
Punctuation	Punctuation	PUNC
Unclassified	Unclassified	UNC
Total		**30 tags**

APPENDIX C: Sample rule produced by J48 classifier

```
=== Run information ===
Scheme:       weka.classifiers.trees.J48 -C 0.2 -B -M 2 -A
Relation:     5context_withPuncTag2-weka.filters.unsupervised.attribute.Remove-R1-3,8,15-17
Instances:    19634
Attributes:   11
              Tag_2
              Tag_1
              Target word
              Has numeric character?
              Prefix
              Suffix
              First character
              Last character
              Tag1
              Tag2
              Word class
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
------------------
Prefix = NP
|  Tag1 = NA: <PUNC> (792.0)
|  Tag1 != NA
|  |  Has numeric character? = no
|  |  |  Tag2 = NA
|  |  |  |  First character = NFC
|  |  |  |  |  Tag_1 = <AUX>: <PUNC> (2.0/1.0)
|  |  |  |  |  Tag_1 != <AUX>: <AUX> (66.0/5.0)
|  |  |  |  First character != NFC
|  |  |  |  |  Target word = nacew: <AUX> (10.0/2.0)
|  |  |  |  |  Target word != nacew: <V> (701.0/6.0)
|  |  |  Tag2 != NA
|  |  |  |  Target word = ,: <PUNC> (373.41)
|  |  |  |  Target word != ,
|  |  |  |  |  Target word = ;: <PUNC> (137.52)
|  |  |  |  Target word != ;
|  |  |  |  |  Last character = na
|  |  |  |  |  |  Target word = wana: <ADJ> (24.0/1.0)
|  |  |  |  |  |  Target word != wana
|  |  |  |  |  |  |  Target word = yhunna: <VC> (4.0/1.0)
|  |  |  |  |  |  |  Target word != yhunna
|  |  |  |  |  |  |  |  Suffix = awina: <ADJC> (10.0/3.0)
|  |  |  |  |  |  |  |  Suffix != awina
|  |  |  |  |  |  |  |  |  Suffix = NS
|  |  |  |  |  |  |  |  |  |  Target word = wanawana: <ADJ> (2.0)
|  |  |  |  |  |  |  |  |  |  Target word != wanawana
|  |  |  |  |  |  |  |  |  |  |  Target word = Indegena: <ADV> (3.0)
|  |  |  |  |  |  |  |  |  |  |  Target word != Indegena
|  |  |  |  |  |  |  |  |  |  |  |  Target word = amna: <ADV> (5.0/1.0)
```

```
| | | | | | | | | | | | | Target word != amna
| | | | | | | | | | | | | | First character = be: <NC> (3.0/1.0)
| | | | | | | | | | | | | | First character != be: <N> (108.0/17.0)
| | | | | | | | | | | Suffix != NS
| | | | | | | | | | | | Target word = medebeNana: <NP> (2.0/1.0)
| | | | | | | | | | | | Target word != medebeNana
| | | | | | | | | | | | | First character = ye: <VPC> (2.0/1.0)
| | | | | | | | | | | | | First character != ye
| | | | | | | | | | | | | | First character = be: <NPC> (8.0/4.0)
| | | | | | | | | | | | | | First character != be: <NC> (428.0/35.0)
| | | | | | Last character != na
| | | | | | | Target word = Ina: <CONJ> (88.35/2.35)
.                                                    .
.                                                    .
.                                                    .
```

Declaration

I, the under signed, declare that this thesis is my original work, has not been submitted as a partial requirement for a degree in any university and that all sources of materials used for the thesis have been duly acknowledged.

Gebeyehu Kebede

September, 2009

The thesis has been submitted for examination with my approval as university advisor.

Name: ERMIAS ABEBE

Signature:

Date: September, 2009