ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

Amharic Part-of-Speech Tagging

Using Hybrid Approach
(Neural Network and Rule-Based)

By Solomon Asres Kidanu

A thesis submitted to the School of Graduate Studies of Addis Ababa University in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science.

September, 2008
Addis Ababa

Addis Ababa University
Faculty of Informatics
Department of Computer Science

Amharic Part-of-Speech Tagging Using Hybrid Approach (Neural Network and Rule-Based)

By Solomon Asres Kidanu

Approved By

Examining Board

1. Sebsibe Hailemariam, Advisor _____

2. _____ _____

3. _____ _____

4. _____ _____

5. _____ _____

Dedicated To:

My grandmother, Emuhaye Manasebe Azagew

Acknowledgement

I have been fortunate enough to have had the support of many people and without whom the completion of this thesis would have been impossible. Firstly, I must thank my advisor Sebsibe H/Mariam for his valuable time, guidance and understanding through this thesis. Sebsibe H. provided enlightening ideas, comments and experience which enabled me to gain the most. Thank you Sebsibe! You are an ideal advisor.

I would like to thank Dr. Girma Awugeche, Director of Ethiopian Languages Research Center, who provides me the necessary data to this research and expertise. He was extremely cooperative.

I would also to thank W/o Amarech Taddem. If you were not sent that laptop, life would be more suffering.

I would also give my gratitude to Tizeta Zewde, Tensae Berihun, Manchilot Gebeyehu and Desalegn Mequent for their time and dedication to read and gave me excellent feedback on my work which helping me to improve it.

Thank you Dn. Birhanu Admas for your encouragement, support and understanding through this study and even apart from this.

I must also thank Tesfaye Bihonegn. You might forget what you did to me when you were Dessie. But I did not.

My dear friends (Ashu, Nesre & Sole): I couldn't have survived through those years if it was not for your understanding and support. I have got you in addition to the degree.

Mane, my grand-mother: everything I have successfully accomplished in life can be accredited back to your absolute love and positive influence.

Finally and most importantly, I would like to thank God who has been constantly helping me.

Table of Contents

List of figures

List of Tables

List of Acronyms

- **ANN-** Artificial Neural Network
- **BP-** Back Propagation
- **ELRC-** Ethiopian Languages Research Center
- **HMM-** Hidden Markov Model
- **MLP-** Multi Layer Perceptron
- **NLP-** Natural Language Processing
- **NN-** Neural Network
- **POS-** Part-of- Speech
- **RMS-** Root Mean Square
- **TBL-** Transformation-Based Learning

Abstract

The accumulation of information in this electronic age is rapidly increasing. Yet we have very little intelligent tools that will help individuals manage this giant information. Natural Language Processing researches are looking closely at this problem and try to build systems that can understand natural languages. Part-of-speech tagging is one attempt in the effort of understanding human languages. It is the assignment of a category to a word which indicates the role of the word in a given context.

There are a lot of part-of-speech taggers for many languages but is not for Amharic language. This study proposes a hybrid method of Neural Network and Rule-based approach for tagging Amharic words. So this method is based firstly on Neural Network and then anomaly is corrected by Rule-based approach. Back Propagation algorithm and Transformation–Based learning method are adopted for the development of Amharic tagger.

Building the tagger with hybrid approach can improve the performance of the tagger. This study sets better Amharic tagsets, large size corpus and uses two methods for better accuracy. To evaluate the proposed method, a number of experiments have been conducted. A large number of data are used to train and test the tagger. The experimental result of this thesis work indicates that 91% and 94% for rule-based and neural network tagger, respectively. But the result reaches to 98% when the experiment has been conducted on the hybrid tagger.

Keywords: *Part-of-Speech, Amharic Part-of-Speech tagger, Tagset, POS tagger.*

Chapter One

Introduction

1.1 Background

Natural language is a language that is spoken, written, or signed by humans for general-purpose communication [37]. It is any language that human beings learn from their environment and make use of it to communicate with each other in their day-to-day activities. Natural language is used to express our knowledge and emotion in order to convey our response to others. English, French, Arabic, and Amharic are examples of natural languages that have their own structural, lexical, semantic, syntactic, morphological, and other descriptions [1, 2].

Natural Language Processing (NLP) is an area of artificial intelligence research that attempts to reproduce the human interpretation of language by machines [37]. Natural language processing, as a field of study, studies ways to understand how information is represented in a given instance like word, phrase, sentence, of a natural language. Specifically, it deals with devising a mechanism that captures structural information of the natural language. The goal of natural language processing is designing and building systems that will analyze understand and generate natural languages. Having such types of systems will enable to communicate with machines as though one is communicating to human agent [29]. The machine should understand first the natural language before processing it. But, understanding a natural language by machine is not easy and straight forward task. The complexity and richness of human language make the language understanding process very difficult [2]. It involves large number of classes and relations whose existence is not transparent from the surface structure of the natural language. This fact makes the assignment of words to their appropriate classes difficult and also to identify the relation they have with each other in the sentence, paragraph or document. Another major

reason is that understanding natural language by machine has ambiguity at structural, semantic and lexical level.

Though, understanding natural language by machines is a complicated problem, there are various approaches under investigation and some of them have succeeded to some extent in making machines to understand natural language [4].

Amharic, which belongs to the Semitic family of languages, is one of the most widely spoken natural languages in Ethiopia. However, it is among the least researched languages [42]. Recently, researches in the development of different NLP tools for analyzing Amharic text have begun [4, 48, 50, 51]. Part-of-speech tagging is one of the research areas on natural languages [31].

Part-of-Speech is a category of words based on their grammatical function such as nouns, pronouns, verbs, adjectives, adverbs, interjections, prepositions and post positions. A part-of-speech or word class is defined as the role that a word plays in a sentence [32].

Part-of-speech tagging is the process of marking up the words in a text as corresponding to a particular part-of-speech based on its definition and context, i.e., relationship with adjacent and related words in phrase, sentence, or paragraph [56]. In this process, special codes or labels will be attached to each word in the corpora indicating its particular category. The codes attached are known as "tags", and the process of attaching them to a word is called tagging. Let's elaborate these concepts using the Amharic sentence; "ይህ መጽሐፍ ነው" (*This is a book*). The tagging process accepts this sentence as input and then generates "ይህ/*ADV* መጽሐፍ/*N* ነው/*AUX*" as an out put. The codes ADV, N and AUX represent an adverb, noun and auxiliary verb respectively so as to indicate part-of-speech of a word in a given sentence [29].

Tagging can be done either manually or automatically. Part-of-Speech (POS) tagger is a tagging system which assigns a tag for each word in a sentence automatically [16]. POS taggers have been successfully applied to assign a single POS to every word in a corpus by considering the context of the word [2]. There are various approaches which are used for building POS taggers for different languages. The most popular ones are: Stochastic, Rule-based, Artificial Neural Network, and hybrid approaches [2, 24].

Stochastic, statistical, approach tries to define the class label probabilistically for a word and the word categories in their contexts. Rule-based tagging is another approach to tackle the problem of automatic POS tagging. This approach requires set of rules to define the appropriate tag for a word. The rules can be designed either manually by linguists or using machine learning techniques. Artificial Neural Network can be used to extract patterns and detect trends that are complex to be noticed by humans. Hybrid approach uses two or more of the above approaches for determining the class of a given word. This research focuses on hybrid approach which combines Artificial Neural Network and Rule-based approaches

1.2 Statement of the problem

POS tagging is one of the tasks in the area of NLP which needs to be given attention. The absence of POS tagging system limits the effort to conduct high level researches such as parsing and machine translation which helps to quickly and easily retrieve information. It also limits works related to morphological analyzers, morphological synthesizers, phrase recognizers, indexer, stemmers, speech recognizers, speech synthesizers, grammar and spelling checkers in the area of Amharic language. Moreover, the absence of POS tagging system costs much time and effort of researchers for annotating large corpora manually in order to address different issues in computational linguistics. Manual annotation of corpus also requires annotators to have deep

knowledge of the structure of the language. Thus, having mentioned all such pitfalls, it is worth to conduct a research and develop an automatic part-of-speech tagger.

Researches have been conducted on POS tagging by different researchers for different languages; among these, two of them, [42] and [55], are on Amharic language. Stochastic Hidden Markov Model (HMM) and multi-layer perceptron Neural Network approaches have been used in [42] and [55], respectively. Though these two approaches require large size corpus, the researchers haven't used enough size which in turn makes the research activity complicated and inefficient. Besides this, these two approaches, by themselves, have their own limitation which hinders to get better performing tagger. Hence, it is necessary to conduct research on POS tagging for Amharic language in order to alleviate the above limitations and enhance the researches done by other researchers.

1.3 Objective of the study

1.3.1 General Objective

The major objective of this study is to enhance the performance of an automated POS tagger for Amharic language using a hybrid POS tagging approach.

1.3.2 Specific Objectives

In order to achieve the above general objective, the following specific objectives are set.

- Collect an appropriate corpus for the research.
- Determine Amharic tagset.
- Determine appropriate features for tagging
- Design neural tagger and rule-based corrector.
- Construct hybrid POS architecture.
- Adapt an algorithm for the development of Amharic tagger.

- Develop prototype.
- Test and analyze the performance of the system.

1.4 Scope of the Study

The study would have been very interesting if it compared all the approaches for POS tagging. However, due to time constraint this study focused only on exploring the efficiency and potential of a hybrid approach, (Rule based and Artificial Neural Network), for automatic part-of-speech tagger for Amharic.

1.5 Methodology

Since this thesis work is intended to develop an automatic POS tagger for Amharic language, the following methodologies have been employed.

- Literatures have been reviewed on Amharic word classes and approaches of POS tagging. Related works are also assessed.
- Data collection: manually annotated corpus has been collected for training and testing purpose.
- Design approach: to enhance the performance of Amharic tagger, Artificial Neural Network and Rule-based approaches have been taken into consideration in order to design the hybrid tagger.
- Development tools: Programming languages; Python 2.5 and MS-Visual Basic 6.0 ,and database management system; Ms-Access have been used to develop the prototype.
- Experimental Analysis: experimental analysis has been conducted according to the result of the neural tagger, rule-based corrector, and hybrid system. To conduct the experiments,

Brain Maker and Brill's tagger are used as a tool for Neural Network and Rule-based respectively. Charts have been used for displaying results.

1.6 Application of POS Tagging

POS tagging is necessary in most applications of NLP such as speech synthesis, speech recognition, machine translation, etc [6, 29]. It is also useful for further linguistic study like analyzing the syntactic structure of sentences in a text, statistical work such as counting the distribution of different word classes in text corpora, etc. POS tagging, not only assign a word to the accurate part-of-speech tag, but also provides other relevant information such as the inflectional categories of the classes, for example, number, person, and case [11]. Words are often ambiguous in their part-of-speech. This ambiguity is normally resolved by the context of a word. POS tagging is vital for word-sense disambiguation. The task of word-sense disambiguation is to examine word tokens in context and specify exactly which sense, meaning, of each word is being used.

POS tagging is useful to discover the linguistic structure of large corpus which facilitates higher levels of language processes like parsing, morphological analysis and synthesis, stemming, information retrieval, and so on [19]. A word's POS is also used to improve the level of precision in word-based indexing and retrieval systems by disambiguating search terms. Part-of-speech can also be used for stemming in information retrieval systems since knowing a word's POS can help to know which morphological affixes it can take. Part-of-speech tagging is the first full-fledged process which helps to develop parser so as to perform syntactical analysis. POS tagging is also primary step for semantic analysis in machine translation systems. It can also be a component for works related to spelling and grammar checking [3, 35].

Furthermore, it is used in parsing of texts to quickly find names, times, dates, or other named entities for the information extraction applications [29]. Finally, corpora that have been marked for parts-of-speech are very useful for linguistic research. In general, POS tagging is often considered as the first phase of a more complex natural language processing application.

1.7 Organization of the Thesis

The rest of the thesis covers a broad range of topics that are organized in six chapters. Chapter two presents literature review on part-of-speech, importance of POS tagging, and various approaches; like statistical, rule-based, artificial neural network and hybrid, with their advantages and limitations. Related works, specifically on Amharic language part-of-speech tagger, are also addressed in this chapter. This chapter gives more coverage for rule-based and artificial network which are the interest of this thesis work.

Amharic part-of-speech is an issue that is addressed in the third chapter of this thesis. This chapter sets up the classification mechanism of Amharic word categories which is important for building the POS system. Besides, chapter three includes a more detailed description of tag sets which are necessary and important for the design of the POS tagger. After the part-of-speech and tag sets for Amharic language are identified, the next step is to design Amharic POS tagger. Chapter four discusses in detail the design process, algorithm and architecture of the Amharic POS system. Design goals, approaches, and techniques are also included in this chapter.

Chapter five presents the data collection process and procedures of the experiments. Training and testing procedures are also given attention. The results section presents a summary of outputs from the various tests. Furthermore, the chapter deals with the implementation aspects and tools

that are utilized in the training and testing stages. Finally, conclusions and recommendations of this thesis work are presented in the sixth chapter.

Chapter Two

Literature review and Related works

During the last two decades, many Part-of-speech (POS) tagging researches have been performed with regard to their problem to a particular language. In this chapter, an attempt is made to discuss the widely utilized approaches in tackling part of speech problems in general and those that are used for Amharic language in particular.

*2.1 Approaches of Part-of-*Speech *tagging*

A number of approaches have been proposed for automatic tagging. The most common and widely used approaches are: Statistical, Rule-based and Artificial Neural Networks (ANN). There is also a hybrid of these approaches so as to make the tagger accurate by considering the good qualities of two or more approaches [46]. Several kinds of POS taggers using rule-based [11], statistical [40], memory-based [17], neural network [49], and hybrid [46] models have been proposed for different languages. These approaches consider linguistic methods which consist of coding the necessary knowledge in a set of rules written by linguists, frequency of the word in a given part of speech, or learning algorithms. These approaches are discussed below.

2.1.1 Statistical approach

Statistical approach is one of the widely applicable approaches in the POS tagging development environment [44]. It is a probabilistic approach that works based on the frequency information of words and tags in a given context. Based on frequency of words and tags, it calculates the lexical probabilities, i.e., the probabilities of a part-of-speech given the word, and contextual probabilities, i.e., the probabilities of a part-of-speech given the previous part-of-speech. These probabilities are trained on a manually tagged corpus. This method provides the capabilities of

resolving ambiguity on the basis of most likely interpretation. But, it requires statistical information to capture contextual information in order to exploit the power of probabilities and tag sequence of words. Several researchers used statistical methods to model relations between words, tags and context [40, 43].

Statistical approach tags words based solely on the probability that a word occurs within a particular tag. In other words, the tag encountered most frequently in the training set is the one assigned to an instance of that word. An alternative, to the word frequency approach, is to calculate the probability of a given sequence of tags occurring. This is sometimes referred to as the n-gram approach, referring to the fact that the best tag for a given word is determined by the probability that it occurs with the "**n**" previous tags. The next level of complexity that can be introduced into a stochastic tagger combines the previous two techniques, using both tag sequence probabilities and word frequency measurements [18, 33].

The main goal of the probabilistic methods, given a sequence of words $W_1...W_n$ is to find the sequence of tags $T_1...T_n$ that is the most likely tag sequence for these words, i.e. the sequence $T_1...T_n$ that maximizes $P(T_1...T_n | W_1...W_n)$. It is possible to estimate $P(T_1...T_n | W_1...W_n)$ using the probabilities $P(T_i | W_i)$ and $P(T_i | T_{i-1}T_{i-2})$ [33]. These two probabilities are called lexical and contextual probabilities, respectively. Lexical probability is the probability of observing a tag given a specific word $P(T_i | W_i)$. Contextual probability, on the other hand, is the probability of observing a specific tag (T_i) given the surrounding tags. They are typically implemented as bigrams or trigrams[1], i.e., $P(T_i | T_{i-1})$ or $P(T_i | T_{i-1}T_{i-2})$, respectively. The lexical and contextual probabilities, frequency-based information, are often estimated from a correctly annotated, pre tagged, training corpus [57].

[1] *An **n-gram** is a sub-sequence of n word from a given sequence; bigram and trigram are a special case of the n-gram, where N is 2 and 3 respectively [14].*

Stochastic (statistical) approach tries to define the class label of a word as a joint probability of the word and the classes in their contexts and choose the one that maximizes the probability. Mathematically this can be described as [19]:

$$t_{w_i} = t_j \quad where \quad j = \underset{1 \le k \le M}{\arg\max} \; P\{(w_1, t_1), \ldots (w_i, t_k), \ldots (w_n, t_r)\}$$

where M is the number of tags available.

A common formulation of a statistical POS tagger takes the form of a Hidden Markov Model (HMM), where the states correspond to POS tags, t_i, and words, w_i, are emitted each time a state is visited. The training of HMM–based taggers involves estimating lexical probabilities, $P(w_i|t_i)$, and tag sequence probabilities, $P(t_i \mid t_{i-1} \ldots t_{i-n})$. The ultimate goal of HMM training is to find the model that maximizes the probability of a given training text, which can be done easily using the forward-backward process. These probabilities are then used in conjunction with the Viterbi algorithm[2] to find the most probable sequence of POS tags for a given sentence. When estimating tag sequence probabilities, an HMM tagger, typically takes into account a history consisting of the previous two tags [14].

Applying HMM to tagging aims to find the most likely sequence of POS on the sentence level. It brings every word's probability and context of information together by finding the tag sequence that maximizes the likelihood of the product of word probability (p(word/tag)) and a tag sequence probability (p(tag/previous n tags) [16].

[2] *It is an algorithm for finding the most likely sequence of hidden states that result in a sequence of observed events, especially in the context of hidden Markov models [19].*

POS Tagging with Markov Models

The basis for all Markov Models for POS Tagging [14, 42]:

- Look for *t1 t2…tn* that maximizes P(t*1* t*2*. . . t*n*\w*1* w*2*. . . w*n*)
- Using Baye's Rule:
  - $P(t_1t_2 \ldots t_n/w_1w_2 \ldots w_n) = \frac{P(w_1 w_2 \ldots w_n/t_1 t_2 \ldots t_n) * P(t_1t_2 \ldots t_n)}{P(w_1w_2 \ldots w_n)}$
- Find t*1*t*2* . . . t*n* that maximizes the numerator
- Two main independence assumptions:
  - Words are independent of each other:

$$P(w_1w_2 \ldots w_k/t_1t_2 \ldots t_n) = P(w_1/t_1t_2 \ldots t_n) * P(w_2/t_1t_2 \ldots t_n) *\ldots* P(w_n/t_1t_2 \ldots t_n)$$

  - The probability of a word depends only on its tag:

$$P(w_i/t_1t_2 \ldots t_n) = P(w_i\backslash t_i)$$

A POS tagger based on Hidden Markov Model (HMM) assigns the best sequence of tags to an entire sentence. Generally, the most probable tag sequence is assigned to each sentence following the Viterbi algorithm. The task of POS tagging is to find the sequence of POS tags T= { t1, t2, t3,…..tn} that is optimal for a word sequence W= {w1,w2, w3…..wn}. The tagging problem becomes equivalent to searching for $\text{argmax}_T P(T)*P(W/t)$, by the application of Baye's law [40].

The probability of the tag i.e. P(T) can be calculated by Markov assumption which states that the probability of a tag is dependent only on a small, fixed number of previous tags. If it is n-gram the probability of a tag depends on n previous tags, then we have

$$P(T) = P(t_1)*P(t_2\backslash t_1)*p(t_3/t_1,t_2)*P(t_4/t_2,t_3)*, \ldots *P(t_n/t_{n-2},t_{n-1}) \quad [14]$$

To summarize this topic, two sources of information are required to decide a POS of a word concerning this approach [43]:

- ***Contextual Information***: Look at tags of other words in the context of the word of interest.

- ***Lexical Information***: Predicting a tag based on the word concerned. For words with a number of POS, one POS is assigned at a time in a given context.

In addition, these taggers usually take a large manually annotated corpus from which probabilities are extracted. In spite of the fact that statistical models are less accurate than rule-based models, most existing POS analyzers have been based on a probabilistic model because these systems can be automatically trained [38].

2.1.2 Rule-Based Approach

Rule-Based approach is one of the earliest approaches in developing POS tagger system to solve the problem of tagging. In contrast to the probabilistic method, the rule-based approach for disambiguation uses rules created by linguists or automatically by computer programs. The rules may contain a large number of morphological, lexical and/or syntactical information. They are based on linguistic knowledge about the structure of the language on specific languages. With human rule creation, there is a large set of manually constructed rules based on a specific grammar, written in a formal notation so that they can be used by the computer for further parsing. Many systems also use trial-and-error method, i.e. finding sentences where the rules have failed in order to manually add further rules to the system for higher accuracy [11, 39].

Adding a rule to the system may involve over-generation, i.e., one extra rule can result in more harm to the accuracy of general tagging in machine-learning rule-based approach. The manual rule-based tagging system has also some limitations: requires hand-written rules, costly and time-

consuming. Rule-based methods are time-consuming and require a great knowledge of a specific language being tagged in general.

The rule-based tagger has many advantages over stochastic; including [11]

- a vast reduction stored information required
- the perspicuity of a small set of meaningful rules
- ease of finding and implementing improvements to the tagger, and
- better portability from one tag set to another

Rule-based tagger is robust and the rules are automatically acquired. The tagger works by automatically recognizing and correcting its weakness, thereby incrementally improving its performance. The tagger initially tags by assigning each word to its most likely tag, estimated by examining a large tagged corpus, without regard to context [10, 57]. But this approach requires initially manually annotated corpus and large size data. And it generalizes rules to a small set of rules.

2.1.3 Artificial Neural Network (ANN)

Animals are able to react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours. An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems. Neurons work by processing information, i.e., receive and provide information in the form of spikes. An artificial neural network is composed of many artificial neurons that are linked together according to specific network architecture. The objective of the neural network is thus to transform the inputs into meaningful outputs [8, 27].

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems process information. It is composed of a large number of highly interconnected processing elements, neurons, working in unison to solve specific problems. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. Neural network is composed of a large number of highly interconnected processing elements, neurons, working synchronously or asynchronously to solve a specific problem. This is true for ANNs as well. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Neural Networks can also be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques [37,47].

Learning, in the context of ANNs , refers to adaptation of different behaviour on the basis of the data that is given to the network. Unlike telling the network how to react to each data vector separately, as would be the case in the conventional programming, the network itself is able to find properties from the presented data. Network learning will proceed as new data becomes available. Learning, in this case, is said to be adaptive. As data is given to the ANN, it organizes its structure to reflect the properties of the given data. The way the internal structure of an ANN is altered is determined by the implemented learning algorithm. Several distinct Neural Network models can be distinguished both from their internal architecture and from the learning algorithms they use. Basically, three entities characterize an ANN: the network topology or interconnection of neural 'units', the characteristics of individual units or artificial neurons, and the strategy for pattern learning or training. According to their connection geometries, Neural Networks can be classified as feed forward and feedback topologies [58].

2.1.3.1 Topologies of Neural Networks

2.1.3.1.1 Feed-forward networks

Feed-forward ANNs allow signals to travel one way only; from input to output. There is no feedback, i.e., the output of any layer does not affect that same layer. Feed-forward ANNs tend to be straight forward networks that associate inputs with outputs. For feed-forward networks, every neuron in a given layer receives inputs from layers below, that is, from layers nearer to the input layer, and sends output to layers above, that is, to layers nearer the output layer. For such networks, given a set of inputs, the input vector, from the neurons in the input layer, the output vector is computed by a succession of forward passes, which compute the intermediate output vectors of each layer in turn using the previously computed signal values in the earlier layers. Single-layer and multi-layer preceptor can be taken as a typical example of this topology. "Feed forward" means that the values only move from input through hidden to output layers; no values are fed back to earlier layers. A simple feed forward network can be seen in Figure 2.1 [59].

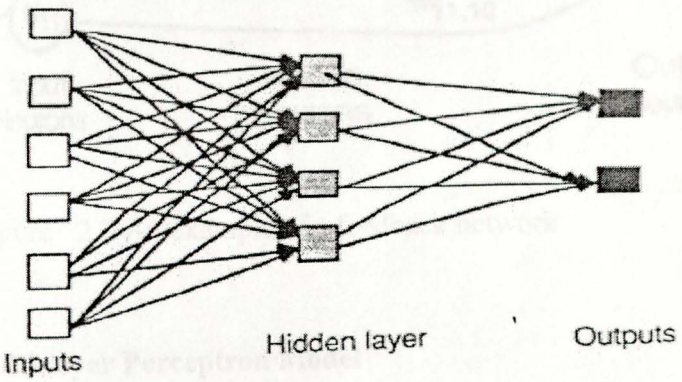


Figure 2.1 An example of a simple feed forward network

2.1.3.1.2 Feedback networks

Feedback networks can have signals traveling in both directions by introducing loops in the network. Feedback networks are very powerful and can get extremely complicated. Feedback networks are dynamic. Their 'state' is changing continuously until they reach an equilibrium

point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback architectures are also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations. Self-organizing map and associative network can be taken as examples of this type of network topology [49, 58], as shown in Figure 2.2.

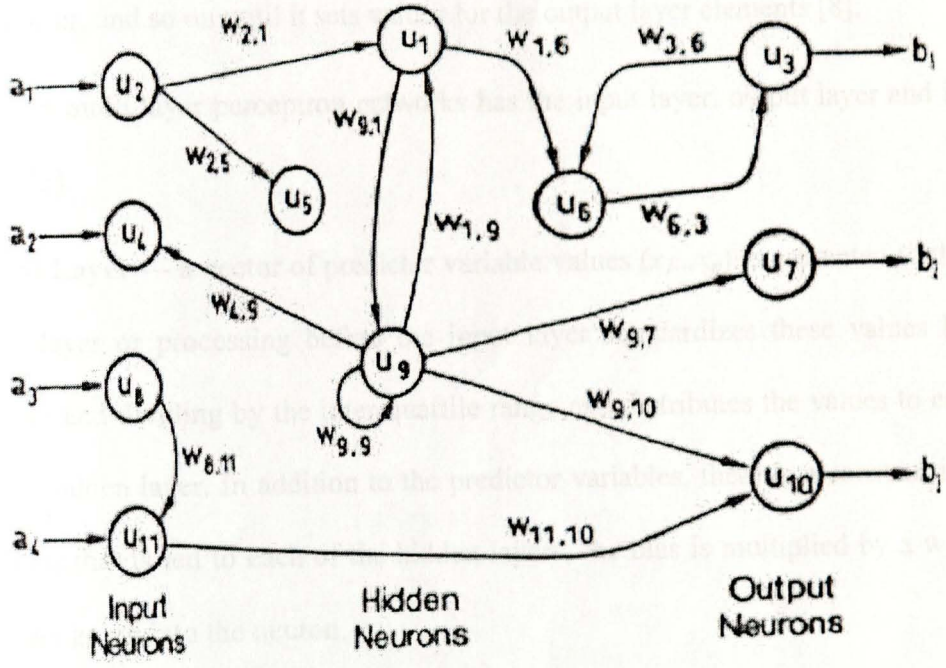


Figure 2.2 An example of a feedback network

2.1.3.2 Multi-layer Perceptron Model

It is one of the most widely used network models that links together processing units into a network made up of layers: input, output, and typically one or more hidden layers. A layer of "**input**" units is connected to a layer of "**hidden**" units, which is connected to a layer of **"output"** units. The activity of the input units represents the raw information that is fed to the network. The activity of each hidden unit is determined by the activities of the input units and the weights on

the connections between the hidden units and the previous unit. The behavior of the output units depends on the activity of the hidden units and the weights between the hidden and output units. The weights between the input and hidden units determine when each hidden unit is active, and so by modifying these weights, a hidden unit can choose what it represents. Each layer is fully connected to the succeeding layer. The network is feed forward. When using or testing a trained network, the input values set the values of elements in the first hidden layer, which influence the next layer, and so on until it sets values for the output layer elements [8].

In short multi layer perceptron networks has the input layer, output layer and in between hidden layer [5].

Input Layer — a vector of predictor variable values ($x_1 \ldots x_p$) is presented to the input layer. The input layer or processing before the input layer standardizes these values by subtracting the median and dividing by the inter-quartile range and distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the *bias* that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.

Hidden Layer — arriving at a neuron in the hidden layer, the value from each input neuron from previous layer is multiplied by a weight (w_{ji}), and the resulting weighted values are added together producing a combined value u_j, which is the activation value. The weighted sum (u_j) is fed into a transfer function, σ, which outputs a value h_j. The outputs from the hidden layer are distributed to the output layer.

Output Layer — Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight (w_{kj}), and the resulting weighted values are added together

producing a combined value v_j. The weighted sum (v_j) is fed into a transfer function, σ, which outputs a value y_k. The y values are the outputs of the network.

In MLPs, learning is supervised with separate training and recall phases. During training the nodes in the hidden layers organize themselves such that different nodes learn to recognize different features of the total input space. During the recalling phase, the network will respond to inputs that exhibit features similar to those learned during training. Incomplete or noisy inputs may be completely recovered by the network. In its learning phase, a training set of examples with known inputs and outputs will be given [27].

In a multi-layer perceptron topology, neurons are grouped into distinct layers as depicted in figure.2.3 [36]. Output of each layer is connected to input of nodes in the following layer. The first layer, input layer, passes inputs to the network while the outputs of the last layer, form the output of the network.

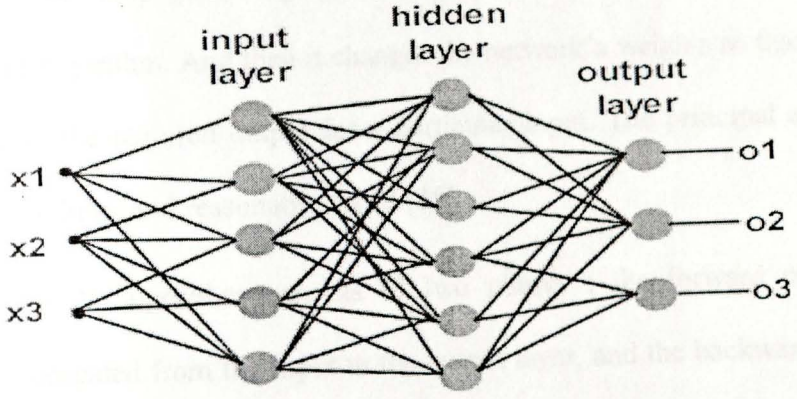


Figure 2. 3 Multilayer Perceptron Network

A multi-layer perceptron is especially useful for approximating a classification function that maps input vector ($x_1,x_2, \ldots x_n$) to one or more classes $C_1,C_2,\ldots C_m$. By optimizing weights and thresholds for all nodes, the network can represent a wide range of classification functions.

Optimizing the weights can be done by supervised learning, where the network learns from the large number of examples. Examples are usually provided one at a time. For each example the actual vector is computed and compared to the desired output. Then, weights and thresholds are adjusted, proportional to their contribution to the error made at the respective output. One of the most widely used algorithms is the Back-Propagation method, in which the errors are propagated in iterative manner. Error in this context is the difference between desired output and the output of actual ANN [27].

During training, error information is propagated back through the network and used to update connection weights. Different neural network architectures use different algorithms to calculate the weight changes. Back propagation (BP) is a commonly used algorithm in MLPs. BP solves the problem of how to calculate the hidden layer errors; it propagates the output errors back to the previous layer using the output element weights [13].

A Back Propagation algorithm updates weight from example. Examples of the target are given to the algorithm. And then it changes the network's weights so that when training is finished, it will give the required output for a particular input. The principal advantage of back-propagation is simplicity and reasonable speed [35].

The BP algorithm consists of two phases : the forward phase, where the activations are propagated from the input to the output layer, and the backward phase, where the error between the observed actual and the requested nominal value in the output layer is propagated backwards in order to modify the weights and bias values [15].

2.2.4 Hybrid Approach

Now a days POS taggers are designed by considering two or more approaches, which are discussed above. Basically, one approach is not full fledged by itself to achieve the best accuracy. In order to fill the gap and the shortcoming of the specific method, additional approaches are incorporated. Due to this better accuracies are achieved [54].

Ma *et al.* [46] has been used neuro and rule-based approach together. Neuron tagger and rule-based corrector are used to assign a part of speech for Thai language. Neuron tagger is used as initial state annotator and rule-based is done to correct the tags, which are outputs of the neuron tagger. Neuro tagger, especially elastic neuron tagger, uses different lengths of contexts on longest context priority in order to perform POS tagging. Three-layer perceptron with elastic input was selected so as to develop the model as it is shown in Table 2.1. The elasticity enables it to use variable length of context for tagging. Transformation rules are used to come out the rule-based corrector in order to increase the accuracy of the hybrid system. The neuro tagger is corrected by the rule-based corrector that will apply to minimize the errors done by the neuro tagger. The result of the experiment indicated that the hybrid system is best when it is compared to HMM, Rule-based, and Neuron as it is shown in Table 2.1. The errors made by the neuro tagger are corrected by the rule-based corrector. This contributed to have the hybrid model to achieve highest accuracy than others.

Table 2.1 Result of different approaches

Model	Baseline	HMM	Rule-based	Neuro	Hybrid
accuracy	83.6 %	89.1 %	93.5 %	94.4 %	95.5 %

2.2 Related works

The task of POS tagging is to assign unique POS tags to sentences that are presented as a linear string of words. POS tagging systems, which annotate corpora written in various languages, are used as components in many applications including phrase recognition, word sense disambiguate, grammatical function assignments and many others. Today, taggers of different kinds have been developed for different languages. Researches have been attempted on a very small corpus using stochastic and neural network approaches on Amharic language [42, 55].

A stochastic approach that considers Hidden Markov Model (HMM) is used to develop Amharic tagger [42]. In HMM approach, the most likely sequence of tags is considered as the most appropriate sequence of tags for the sequence of words in a given sentence. This sequence of tags is assigned to each word in the input sentence. The Markov assumption states that in calculating the probability of any sequence of categories, the probability of a category occurring in the sequence must depend only on the "n" categories where "n" is the number of categories before the category under consideration. The bigram model was considered for this thesis work. The Viterbi algorithm is used to generate the most likely sequence of tags for the sequence of words in a sentence. This algorithm depends on the assumption that the probability of a particular category occurring in a corpus depends solely on the immediate category preceding it. This algorithm helps to determine the optimal path of tags through the search space ranging from one unambiguous tag to the next. 25 tagsets were designed for this study. Manually tagged text was used to get the training and test set for the experiment. The training set was used to estimate the lexical and transitional probabilities required for the knowledge base. This set, the training set, consists of 90 percent, 210 words, of the total data. The test set used in the experiment consists of

10 percent of the total data to be tagged. Experiments were conducted in two phases, one on the training set and the other on the test set. The result obtained based on the small sample was high (97 %) on the training set and approximately 90% on the test set. Although the accuracy of the tagger developed is high, the tagger developed was not trained on such huge data.

TAGGIT was one of the first systems which used rules for disambiguate with an accuracy of 77% [25]. A rule-based POS tagger which automatically infers rules from a training corpus based on transformation-based error-driven learning is presented in [11]. It avoids most of the limitations of traditional rule-based taggers because it automatically infers rules from a training corpus and it utilizes some corpus statistics. It takes less space than probabilistic taggers. Hence, the system has become a competitive alternative to probabilistic tagging models [10, 21].

Eric Brill introduced a POS tagger in 1992 that was based on rules, transformations as he calls them, where the grammar is induced directly from the training corpus without human intervention or expert knowledge [11]. The only additional component necessary is a small, manually and correctly annotated corpus - the training corpus - which serves as input to the tagger. Lexical and contextual information is derived from the training corpus. Having such information enables the system to learn how to figure out the most likely POS tag for a word. After the training is completed, the tagger can be used to annotate new, unannotated corpora based on the tagset of the training corpus. This tagger does not use hand-crafted rules or pre-specified language information, nor does the tagger use external lexicons or lists of different types. According to this technique, there is a very small amount of general linguistic knowledge built into the system [10].

The general framework of Brill's corpus-based learning is called Transformation -based Error-driven Learning (TEL). The name reflects the fact that the tagger is based on transformations or

rules, and learns by detecting errors. The TEL begins with an unannotated text as input which passes through the initial state annotator which assigns tags to the input. The output of the initial state annotator is a temporary *corpus* which is then compared to a goal corpus that has been manually tagged. For each case, the temporary corpus is passed through the learner; the learner produces one new rule. This rule is applied and replaces the temporary corpus with the analysis that results when this rule is applied to it. By this process, the learner produces an ordered list of rules. The tagger uses TEL twice: first in the lexical module deriving rules for tagging unknown words, and second in the contextual module for deriving rules that improve the accuracy. Both modules use two types of corpora: the goal corpus, derived from a manually annotated corpus, and a temporary corpus whose tags are improved step by step to resemble the goal corpus more and more. In the lexical module of the tagger, the goal corpus is a list of words containing information about the frequencies of tags in a manually annotated corpus. A temporary corpus, on the other hand, is a list consisting of the same words as in the goal corpus, tagged in some fashion. In the contextual learning module the goal corpus is a manually annotated running text while a temporary corpus consists of the same running text as in the goal corpus but with different tags [38].

The rule has two parts: a condition and a resulting tag. The rules are instantiated from a set of predefined transformation templates. The ideal goal of the lexical module is to find rules that can produce the most likely tag for any word in the given language, i.e. the most frequent tag for the word in question considering all texts in that language. The problem is to determine the most likely tags for unknown words, given the most likely tag for each word in a comparatively small set of words [10, 11].

Once the tagger has learned the most likely tag for each word found in the manually annotated training corpus and the method for predicting the most likely tag for unknown words, contextual rules are learned for disambiguate. The learner discovers rules on the basis of the particular environments, or the context, of word tokens.

The contextual learning process needs an initially annotated text. The input to the initial state annotator is an untagged corpus. The initial state annotator needs a list, thus so called training lexicon. It consists of a list of words with a number of tags attached to each word. These tags are the tags that are found in the first half of the manually annotated corpus. The first tag is the most likely tag for the word in question and the rest of the tags are in no particular order. The initial state annotator assigns to every word being in the untagged corpus the most likely tag. It tags the known words, i.e. the words occurring in training lexicon, with the most frequent tag for the word in question. The tags for the unknown words are computed using the lexical rules. The annotated text obtained is the initial temporary running text that serves as input to the contextual learner [10, 11, 38].

According to [55], Artificial Neural Network approach that employs a supervised learning mechanism was selected to build the Neural Network model for part of speech tagger. It is three layer MLP network. Multi layer feed-forward network with back propagation algorithm was designed because of its capabilities of expressing non-linear decision. This work considered neighborhood context, localized information, for tagging of words in a small tagged corpus. The input to the network is the set of words that fall into a window of pre-specified size centered on the target word to be tagged. The output of the network is the corresponding tag for the target word. The study used 24 tagsets which were developed in [42]. Manually tagged corpus consisting of 159 sentences was prepared, 136 sentences, 2429 words, for training and the other

23 sentences, 392 words, for testing. The result of the MLP-part of speech tagger was an average accuracy of 93 % on words in test data that are not seen during the training process. The accuracy may increase if it were tested with a large data.

2.3 Summary

It has recently become clear that automatically extracting linguistic information from a sample text corpus can be an extremely powerful method of overcoming the linguistic knowledge acquisition bottleneck inhibiting the creation of robust and accurate natural language processing system. POS tagging is one of the tasks performed on natural languages, which assigns the correct POS to each word in the context of the sentence. Several kinds of POS taggers using rule-based, statistical, neural-network have been proposed for some languages.

The statistical approach requires considerable training samples to estimate the probabilities of word sequence and considerable memory capacity to process these probabilities. Additionally, it is difficult to predict unseen data, which never appeared in the training data. Rule –based approach is another alternative to assign tags to words by set linguistic rules or automatic generation of rules through machine learning process. Neural networks are also interesting mechanisms which can learn general characteristics or patterns from limited sample data. It is also exercised to pass through two approaches in order to achieve better accuracy. The next chapter is describing the Amharic word categories which are the bases for the building of automatic POS tagger.

Chapter Three

Amharic POS and Tagsets

3.1 Amharic POS

3.1.1 Overview

Amharic served as a government language more than a century. It has also relatively been studied than other Ethiopic languages. But there is no clear cut on Amharic part of speech categorization. One of the most cited classical work on Amharic grammar, written in Amharic, put the word classes of this language as eight [41]. These are preposition, noun, conjunction, interjection, verb, adjective, pronoun and adverb. Recent works in [7, 2], have stated that Amharic has only five word classes. They did by leaving out interjection from the inventory and putting together prepositions and conjunctions in one class and considering pronouns as a sub-class of nouns. Baye's [7] reduction of Mersehazen's [41] classification seems based on the role of words in syntax, i.e., considering words that have clear role in Amharic sentence grammar.

According to a study in [23], Amharic has only four basic word classes. These are nominals, verbs, adpositions-conjunctions and interjections. As it is the case in any language, the above four classes can be divided into sub-classes. The subclasses in turn can be divided into mini-sub-classes and subdivision process may be continuing iteratively depending on the level and aim of the investigation.

3.1.2 The Need for Categorizing Words

Words are basic components of every sentence. The meaning of a sentence is analyzed from the meaning of individual words and the way they are arranged. This shows words are rarely used alone. Words more often work together in small groups which together make up whole sentence.

These small groups possess a single coherent meaning. In addition, the same word can be used in different sentences and belong to a different word class in each sentence.

Three basic criteria are considered in order to categorize words in a language. They are: the meaning of the word, the form or shape of the word, and the position or the environment of the word in a sentence. These can be taken as the main criteria to determine the categories of a given word [26].

3.1.3 Baye's Classification of Amharic Words

Baye [7] shows the categorization of Amharic words reduced to five. These are: preposition, noun, verb, adjective, and adverb. Pronouns and conjunctions are put under noun and preposition categories respectively. Interjections are words without syntactic functions. In this categorization, interjections are not considered as part-of-speech.

3.1.3.1 The Amharic Noun Class

A noun is a word which is used to label things, such as a real thing (for example, cat), an imaginary thing (for example, ghost), an idea (for example, love), person name (for example, 'Abebe'). Amharic nouns, like English, are words used to name or identify a class of things, people, places or ideas.

3.1.3.2 The Amharic Verb Class

Verbs are words that tell us the state of doing or being. Verbs are morphologically the most complex POS in Amharic, with many inflectional forms; numerous words with other POS are derived primarily from verbs. There are two major criteria to identify verbs from other word categories. These are: syntactically and morphologically. In the former case, verbs function as predicates in a simple sentence and they are found at the end of a sentence. In the later case, they reflect grammatical categories such as aspect, mood and agreement.

3.1.3.3 The Adjective Class

Adjectives act as describing words or phrases that add extra information to a noun. Adjectives in Amharic usually precede the nouns that they modify or describe. Let's look at the following example.

Example: ጎበዝ ተማሪ ነው። (He is a clever student). In this example, the adjective "ጎበዝ" (clever) precedes the noun "ተማሪ" (student) which it modifies. But this does not mean that a word is an adjective just because it precedes a noun. For instance, in the sentence "ይኼ ተማሪ ነው።" (this is a student), the word "ይኼ" (this) precedes the noun "ተማሪ" (student). Although the word "ይኼ" functionally shares the feature of an adjective, modifier, it is a demonstrative pronoun.

3.1.3.4 Prepositions in Amharic

Prepositions are small words which will have meanings only when they are attached or used together with other words such as nouns, verbs, pronouns and adjectives.

Prepositions can appear as:

- A simple prepositions that stand alone as separate words

  Example: ስለ ትምህርት (for the sake of education)

  ወደ ቤት (to the house or home)

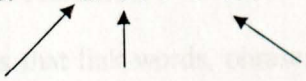
- A simple preposition prefixed or attached with other words (e.g. nouns and verbs).

  Example: በ-መኪና (by car)

  ለ-ተማሪ (to/for the student)

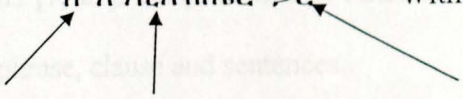
- As compound prepositions having two parts, prepositional prefixes and post positions. The postpositions can either be single preposition that stand by their own or a preposition not separated from a noun.

Example: በ ሳጥኑ ውስጥ "inside the box"



Preposition box (noun) a preposition (noun)-inside

ከ እግዚአብሔር ጋር "With God"



Preposition noun (God) preposition noun (with)

3.1.3.5 The Adverb Class in Amharic

An adverb is another word class that acts as a modifier. Adverbs function to add more information to verbs, other adverbs, adjectives, and even to whole sentence. In Amharic, adverbs can be found in either primitive form, i.e., as separate words or in compound forms as combinations of prepositions and some other words. In both cases, they refer to place, time, circumstance, etc. There is also what is called adverbs of position, which may be referred to as noun adverbs. Noun adverbs can be used either as a noun or as an adverb depending on their context.

Examples: ወደ ውስጥ ገባ (he went to inside). In this case, " ውስጥ" is used as a noun.

ቤት ውስጥ ይሰራል (he works in the house). In this case, ውስጥ is used as an adverb.

As it can be seen from the above examples, noun adverbs or adverbs of position such as "ውስጥ" (inside) can function alone as an adverb or as a noun. If noun adverbs are used alone without prepositions, like "በ", "ወደ", "ከ", they are mostly treated as nouns. On the other hand, noun adverbs prefixed with prepositions are treated as adverbs. That is, "ውስጥ" is a noun while "በ -ውስጥ" is an adverb. Adverbs can also be formed from nouns and adjectives by prefixing "በ" as in "በ -ኃይል" (by force) and "በ ደህና" (being well) respectively.

3.1.3.6 Conjunction in Amharic

Conjunctions are words that link words, phrases and clauses to create larger grammatical units. Conjunctions were treated as a separate category [41]. However, conjunctions are treated also as a subclass of prepositions [7, 21]. Conjunctions in Amharic are coordinating or subordinating. They coordinate words, phrase, clause and sentences.

3.1.3.7 Interjections

Interjections are words that function to suggest sudden, often unexpected, emotion. Like English, Amharic has many words or phrases used to express such emotions as sudden surprise, pleasure, annoyance and so on. Such Amharic words are called interjections. These Amharic interjections can stand-alone by themselves outside a sentence or can appear anywhere in a sentence.

Example: ጎሽ !

ጎሽ ! በጣም ጥሩ ውጤት ነው

3.1.3.8 Numerals

These are words representing numbers. In Amharic, the ordinal numbers are formed from the cardinal numbers by adding the suffix "ኛ".

Example:	cardinal	ordinal
	አንድ (one)	አንደኛ (first)
	ሁለት (two)	ሁለተኛ (second)

Like English, compound Amharic numerals are put separately. The following are examples to illustrate this.

Example: አንድ መቶ አንድ (one hundred one)

አንድ መቶ አንደኛ (101^{th})

In Amharic, there are also numerals that indicate distribution. These numerals are called distributive numerals. Example: ሶስት ሶስት (three three).

There are also special numerals in Amharic that correspond to the English "half", "quarter", etc. Example of these include ግማሽ (half) and ሲሶ (one third).

3.1.4 New approach to categorize Amharic word class

As it is highlighted in the introduction part of this chapter, Amharic word class is categorized in to four classes according to [23]. These are nominal, adposition-conjunctions, prepositions and interjections. In syntax, since pronouns act like nouns, categorizing pronouns under the class of nouns seems appropriate. Moreover, since most prepositions in Amharic function also as conjunctions, putting these two into one word class again seems logical. Interjections are words that function beyond syntax. Leaving out this word class from the inventory of word classes, therefore, can also be justified if the focus is only on syntactic words. However, even from syntactic point of view, Baye's [23], classification cannot be considered as a refined and complete. For instance, adjectives in Amharic, in fact, in other Semitic languages too, can be categorized as a sub-class of nouns. Adverbs too can be categorized as a sub-class of nouns, as is also the case in the traditional treatment of these word classes in Arabic [53].

3.1.4.1 Nominals

Pronouns, adjectives, adverbs, verbal nouns, i.e. infinitives, proper nouns, common nouns, etc. are categorized under nominals. The sub-categorization of adjectives and adverbs under the class of nouns is based on the facts stated below.

Adjectives and Adverbs as sub-class of nominals

In Amharic, adjectives function as subjects and objects of a clause besides inflect for grammatical categories that are associated with nouns such as gender, number, and case. Adverbs also can be categorized as a sub-class of nouns for two major reasons. First, like other Semitic languages, this class of words in Amharic, are not many in number. Second, those limited words may function

also in syntax as subjects and as objects in clause, i.e. being a lexical head of such phrases like other nominals do.

It is important not totally to abandon the categories of adjectives and adverbs in Amharic. This is because these classes of words have distinct properties that differentiate them from other subclasses of nominals. For instance, while adjectives can modify nouns without any additional element, i.e. prepositions, other sub-classes of nominals do not. Similarly, adverbs can modify verbs without a preposition while other class of nominal cannot without a prepositional element or without being marked for case.

3.4.1.2 Types of Adverbs

Under the category of adverbs, words, such as "ፀጥ ", are included which do not give sense when they stand alone. This type of words in Amharic can be used as a base for the derivation of other words by taking derivational morphemes as in "ፀጥታ" ('being quiet'), "ዝምታ" (being silent). Due to this, these words are categorized as a sub-class of adverbs even if they require another word to give full sense. Their composition is not always observed in the lexicon. This is especially true in their appearance with the semantically dummy verbs "አለ" (he said) and "አደረገ" (he does), as can be inferred from constructions such as "ብድግ እንደ-አለ" (as he was/is standing) where the complementizer "እንደ" is attached with the dummy verb "አለ" as is also the case with other verbs. In this type of construction, although the word "ብድግ" must go with "አለ" or "አደረገ" but not with other verbs, the form "ብድግ ... አለ" cannot be considered as a compound word which is formed in the lexicon because these words do not behave as a single (compound) word in syntax, as is evident with the insertion of "እንደ" separating the two which is purely a syntactic operation. Hence, "ብድግ" in this type of construction, must be a phrase on

its own. As mentioned above, words that modify a verb without a preposition are only adverbs. Hence, considering words like "በድግ" as an adverb seems logical.

Beside the above type of adverbs, we may consider words such as "ዛሬ" (today), "አምና " (last year), "ትናንት" (yesterday), "አሁን" (now), etc. which mostly mark time and realized unlike the above type of adverbs on their own, i.e. being a prosodically complete grammatical word.

The third types of adverbs are words that mark degree. These are not many in number in the language. It includes words such as" እጅግ" (extremely), "በጣም" (very). Unlike the above two types, these adverbs do not have lexical content on their own right. This makes them to be different not only from other types of adverbs but also with any other nominals. They can also modify verbs by their own.

3.1.4.2 Verbs

In Amharic, words that are used as a predicate in a clause and inflect for person, gender and number in agreement with an NP subject being marked also for aspect mood or sense can be categorized as verbs.

3.1.4.3 Adpositions- conjunctions

Words do not have lexical content and attached with noun phrase, words that form prepositional phrase, and words that make a clause subordinate to another can be considered as prepositions. These include words such as "እንደ", "ስለ", "እና ", "ወደ". Words such as "ወደ" and "ስለ" are only used as adpositions and words such as "እና " and "ግን " are used only as conjunctions. Most other words in this category, however, can be used as conjunctions and adpositions.

3.1.4.4 Interjections

These are words, such as"አይ" (no),"አዎ "(yes), and "ኧረ " (oh), that express emotions and are not bound within sentential syntax.

3.2 Amharic Tagset

Based on the discussion in the first section of this chapter, it is appropriate to come up with the tagsets for automatic assignment of POS for Amharic texts. Three main points are considered to decide the tagsets for this thesis work. First and most is the Amharic word categories and their classification by different Ethiopian linguists [7, 22 23, 41]. Previous attempts on setting the tags are also other determinant factors. Mesfin [42] has come up with 23 tags plus two for all punctuation marks and unknown words. Other researchers in [51, 55] also adopted tagsets which were used in [42].

The most recent work in [24] shows 30 tagsets. The main function of this tagset was for the annotation of 10, 000 news items which are acquired from Walta Information Center. For this work, these tags are adopted because of the following reason. These tagsets are done for the Ethiopian Language Research Center so that it was done by linguists and reviewed by scholars on the domain. In addition, the training and testing data are annotated with these tagsets. Thus, the discussion in this part is according to these 30 tagsets.

The tagsets are classified as basic class and subclasses under the former one which depends on the word categories for Amharic context. Noun, pronoun, verb, adjective, preposition, conjunction, adverb, interjection are the basic classes. In addition, numeral, punctuation, and unclassified or unknown word tagsets are also incorporated with the above main classes.

Nouns

Nouns can be proper nouns or pronouns. They have attributes like gender, number, case, and definiteness. But for this study all these attributes are not taken into considerations. Verbal noun,

verbal noun with propositions, verbal noun attached with conjunctions, noun including verbal noun with preposition and conjunction are sub classes under this noun.

- In Amharic language nouns might come with preposition and conjunctions that cannot be separated from the nouns. Nouns affixed with prepositions are considered as noun preposition and tagged as **NP**. Example: **በተያዘው የበጀት ዓመት ለአገልግሎት ይበቃሉ ተብለው እንደሚጠበቁ አስታውቀዋል።**
- Likewise, nouns may be attached with conjunction and when the conjunction can't be separated from the noun, it is considered as noun conjunction. NC is the tag for such occurrence. Example: **የአሸባሪዎችን ድርጊት የኢትዮጵያ መንግስትና ሕዝብ እጅግ እንዳወገዘው ገልፀዋል።**
- Some nouns can also be formed from any verb forms like active, passive, by attaching prefix and tagged as **VN**. Example: **ከተጀመሩ 12 ፕሮጀክቶች መካከል አብዛኞቹ ተጠናቀው ሃምሳ ነዋሪዎችን ተጠቃሚ ማድረጋቸው ተገለጸ ።**
- Nouns are coming with preposition and conjunction as we saw it above but when the preposition and conjunction is procliti, is grouped as noun preposition conjunction and tagged as **NPC**. Example: **ችግሮችን ለይተው የሚያካሂዱት በመሆኑ ከምንጊዜውም በበለጠ ተቋማዊ ውሳኔዎች ይተላለፋሉ ብለው እንደሚጠበቁ አስተያየት ሰጪዎች ተናግረዋል።**
- Proper nouns, like personal names, geographic names, organization names, Amharic initials such as Ato, W/o, W/t, days of the week, months of the year, directions, compound words, either multi words or fused words, are simply considered as noun and tagged by **N**. N is the general tag for nouns if a word can not be categorized under sub-

categories. Example: ኢትዮጵያ የተያያዘቻቸውን ዘርፈ ብዙ የልማት እንቅስቃሴዎች .....::

Pronouns

Pronouns are special types of nouns. They occur frequently in any document. So that they are treated independently. There are three sub categories and one general tags.

These are:

- Pronouns which come with prepositions (**PRONP**)

  Example: ለምርምር እየተባለ በዘፈቀደ ስለሚወጡበት አሰራር እንዲሁም ለአክሱም ሃውልት.....

- Pronouns attached with conjunctions (**PRONC**)

  Example: ለፈጠራ ሥራዎች መበራከት አስተዋፅኦ ማበርከታቸውን አስረድተዋል:: ይሁንና አገሪቱ ካላት የቴክኖሎጂ ፍላጎት አንጻር አብዛኞቹ የፈጠራ ሥራዎች.....

- Pronouns with a proclitic preposition and enclitic conjunction (**PRONPC**)

  Example: ሆኖም በሁሉም ክልሎች በቅርቡ በተደረገ ቆጠራ ...

- **PRON** is assigned for the rest of the pronouns which is the general tag

  Example: በተለይም አንዳንድ የሥራ ኃላፊዎች መረጃ አንሰጥም ሲሉ የሚወሰድባቸውን እርምጃ ...

Verbs

Verbs are the main components of any text including Amharic. These verbs indicate the action that is done by the subject. We can say that without verbs, there exists no sentence. Due to high occurrence of verbs, one general tagset and five subcategories are assigned to tag verbs. The sub

categories indicate the role of the verb and its appearance with other part-of-speech in conjunction.

- Auxillary verbs are one sub categories which are tagged by **AUX**. AUX is used to show such verbs in all forms without any distinction like number, gender, tense etc.

  Example: **የአገሪቱ መሬት ለወባ ትንኝ መራቢያ አመቺ ነው ::**

- Verbs which are attached with conjunction are tagged as **VC** for any type of verb, auxiliary or relative.

  Example: **የሚመለከታቸው አካላት ለጎብኝዎች አላስተዋወቁትም ::**

- VP is used to tag verbs which are suffixed by prepositions. Example: **....ከሚደረገው እንቅስቃሴ በተጨማሪ የተጠናከረ ሕክምና ለመስጠት ከወረዳ እስከ ቀበሌ ደረጃ ላሉ የጤና ተጠሪዎች ሥልጠና ይሰጣል ብለዋል::**

- Relative verbs are tagged by **VREL**

  Example: **የተጀመረው የመከላከል እንቅስቃሴ የተጠቂዎችን ቁጥር ለመቀነስ ያስችላል ብለዋል::**

- Any verb including relative and auxiliary might come with proclitic preposition and an enclitic conjunction. In such case, it is tagged as VPC.

  Example: **ስለቀደምት አባቶች ታሪክ በተስተካከለና ስርዓት ባለው መልኩ ግንዛቤውን ለማዳበር ....**

- All other verbs which can not belong to the above catagories, it is simply tagged as **V**.

  Example: **... የአዲስ አበባ ሙዚየም በርካታ ቅርሶችን በመሰብሰብ ላይ እንደሚገኝ አመልክተዋል::**

Adjective

Adjective is also another word category in Amharic language. In order to tag such word, one general tag and three sub categories are designed. The general tag is used to tag adjectives without considering attributes like gender, number, and definiteness.

- **ADJP** is the tag to assign adjectives prefixed by prepositions. Example: **ኢትዮጵያ የተለያዩ ሃይማኖት ተከታዮች ለብዙ ዓመታት ተጋግዘው የሚኖሩባት ሀገር በመሆኑዋ ...**
- Adjectives also suffixed by conjunction and **ADJC** is used to tag such adjectives. Example: ... **በዚሁም መንግስታዊና መንግስታዊ ያልሆኑ ድርጅቶች**
- Adjectives with a proclitic preposition and an enclitic conjunction assigned **ADJPC**. Example: ... **የአነስተኛና መካከለኛ ኢንዱስትሪዎች እንዲጠናቀቁ**
- All adjectives which cannot fall in either of those sub categories are tagged as **ADJ**. Example: ... **ይህም በሀገሪቱ የእንስሳት ሃብት ላይ ከፍተኛ ጉዳት ስለሚያደርስ ....**

Prepositions

It is very difficult to tag prepositions because prepositions are not separated from nouns, verbs and adjectives [43]. But **PREP** is assigned to tag prepositions as separate words.

Example: **በደረጃ ዕድገት አፈጣጠር ላይ ችግር በመፍጠሩ**

Adverbs

ADV is used to assign all adverbs. If compound adverbs appear as multi-words, the tag ADV is assigned once at the end of the compound adverbs. ADV is applied more widely, and as such it does not differentiate among adverbs of time, place, manner and adverb.

Example: **ትናንት በጎንደር ከተማ በተዘጋጀ የአንድ ቀን ትውውቅ መድረክ ...**

Conjunction

Like prepositions, tagging conjunctions is also another challenging task in Amharic language. This is because most of the conjunctions are also prepositions and they are attached with other words. So that conjunction appeared separately as a single word will be assigned the tag **CONJ**. Multi word and compound conjunctions are also assigned the same tag.

Example: ሁለት ጊዜ <u>ወይስ</u> ሦስት ጊዜ ይጠራ በሚለው ሃሳብ.....

Numerals

Amharic numerals can be expressed as ordinal or cardinal. In such case NUMOR is assigned to ordinals and cardinals like one, two are assigned the tag NUMCR. Numerals may be attached with prepositions, conjunctions, proclitic preposition and enclitic conjunction, in these cases, NUMP, NUMC, and NUMPC are assigned respectively.

Example:

- በዛሬው እለት በባንኮች መካከል በተካሄደው የውጭ ምንዛሬ <u>አንድ</u> የአሜሪካን ዶላር ...
- <u>11ኛው</u> ሀገርአቀፍ የትምህረት ጉባኤ ዛሬ ...
- <u>ለአምስት</u> ቀናት በሚቆየው በዚሁ ጉባኤ ከሁሉም የሀገሪቱ ክልሎች የተውጣጡ ...
- ከሰላሳ በላይ የምርምር ቡድኖች ቢኖሩም ቅሬታ የሚያቀርቡት <u>አንድና</u> ሁለት ብቻ መሆናቸውን ጠቁመዋል ::
- ... <u>የሁለቱም</u> ወረዳ ነዋሪዎች የንጹህ መጠጥ ውሃ ተጠቃሚ ሆነዋል::

Interjection

Interjections, words or phrases used to express emotions, can stand by themselves or can appear anywhere in a sentence. INT is the tag assigned to interjection.

Example: ... እጅግ በጣም አነስተኛ ቁጥር ያላቸው ጎዳና ተዳዳሪዎች ...

Punctuation Marks

To all Amharic punctuation marks like :-, :, ::, ?, ! are assigned the tag PUNC.

Example: ... በየዓመቱ ወደ ጎዳና ከሚወጡት አንጻር ሲታይ ግን ብዙ ይቀይረዋል ብለዋል::

Unknown words (Unclassified words)

For words not found in the lexicon of the tagger, **UNC** is assigned because there might be unrecognized words.

3.3 Summary

Amharic part–of-speech can be categorized into eight, five or four classes according to the thoughts of the early and the recent studies. From this discussion, we can conclude that, though, Amharic is used as the language in most areas in Ethiopia, still there is no consensus on the number of classes of this language. The purpose of the classification in each study has not also been clearly defined. The classification type can be vary according to the objective set by the researchers. For this thesis work, all essential word categories are tried to consider for the tagging process. These are very important to set the tags for Amharic language.

Without understanding Amharic grammatical structure, it is very difficult to design a system that might serve the user of the language. Amharic tagsets, which are essential in tagging the words, are extracted from the Amharic POS. This indicates that to propose a better tagset, knowledge of the language is the key factor. In this thesis work, it is tried to use 30 tagsets that serve the

Amharic language in tagging process with minimal modification. The tagsets are summarized in tabular form below.

Table 3.1 Summery of the Tagsets

No.	Basic Class	Code of the tag
1.	*Noun*	VN
		NP
		NC
		NPC
		N
2.	*Pronoun*	PRONP
		PRONC
		PROUNPC
		PRON
3.	*Verb*	AUX
		VREL
		VP
		VC
		VPC
		V
4.	*Adjective*	ADJP
		ADJC
		ADJ
5.	*Preposition*	PREP
6.	*Conjunction*	CONJ
7.	*Adverb*	ADV
8.	*Numerals*	NUMER
		NUMOR
		NUMP
		NUMC
		NUMPC
9.	*Interjection*	INT
10.	*Punctuation*	PUNC
11.	*Unclassified*	UNC

Chapter Four

Design of Amharic POS Tagger

A POS tagger is a program that assigns part of speech to Amharic words based on the context in which they appear. As it is described in chapter 3, hybrid tagger is developed that consists of neural network and rule-based. In this chapter, the design of the hybrid tagger as well as approaches and design goals are discussed.

4.1 Design approaches and techniques

Part of speech tagging is the process of marking up the words in a text as corresponding to a particular part of speech, based on its definition as well as its context i.e. relationship with adjacent and related words in a phrase, sentence, or paragraph. Part-of-speech tagging is not a mere list of words and their parts of speech, since some words can represent more than one part of speech at different times. There are many approaches and techniques to tackle this problem such as rule-based, statistical, and neural networks. These approaches help us to device a POS tagger for a specific language like Amharic.

Rule-based approach is the earliest from others in tagging words. This approach can be classified in to two: linguistic and machine learning. Linguistic approach consists of coding the necessary knowledge in a set of rules written by linguists. This approach is more or less manual, time taking, tiresome and requires expertise in the domain. In addition to these facts, it is inconsistent and subjective since it is determined by the understanding of specialist and their linguistics background of the specific language. The approach proposed in [11], transformation-based learning, can be categorized under this group as machine learning process. It learns the series of

transformations (rules), exercising these transformations, and then it generates rules that help to tag words to their most appropriate category [10].

Statistical approach is the most widely applicable and successful model during the last few decades in POS tagging [14, 20, 42]. In comparison to the linguistic approach, it requires much less human effort. Hidden Markov Model and related techniques have focused on building probabilistic models of tag transition sequences in a sentence. This approach has a shortcoming in that it assigns a tag by considering only the frequency.

The third family, Artificial Neural Networks, use learning algorithm that acquires a language model from training corpus. It is the recent approach that attracts attention in solving many computational linguistics problems [36]. It has been applied successfully for the assignment of POS tags due to its advantages over others, as it is discussed in chapter 2.

POS tagging systems aim at classifying input sequences of lexicons by assigning each sequence a corresponding sequence of most probable POS tags. It is often assumed that for each input lexicon there is a set of priori possible POS tag categories, or a probability function over them, and the tagger has to choose from this limited set of candidate categories. In recent work [20], it is indicated that two ways can be used to obtain lexicon: data-driven and lexicon based approach. Data-driven approaches employ the lexicon only implicitly when extracting features on possible POS tags from annotated corpora that are used for training the POS tagger. Lexicon based approach uses the lexicon that is extracted from a manually constructed morphology.

Techniques of tagging can also be categorized as supervised and unsupervised [30, 37]. Supervised technique usually relies on pre-tagged corpora to automate the tagging process. From these pretagged corpora, it requires linguistic information in the form of probabilities so as to perform the training. Such type of tagging technique requires probabilities acquired in the

training phase to assign tags to the words. Training and tests are performed based on tagged corpus.

Taggers that do not require pre-tagged corpus during the learning process are called unsupervised. Unsupervised techniques do not require pre-tagged corpus but use sophisticated computational techniques. Induction of dictionary and tagset are possible in this technique without having tagged training data. As a result, tagging of test data can be performed using induced dictionaries. It is characterized by randomly generating tags and usually has low accuracy [56].

These approaches and techniques have direct implication and reflection on the design process in particular and the development of the tagger system in general [34]. As a result, the design of this thesis work is dependent on mixed approach that comprises machine learning rule-based and neural network. Artificial Neural Network approach enables the system to learn the pattern through the training period. Based on pattern information, the system can classify words to their appropriate category unlike the statistical approach. But it is very difficult to train the Neural Network up to 100% [46, 49]. To fill this gap, rule-based approach is introduced. This is to maximize the expected output by filling the gap of one approach by the second. Because each approach tackles the POS tagging differently, it is advantageous in minimizing their limitation.

4.2 Design Goals

In general, prior to designing a POS tagger, it is recommended to set the design issues and goals for evaluating the system. This equally works for Amharic POS tagger. Though many goals can be set in developing POS tagger, only few of them are considered. The first goal of designing the tagger having two approaches is to achieve better accuracy than tagger which considers only one approach. Accuracy refers to the closeness of agreement between a test result and the accepted

reference value. The other issue is adaptability which is open to learn. From the perspective of twisting Neural Network with rule-based approach, it makes the work better because both approaches are open to learn from the given training data set and adjust themselves from time to time by correcting errors. This is also highly significant in maximizing the accuracy of the hybrid tagger. Speed is also another crucial issue in designing taggers from the user point of view. Brill E. [10] stated, taggers which are developed using rule-based approach are ten times faster than the fastest stochastic approach. In addition to that, taggers developed using neural network and rule-based approach requires less storage than taggers developed using statistical approach.

The main goal of designing hybrid POS tagger (rule-based and neural network) is to achieve better performance in tagging Amharic text. In addition to this, the tagger is expected to be easy to implement and increase speed in responding a tagged text, and easy in obtaining the required knowledge that can be estimated using large tagged corpora [28, 33].

4.3 Designing Neural Network Model

As it is explained above, the tagging process of Amharic text has gone through the neural network first. After a word is tagged by the neural tagger, it has also passed through rule-based tagger for further correction. This is to enhance the accuracy of assigning the right tagger that indicates the role of the word in the given context. The design issues are the major concern of the tagger developer. The process of designing Neural Network model is an iterative course of action that achieves an optimal result by adjusting the parameters of the network. In general, designing an ANN model consists of the following decisions and activities:

- Deciding network topology, which involves deciding the number of layers in the network, the number of neurons in each layer, and the type of connection and communication among neurons for different layers, as well as among the neurons within the same layer.
- Deciding the learning algorithm, which is a prescribed set of well defined rules for the solution of learning problem
- Encoding schema, which refers to deciding the way a neuron receives input and produces output: how the input and output data are represented or encoded, is a major component to successfully instructing a network.

In this section, the design of the neural tagger and the selected algorithm are addressed.

4.3.1 Feature Extraction

Feature extraction is the process of mapping the original features into fewer features that include the main information of a given word. With the advent of neural networks, more and more problems are solved by simply feeding large amounts of 'raw data' to a neural network [52, 55]. During training, the network learns what value to place on what feature. The features are extracted from the tagged corpus and pre defined tagsets. Mainly, the features are extracted from the following information.

- ***POS information***: the candidate POS tags of the current word and word sequence which is centered on target word and consider the right and left word. The input length has elastic nature to be extended from a small to a large size. The candidate POS are determined in advance for each word using the Amharic corpus.
- ***POS and order information***: The pair of candidate POS tags and their occurrence order in the current and adjacent words are taken in elastic manner starting from the most nearby

word in both directions. The occurrence order indicates the frequency order of the POS in the training data when it is used for the current word.

- ***Word information***: besides the POS tags of the current and surrounding words, the word by itself is also important information. Not only the current word but also neighboring words are also essential to understand their association.

Words, tags, and associated tags are very essential information in POS tagging process. As it is mentioned in chapter three, 30 tags are identified for this thesis work. So, for each word, its occurrence is calculated to be one specific tag. This is calculated using lexical probability concept, i.e., word frequency approach. It is based on the probability that a word occurs with a particular tag. Frequency measurement, i.e. (p (word/tag)), is the probabilities of a word given a category. The lexical generation probability is the probability that a given category is realized by a specific word and is estimated simply by counting the number of occurrence of each word by a category:

$$P(Wi\backslash Ti) = \frac{\text{number of times } Wi \text{ appears in tag } Ti}{\text{Total number words with tag } Ti}$$

For each word, there are 30 values, as it is shown in Table 4.1., indicates the presence of 30 input vectors in each word

Table 4.1 Probability of words given a tag

Word/tag	*T1*	*T2*	*T3*	...	*T30*	**Total**
W1						
W2						
W3						
...						
Wn						
Total						

The second feature extracted for each word is represented from the contextual information, which is calculated by transitional probabilities. The best tag for a given word is determined by calculating the probability that the word occurs with n-previous tags. Tag sequence probabilities indicate the probability of one tag to follow another tag. This is estimated simply by counting the number of times each pair of tags occurs compared to individual tag count.

$$P(tag_1, tag_2 / tag_1) = \frac{\text{the number of times } tag_1 \text{ and } tag_2 \text{ occur together in the corpus}}{\text{Number of times } tag\ 1 \text{ occur in the corpus}}$$

In this case, the left and the right tags for a given word have to be considered (l, t, r) where l is the left tag, t is the tag for the current word and r is the right tag to the current word. As a result, there will be many sequences of tags in the right and left of the current tag. Due to several reasons, it is very difficult to consider all sequence of tags in both sides. Thus, only the maximum two tags on the left and another two tags on the right are considered and seven arrangements are found. These are:

- (0, t, 0): adjacent tags to the left or right side are not considered. The tag of the word is assigned based on the input of the current word and tag, which is purely unigram and it is represented by 30 input vectors
- (1, t, 0): Only one tag which is found on the left is considered and 60 numbers of neurons are required as input.
- (0, t, 1): 60 input vectors from the target word tag and the right tag.
- (1, t, 1): one word in each side is considered and 90 input vectors.
- (2, t, 1): a total of 120 input neurons where 60 from left, 30 the target word, and 30 neuron from right side are considered.

- (1, t, 2): similar to the previous scenario except more from the right side is considered.
- (2, t, 2): equal size input neuron is given in each side of the target word and a total of 150 input neurons are considered including the tag of the current word.

4.3.2 Representation of input for MLP _tagger

The tagging of POS can be tackled in two levels: at sentence level and at word level. In the case of word level tagging, the problem can be posed as a classification problem. Whereas, in the case of sentence level tagging, a series of tags corresponding to the sequence of words in the sentence need to be found. In this case, the context provided by the whole sentence influences the tag assignment.

In this thesis work, the MLP-Tagger with back propagation algorithm addresses the classification problem posed by the word level POS tagging. The input to this network is the set of words that fall into a window of pre-specified size centered on the target word to be tagged. The output of the network is the corresponding tag for the target word. The network learns the word-tag mapping as a complex function; ***F (target word, context) = tag***. The context refers to the set of words in the immediate neighborhood of the target word, i.e, it performs tagging using a fixed length context.

Each word ,W, from the corpus is encoded as n-element vector $\mathbf{W} = (p_1, p_2, p_3, \cdots p_k \cdots p_n)$, where n corresponds to the total number of tags and P_k is the prior probabilities (lexical probabilities) that the word W corresponds to the tag Tk. P_k is estimated from ***P(word/tag)***. Each word in a text document is processed and represented in a form of vector with 30 elements, since 30 tag categories are considered in this study.

Generally, the input IN to the MLP tagger that uses L left context and R right context for a word at index t is represented as:

$(V_{t-1}, V_{t-L+1}, \ldots, V_{t-1}, V_t, V_{t+1}, \ldots, V_{t+R})$ where $V_{t\pm i}$ is an n dimensional vector.

Hence, the elements of a target word and the elements of its context word(s) are concatenated together to form an input vector to the neural network. Input **K(R+L+1)** where R and L are lengths of the left and right context for the target word t and K is 30. R and L will be determined after experiment.

4.3.3 Representation of output for MLP_Tagger

The expected output is from the defined 30 tagsets. These tags are given numeric code in decimal number. The decimal number is converted into binary number, which has 5 bits because the total tagset is 30. The output is expected by these 5-bits binary number. Each tag has its own unique 5 bit representation.

4.3.4 Back Propagation algorithm

Multi-layer perceptron model (having three layers) and Back Propagation algorithm is selected as a supervised learning algorithm. MLP has an input layer of source nodes and an output layer of neurons. These two layers connect the network to the outside world. In addition to these two layers, usually it has one or more layers of hidden neurons. These neurons are not directly accessible. The hidden neurons extract important features contained in the input data. The goal of this type of network is to create a model that correctly maps the input to the output using training data so that the model can be used to produce the output when the desired output is unknown.

The back propagation algorithm has a number of properties that made it to be highly appropriate to use. The network does learning and re-learning, i.e., if the network is distorted to a particular performance level, it re-learns faster than a new network starting at the same performance [45]. Due to this reason, this algorithm is selected for this thesis work.

In BP algorithm, the network is first initialized by setting up all its weights to be small numbers. Next, the input pattern is applied and the output calculated. The calculation gives an output which is completely different from what is expected, since all the weights are random. Error of each neuron is calculated, which is essentially: Target - actual output. This error is then used mathematically to change the weights in such a way that the error will get smaller. In other words, the output of each neuron will get closer to its Target. This process is repeated again and again until the error is minimal.

The algorithm works as the following:

1. First apply the inputs to the network and work out the output
2. Next work out the error for neuron.

   Assume neuron B;

$$Error_B = Output_B (1- Output_B)(Target_B - Output_B)$$

The "Output(1- Output)" is necessary in the equation because of the Sigmiod Function- it would just be (*Target - Output*) if threshold neuron was used.

3. Change the weight. Let W^+_{AB} be the new (trained) weight and W_{AB} be the initial weight.

$$W^+_{AB} = W_{AB} + (Error_B \times Output_A).$$

4. Calculate the Errors for the hidden layer neurons. This is done by taking the Errors from the output neurons and running them back through the weights to get the hidden layer errors.

   For example if neuron A is connected to B and C then the errors from B and C to generate an error for A can be taken.

   $Error_A = Output_A (1- Output_A) (Error_B W_{AB} + Error_C W_{AC})$

   Again the factor "*Output (1-Output)*" is present because of the sigmoid function.

5. Having obtained the Error for the hidden layer neurons now proceed as in stage 3 to change the hidden layer weights. By repeating this method we can train a network of any number of layers.

BP algorithm helps to know some features of it when training neural networks. Internally, most BP networks work with values between 0 and 1. If inputs have a different range, it will scale each input variable minimum to 0 and maximum to 1. The algorithm changes the weights each time by some fraction of the change needed to completely correct the error. This fraction is the learning rate. High learning rates cause the learning algorithm to take large steps on the error surface, with the risk of missing a minimum, or unstably oscillating across the error minimum. Small steps, from a low learning rate, eventually find a minimum, but they take a long time to get the minimum. Some neural network simulators can be set to reduce the learning rate as the error decreases.

The algorithm finds the nearest local minimum, not always the lowest minimum. The commonly used solution in BP is to restart learning every so often from a new set of random weights (i.e. somewhere else in the weight space). It finds the local minimum from each new start keep track of the best minimum found.

4.4 Design of Rule-Based Tagger

The neuro tagger tries to tag based on the complete context. However, the POS of a word can be determined with certainty only by using the word on the left. What the neuro tagger can easily acquire by learning is the rules whose conditional parts are constructed by all inputs that are joined with AND logical operator. In other words, it is difficult for the neuro tagger to learn rules whose conditional parts are constructed only by a significant input. Even though lexical information is very important in tagging, it is difficult for the neuro tagger to use it, because doing so would make the network enormous. That is, the neuro tagger cannot acquire rules. Furthermore, because of convergence and over-training problems, it is also not advisable to train neural nets to an accuracy of 100%. The training should be stopped at an appropriate level of accuracy. Thus, neural net may not acquire some useful rules. Rule-based is designed in this work to alleviate this limitation. Transformation-based error-driven is the technique which is selected for developing rule-based tagger.

4.4.1 Transformation-based learning

The rule-based tagger makes up the aforementioned crucial shortcomings by acquiring rules from a training corpus using a set of transformation templates by transformation-based error-driven learning. The templates are constructed using only those that supply the rules which the neuro tagger can hardly acquire. It acquires the rules with single input, with lexical information, and with AND logical input of POSs and lexical information.

Sequences of transformations to improve the tagging of an initial basic tagger are learnt from a tagged corpus. The transformations are in the form of rules: if the following triggering conditions match, then change tag X to tag Y. The main advantage is that the transformation rules

can use much more specific information and context in determining whether they should be applied.

Transformation based learning starts with, less accurate, rules that learn better ones from tagged corpus. A word is tagged initially with the most likely POS. Each set of transformations is examined to see which rule improves tagging decisions compared to tagged corpus. The process goes iteratively: retag corpus using best transformation and repeat until performance becomes more improved. The result will be used as tagging procedure (ordered list of transformations), which can be applied to new, untagged corpus.

As it is shown in Figure 4.1, this type of learning goes through a number of stages. First, the unannotated text has to be given, and then initial is applied on these unannottated corpuses. The rule–based tagger learns from the annotated corpus and the truth. Then iteratively applied on already annotated text. Finally, rules are generated from this process.

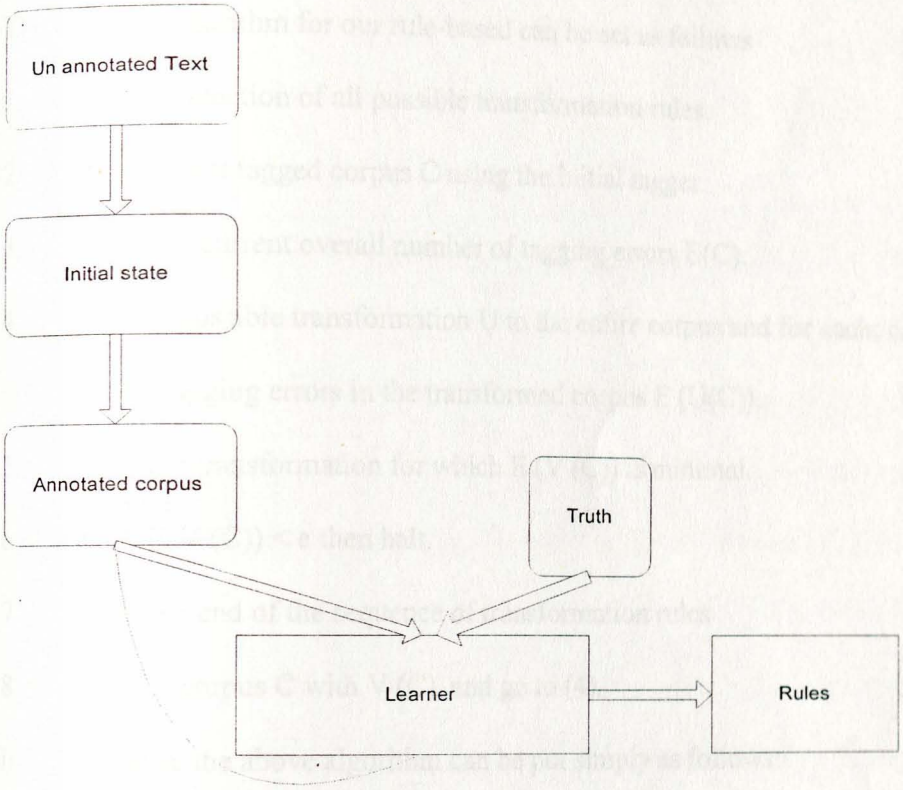


Figure 4.1 Transformation-based learning process

4.4.2 Learning Algorithm

Rule-based tagging using transformation-based learning requires two main components: constructing the set of valid possible transformation triggering conditions and the learning algorithm.

The inventory of transformations is a function of the types of possible triggers for applying a transformation. The most common types of triggers are:

- Tag X occurs/doesn't occur in one of the previous/next Y positions.
- Word X occurs/doesn't occur in one of the previous/next Y positions.
- Morphological conditions on preceding/following words
- Combinations (multiple conditions) of the above

The learning algorithm for our rule-based can be set as follows

1. Create the collection of all possible transformation rules.
2. Create an initial tagged corpus C using the initial tagger.
3. Calculate the current overall number of tagging errors E(C).
4. Apply each possible transformation U to the entire corpus and for each; calculate the number of tagging errors in the transformed corpus E (U(C)).
5. Select V, the transformation for which E (V (C)) is minimal.
6. If E(C) < E(V (C)) < e then halt.
7. Add V to the end of the sequence of transformation rules
8. Replace the corpus C with V (C), and go to (4).

In other words, the above algorithm can be put simply as follows:

- Initialization:
    - Each word is tagged with the most likely tag.
- Learning Phase
    - Iteratively compute the error score of each candidate rule (difference between the number of errors before and after applying the rule)
    - Select the best (higher score) rule.
    - Add it to the rule set and apply it to the text.
    - Repeat until no rule has a score above a given threshold, that is, until applying new rules leaves the text in the same state, which is then supposed to be the final state of the tagging.

According to this algorithm, the rules are applied on the annotated text and rules are taken at one time learning. But rules which generate minimal error are considered in transformation-based learning.

4.5 Architecture of Hybrid System tagger

The hybrid system consists of a neuro tagger, which is used as an initial-state annotator, and a rule-based corrector, which corrects the outputs of the neuro tagger. When a word sequence W_t is given, the neuro tagger output will have a tagging result $T_N(w_t)$ for the target word w_t at first. The result of the neuro tagger is given to the output analyzer. Depending on the threshold value, the output analyzer makes decision. If the threshold value of a given word is below the expected confidence level, a word is given to the rule-based corrector. Otherwise, the result of the neuro tagger is taken as a final tag for a given word. This is accepted when the threshold value is equal or above to the expected confidence level. The threshold is used as a lower bound, i.e., tags with lower probabilities are given to the rulel-based tagger. As it is seen in figure 4.2, the architecture of the hybrid tagger is discussed briefly so as to get the expected result using the above mentioned approaches.

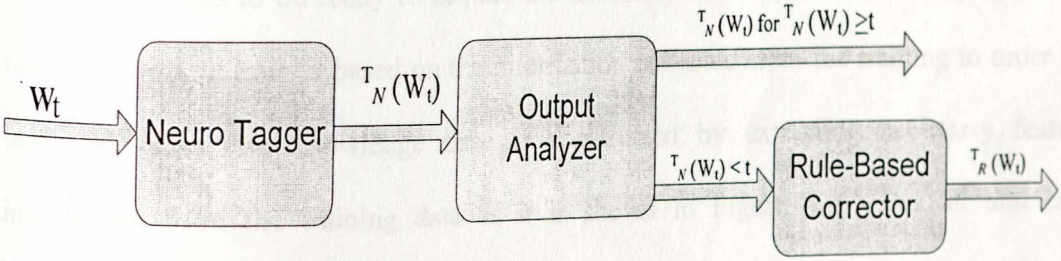


Figure 4.2 Hybrid Tagger architecture.

The intention of this work is to give and produce a text which is tagged by appropriate tagset depending on context. But to perform such assignment, at high level of description, the system should take Amharic text as input. The next step is processing this text to assign a tag that

describes words' role in a given sentence or phrase. The tagger, which is made up of the neuro and rule –based taggers, is taken as processor to this task. Two main inputs are necessary in order to process the word to assign tag as shown in Figure 4.3. These are: the text to be tagged and the knowledge which is acquired from the training data. Finally, by accounting these two inputs, it produces the output which is a tagged text. Knowledge Base module and Tagger module are found in this system plus different inputs and outputs to and from the system.

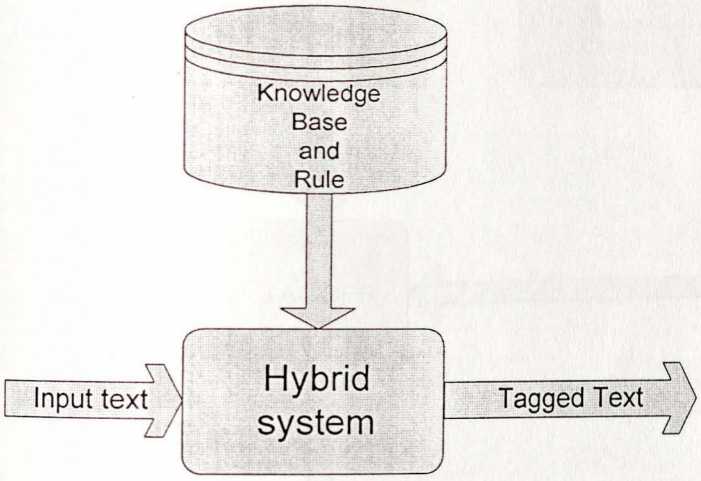


Figure 4.3 High level description of hybrid

4.5. 1. Knowledge Base Module

Training data has to be ready to acquire the necessary information in the tagging process. The neural network is trained based on the information collected from the training in order to assign tags to words. The knowledge base is constructed by extracting necessary features and information from the training data as it is shown in Figure 4.4. Lexical and contextual information of words and their tags from the training set are the basic components to make the knowledge base. Lexical information is gained by taking in to account a word given category, tagset, in a given corpora, training set.

In a clear term, this gives us the probabilities of a given Amharic word in a given training data to be a noun, verb or others. Contextual information is also another essential element to the knowledge base. Given one or more previous tags, there is a potential of a tag to be presented. To summarize, the knowledge base handles these two important data by analyzing the relationship of each word and tag against the tagsets from the training data.

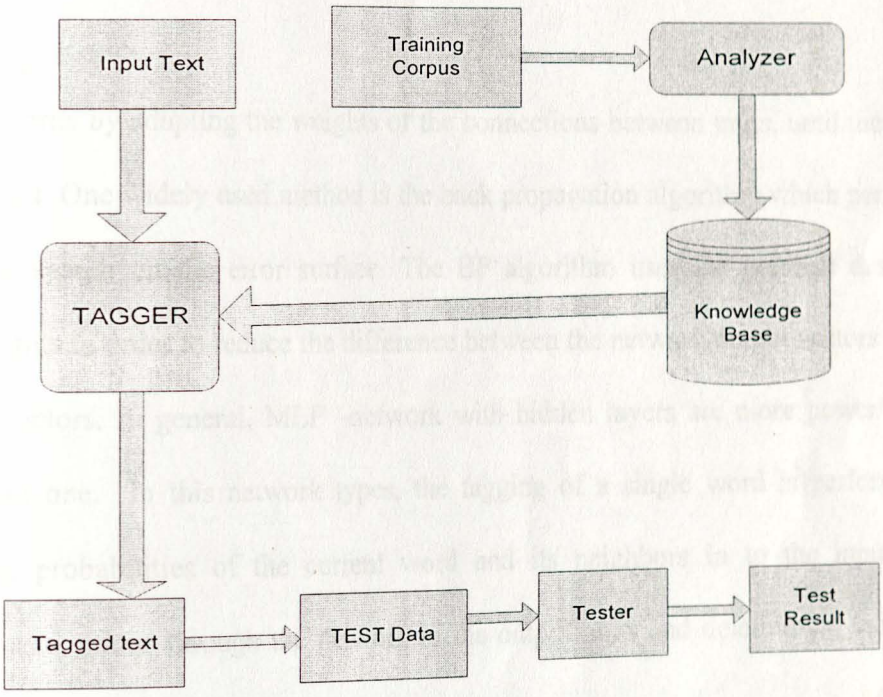


Figure 4.4 Process of Hybrid

4.5.2 Tagger Module

Tagger module is the second core module to the system of the hybrid tagger. It has a direct attachment to the knowledge base because it is an area that the tagger can get the necessary information about each word and tagset. Both approaches, neural and rule-based, are doing by learning so that these information are necessary to do the tagging processes. After acquiring this important information, the next step is to perform the actual tagging task. But, two inputs must be

there additionally for the pre-defined tagsets and the text to be tagged. Tagset is a set of tags, which are a collection of distinct coding or tags for all classes of words having distinct grammatical behavior. The possibility of one word to be assigned at a time requires one tag only from the tagset. Within the tagger, there are sub module taggers which are referenced as neuro and rule-based tagger one after the other respectively. After the tagger is going all these through, it generates the tagged text

4.6 Summary

MLP-Network learns by adapting the weights of the connections between units, until the correct output is produced. One widely used method is the back propagation algorithm which performs a gradient descent search on the error surface. The BP algorithm uses the gradient descent to change link-weights in order to reduce the difference between the network output vectors and the desired output vectors. In general, MLP –network with hidden layers are more powerful than networks without one. In this network types, the tagging of a single word is performed by copying the tag probabilities of the current word and its neighbors in to the input units, propagating the activations through the network to the output units and determining the output unit which has the highest activation value.

Besides this, Transformation- based learning process is adopted to design the rule-based tagger. The incorporation of these two approaches makes the system hybrid. This system is expected to be better because no tagger is designed that consists of neural network and rule-based approach concerning Amharic language. This type of design helps a better accuracy than a system designed by single approach.

Chapter Five

Experimentation and Discusion

In chapter four, the design of Amharic part of speech tagging and algorithms are discussed. This chapter presents the process of data preparation and the experiments conducted. Experiments are done in different scenarios so that the results are displayed accordingly with their explanations. The following two sections describe these phases.

5.1 Data preparation

Most of the time Natural Language Processing is a data-intensive field. The success or failure of most NLP applications depend on the quality and availability of appropriate data. The data used in computational linguistic tasks generally takes the form of corpora. Corpora can be divided into two categories: annotated corpora and unannotated corpora. Unannotated corpora are simply large collection of raw text, where as annotated corpora add additional information to the text, such as phonetic transcription, part-of-speech tags, parse trees, etc. Annotated corpora with appropriate part of speech are useful to train part of speech taggers.

5.1.1 Source of Sample Data

The sample data is acquired from Ethiopian Languages Research Center (ELRC), Addis Ababa University, Ethiopia. ELRC conducted a project called "The Annotation of Amharic News Documents" [24]. This project was meant to tag each Amharic word in its context with the most appropriate part of speech manually. The project aimed to annotate 210,000 prosodic words found in 1065 Amharic news documents. The news documents were obtained from Walta Information Center, a private news agency located in Addis Ababa, Ethiopia, that makes daily news in Amharic and English through their website [24]. The news items, as available in the website, are represented using Ethiopic script (Fidel) and were archived according to the

Ethiopian calendar. A number of works have been done to bring the news texts obtained from Walta Information Center for tagging and making it finally available in a way easier for users.

Neural network requires large amount of data in order to do successful training process. Thus, it is important to have large size data which is annotated. The data is prepared mainly to support researches conducted on Amharic Natural Language Processing. This is the first and most criteria to use this data. Secondly manual annotation of text requires language expert and time. The manually annotated corpus by ELRC avoids such burden. At the same time, the data is annotated by linguists and even evaluated by senior experts in the domain. So that it is more reliable and available data to those who are interested to do their researches on Amharic language in this specific area. Due to these main reasons, the sample data for training and testing comes from ELRC.

5.1.2 Lexicon preparation

A lexicon is prepared from the sample corpus which is tagged manually. This lexicon is important to prepare the matrix of lexical probabilities as shown in Table 5.1.

Table 5.1 The lexical probabilities

Words	N	V	ADJ	PREP	ADV	....	Total
W1	5	1	0	0	1		7
W2	1	4	2	0	2		8
Total	***6***	***5***	***2***	***0***	***3***		***15***

As we can observe from the above table, the horizontal part provides the frequency of each word in association with each part of speech in the corpus. Each entry also indicates how many times each word occurred in each part of speech and the sum of the word in the corpus. It is possible to

get additional information from the vertical part of the table because it shows the number of words in each part of speech category.

5.1.3 Lexical probabilities

Each word in a text document is represented in a form of vector with 30-elements. The lexicon probabilities are essential to represent each word in those vectors. These lexicon probabilities can be obtained from the lexicon which is shown in Table 5.1.

The lexicon probabilities are estimated simply by counting the number of occurrences of each word by category. The probability values were computed using the information in Table 5.1 and the formula:

$$P(W_i \mid C_i) = \frac{\text{number of times } W_i \text{ appears in category } C_i}{\text{total number of words with category } Ci}$$

5.1.4 Database design

As it is described in chapter 4, knowledge base is one aspect to store the information which are important to train and test the tagger. A simple database that has three tables is constructed using Microsoft Access software. These are lexicon table, training table and testing table. The full form lexicon contains distinct words of the corpus with their corresponding lexical probabilities. The train and test tables contain words for training and testing respectively.

5.1.5 Representation of Input for MLP Tagger

The input to the neural network is the set of words that fall into a window of pre-specified size centered on the target word to be tagged. The output of the network is the corresponding tag for the target word. The network learns the word-tag mappings as a complex function like the following:

F (target word, context) = tag. Here, the context refers to the set of words in the immediate neighborhood of the target word; i.e. it performs tagging using a fixed length context. For example, in order to determine the tag of the target word **በነጻ** in the Amharic sentence " ትምህርት ለሁሉም ዜጋ በነጻ መሰጠት አለበት" the words ትምህርት, ለሁሉም, ዜጋ, መሰጠት,and አለበት can be taken as context words i.e. F (ትምህርት_ ለሁሉም_ ዜጋ _በነጻ _መሰጠት_ አለበት) = tag.

Each word W from the corpus is encoded as n-element vector W= (P_1, P_2, P_3, ...P_k...P_n), where **n** corresponds to the total number of tags and $\mathbf{P_k}$ is the lexical probability that the word W corresponds to the tag $\mathbf{T_k}$. $\mathbf{P_k}$ is estimated by using:

$$P(word/tag) = P_k = P(w/T_k) = n(Tk,w)/n(w)$$

Where

- $n(Tk,w)$ is the number of occurrence of the word W labeled as the tag $\mathbf{T_k}$ in the corpus and
- $n(w)$ is the number of occurrences of the word W in the corpus.

For example, P(ሀይሌ/N) = 8/9 = 0.889 and P(ሀይሌ/NC)= 1/9 = 0.111 because n(ሀይሌ, N) = 8, n (ሀይሌ, NC) = 1 and n(ሀይሌ) = 9.

Each word is represented and processed in a form of vector with 30-elements because there are 30 tags in this work.

Generally, the input **IN** to the MLP tagger can be represented as follows:

IN=(W_{T-L},... W_{T-2}, W_{T-1}, $\mathbf{W_T}$, W_{T+1},W_{T+2},...,W_{T+R})

Each element Wk (Wk Σ W_{T-L}, ... W_{T-2}, W_{T-1}, $\mathbf{W_T}$, W_{T+1},W_{T+2},...,W_{T+R}) is an n-element lexical probability vector encoded as described above. Thus, the input **IN** to the MLP-tagger comprises information about the target word $\mathbf{W_T}$, **L**- number of words on its left and **R**-number of words on its right.

Hence, the elements of a target word and of its context word(s) are concatenated together to form an input vector to the neural network.

- Scenario 1: An input vector with 60-elements for a target word and one word immediately before it as its context. This can be represented as *previous word_target word* (1_T_0; L=1 and R=0).
- Scenario 2: An input vector with 90-elements for a target word and two words immediately before it as its context. This can be represented as *second previous word_first previous word_target word* (2_T_0; L=2 and R=0).
- Scenario 3: An input vector with 90 –elements for a target word and one word immediately before it and one word immediately after it as its context. This can be represented as *previous word_target word_next word* (1_T_1; L=1 and R=1).
- Scenario 4: An input vector with 120–elements for a target word, two words immediately before it and one word immediately after it as its context. This can be represented as *second previous word_first previous word_target word_next word* (2_T_1; L=2 and R=1).
- Scenario 5: An input vector with 150–elements for a target word, two words immediately before it and two words immediately after it as its context. This can be represented as 2_T_2; L=2 and R=2.

5.1.6 Representation of output for MLP Tagger

The output of each word is expected from the defined 30 tagsets. Each tagset is represented by 5-digit binary number. The decimal and binary representation of each tag is presented in table 5.2

Table 5.2 Binary and decimal number representation of tags

NO.	*Type of tag*	*Binary representation*	*Decimal representation*
1	N	11111	31
2	VN	11110	30
3	NP	11101	29
4	NC	11100	28
5	NPC	11011	27
6	PRON	11010	26
7	PRONP	11001	25
8	PRONC	11000	24
9	PRONPC	10111	23
10	V	10110	22
11	AUX	10101	21
12	VREL	10100	20
13	VP	10011	19
14	VC	10010	18
15	VPC	10001	17
16	ADJ	10000	16
17	ADJP	01111	15
18	ADJC	01110	14
19	ADJPC	01101	13
20	PREP	01100	12
21	CONJ	01011	11
22	ADV	01010	10
23	NUMCR	01001	9
24	NUMOR	01000	8
25	NUMP	00111	7
26	NUMC	00110	6
27	NUMPC	00101	5
28	INT	000100	4
29	PUNC	000011	3
30	UNW		

The output for MLP_Tagger can be represented as follows: **OUT= (O1, O2, O3, O4, O5).** For example: F (target word, context) = N if OUT = (11111) =31. It is the same for all tags and finally otherwise F (target word, context) =UNW (unknown, word).

5.2 Experiment

In this study the problem of part of speech tagging is carried out by using Artificial Neural Network and Rule-based technique. BrainMaker is a tool that is selected for modeling and testing purposes concerning the Artificial Neural Network approach; and Brill's Tagger is the tool used for the Rule-based approach.

5. 2.1 BrainMaker Neural Network Software

Brain Maker is designed by seven Caltech mathematicians and engineers at California Scientific Software Company [12]. This tool is selected because of the following reasons. It is for people who demand the most highly powered development tools and greatest ease of use. This software uses Back Propagation Algorithm in developing neural network model. It allows one to easily find a network that tests well and terminates at a predefined point. Network Progress Display shows graphically how well the network is learning. This tool helps to determine the accuracy level, how soon it will do training, and how well it is doing at any time. It is possible to train a neural network and then reduce or increase the number of hidden neurons without having to start training all over. BrainMaker cuts out the least significant neuron. This helps us to train a network that generalizes well [12].

5.2.1.1 How BrainMaker Works?

BrainMaker Neural Network software has two programs known as NetMaker and BrainMaker. NetMaker can import data file from Lotus, ASCII, EXCEL, Text, dBase, Binary, etc. Both numeric data and text data can be accepted by NetMaker and converted into a representation that

the neural network can understand in the range of 0 and 1. The imported files are seen on NetMaker as a spreadsheet. After importing the data set in to NetMaker, the input fields (independent variables) and pattern fields (dependent variables) are determined and labeled. For this thesis, the independent variables are features of a target word and its context while a pattern (dependent variable) is the classification field that indicates whether the tag is noun, verb, etc. Then the prepared file is saved as a ***.dat*** extension.

Three BrainMaker files are created based on the file *.dat. The first file is the definition file (*.def) that has the definition information for training such as what columns are inputs and patterns, and how the information is displayed. The second BrainMaker file is the fact file (*.fct) for training and the last one is running fact file (*.in) created to predict future records. After the above three files are created, we have to move from NetMaker to the BrainMaker program to accomplish the training and testing activities.

The BrainMaker program has different parameters with different possible values for each parameter that are essential in neural network training and testing. Training tolerance, learning rate, smoothing factor, number of hidden layers, number of neurons in hidden layer, and type of function are the most important parameters in BrainMaker. Training is started using the ***Operate/Train Network*** command after determining those parameters. While training is in progress, statistical information are provided on the screen like which fact the BrainMaker is processing at specific time, the number of facts which meet and did not meet the training tolerance, and the number of run. There are also two graphs that display the progress of training. Histogram is the first graph that presents the distribution of error over an entire run. The horizontal axis represents the error level and the vertical axis signifies the number of output values at a particular level. As training is in progress and fewer facts are classified as incorrect,

and the bars move to the left. The second shows the progress of the error rate as network trains. In this graph, the horizontal axis shows the number of runs while the vertical axis represents the overall error level (root mean square (RMS)). For good training, the value of RMS would decrease as the number of runs increase.

Training can be stopped at any time without instruction to terminate the training. The default for stopping training process is when the incorrect classification of facts in a single run (epoch) is zero. Better network model can be obtained before the criteria for stopping training are met. Therefore, it is advisable to test and save a network model periodically.

5.2.1.2 Experiments With BrainMaker Software

This part discusses the experiment conducted using BrainMaker neural network software for predicting the type of tag for each word in a given sentence. 210, 000 words are considered for the experiment of this thesis work. 10 % of the total data is for testing and the rest is for training. The main reason is that performance declines when the test data is below 10% and no major variation in the result is observed as the test data is above 10%. The BrainMaker software has the facility where data are classified in to training and testing sets. The suppliers of this software also suggested that BarinMaker performs better when the test data is 10% of the total data[12].

Training and testing a neural network model for part of speech tagger can be done using the following procedure [9]:

1. Determine neural network size (number of input, hidden and output nodes).
2. Decide on the node activation function to be used
3. Initialize the network weights
4. Training the neural network using the back propagation algorithm and the selected training data set.

5. If training time, error goal or epochs are satisfactory, save weights.
6. If not, vary number of hidden nodes and/or re-initialize weights and go to step 4.
7. Test network
8. If the network is at expected performance level, stop it
9. If not, go to step 6.

Using this procedure, the number of input nodes is determined as 60, 90, 90, 120 and 150 for scenario one, two, three, four and five respectively as described in section 5.2.6. The number of output nodes is 5 for each scenario.

The first training is conducted using scenario one that is an input vector with 60-elements to represent a target word and one word immediately before it as its context (***Previous word_Target word***). Initially, training is started based on the default network parameters. The default parameters for the neural network are: training tolerance=0.100; learning rate=1.0; training noise=0.00; smoothing factor=0.9; number of hidden layers=1; number of neurons in hidden layers=60 which is the same as number of nodes in the input layer by default; type of neuron function=sigmoid; network weights=small randomly chosen numbers and gives the progress.

The training process can be interrupted when the distribution of errors and the root mean square error value stopped to decrease. The model constructed using the default values predict 93 % on average. Since the test result of this network model was encouraging, the experiment continued by considering various options suggested to improve the performance of neural network models. For example, the vendors of BrainMaker software suggested that when a network model got a sufficient number of training facts, correct and perform fairly well in testing process, build another network model by changing number of nodes in the hidden layer, add noise to the

network, by changing the number of hidden layers, change training tolerance and learning rate [12].

First, it is tested whether an additional hidden layer in the network would improve the accuracy of tagging by keeping the default values for other parameters constant. This network model also fails to learn all the training facts and the accuracy deteriorated slightly in comparison to the above network.

Secondly, by keeping the default value for other parameters constant, the default values of training tolerance was changed to 0.01. But in this experiment, the default value smoothing factor (0.900) had not been changed for any of the network models because the providers of this software stated that adjusting the smoothing factor has not been found to reduce training time or improve prediction power of the network in every case.

This network model also fails to learn all the training facts and resulted in a little bit better performance than the network model under default values. Again when the training tolerance was changed to 0.001, it is resulted in a little bit better performance than the network model under default values.

Generally, numerous experiments were carried out by varying the values of parameters. From all the experiments for scenario one (1_T_0), it became clear that variation of default values was not resulting in a significant change to the performance of the network model. This is true for scenario two, three, four and five as it is displayed in Table 5.3.

Table 5.3 Performances of Network Models for different values of parameters

Scenario type	HL=1 HN=60 TT=0.1	HL=2 HN=60 TT=0.1	HL=1 HN=60 TT=0.01	HL=1 HN=60 TT=0.001
1_T_0	93.1 %	91.2 %	93.9 %	93.11 %
2_T_0	92.7 %	91.1 %	92.8 %	91.5 %
1_T_1	94.1 %	93.8 %	94.3 %	95.4%
2_T_1	93.4 %	92.6 %	92.1 %	92.0 %
2_T_2	91.1 %	78.3 %	83.2 %	86.7 %

Legend:

HL= number of hidden layers TT= training tolerance

HN= number of nodes in the hidden layer

As it is observed from Figure 5.1, better accuracy is achieved on the third scenario i.e. 1_T_1. The neuro tagger assigns the more appropriate tag to each target word when it considers one word immediately before it and one word immediately after it as its context. Even tough there are minor variations when the number of hidden layers and nodes in hidden layers, training tolerance and default values are changed, its accuracy reaches 94.4% on average on this scenario.

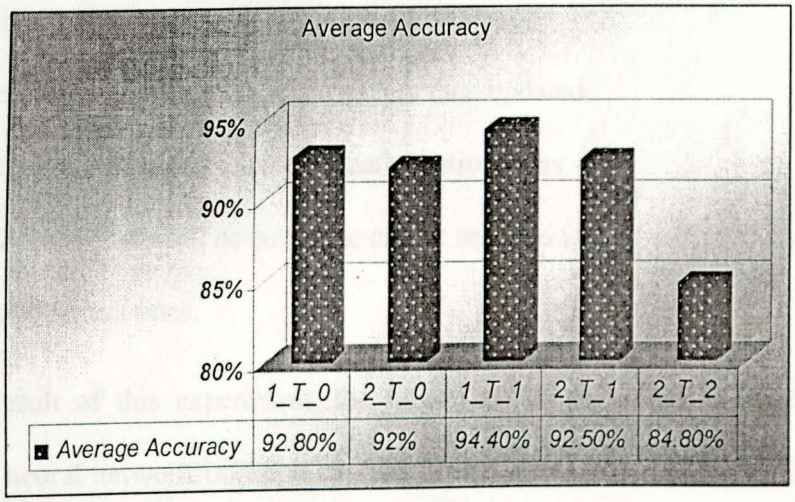


Figure 5.1 Average accuracy in each scenario

5.2.2 Experiment with Brill's Tagger

Transformation-based learning method is chosen to evaluate the rule–based approach. Brill's tagger is used to train and test data. This tool was developed by Eric Brill for such research purposes specifically on transformation-driven rule-based approach [11]. The next step is applying the templates to the training corpus which had already tagged by the neuron tagger.

According to the learning procedure listed below, an ordered list of transformation rules is acquired by applying the template set to a training corpus. After the transformation rules are acquired, a corpus is tagged as follows. The tagged corpus is then corrected by using the ordered list of transformation rules. The correction is a repetitive process applying the rules on the corpus, which is then updated, until all rules have been applied.

Procedures for learning transformation rules:

1. Apply neuro tagger to training corpus, which is then updated.
2. Compare tagged results with desired ones and find *errors.*
3. Match templates for all errors and obtain set of transformation rules.
4. Select rule in corpus with the maximum value.
5. Weight to control the strictness of generating the rule.
5. Apply selected rule to training corpus, which is then updated.
6. Append selected rule to ordered list of transformation rules.
7. Repeat steps 2 through 6 until no such rule can be selected i.e. rules that transform correct tags to incorrect ones.

Based on the result of this experiment, the rule-based tagger achieved around 91% without considering the neural network information. This result shows the rule-based tagger accuracy is less than a neural tagger. Due to this, the rule-based tagger is used as a corrector after a word is

tagged by a neural tagger and its value is below a minimum threshold value which makes it a hybrid tagger.

5.2.3 Result of the hybrid tagger

As it is stated above, rule-based tagger is considered as a corrector in this thesis work. It is used to post processes the outcome of the neuro tagger if the value of the tag is below a given threshold value. The experiment is done by setting different threshold values from starting from 0.5. 0.5 is set as a threshold value because there is no significant change when the threshold value is below this value. The experiment is conducted in the following procedure.

Let **N** be the total word to be tagged by the hybrid tagger, 21,000 words and **M** of them are above the given threshold value. $\mathbf{M_1}$ of them are correctly tagged by the neural tagger and $M_2 = M\text{-}M_1$ are wrongly tagged words. Let **K=M- N** words which are below the threshold are given to the rule-tagger and K_1 of them are correctly recognized and $K\text{-}K_1 = K_2$ is wrong. Table 5.4 shows the result at various thresholds.

As it is shown in Table 5.4, columns 5, 6, 10 and 11 show the various performances. Column 5 shows the performance of the NN among the words above the threshold value and Column 6 shows the performance of the NN out of the total words, N. Column 10 indicates the performance of the rule-based tagger on the K words whereas Column 11 shows the performance of the entire hybrid system. As shown in Table 5.4, the hybrid system shows the maximum performance when the threshold value is 0.9

Table 5.4 Performance of the NN, rule-based and hybrid system at various threshold values.

Threshold Values	M	M_1	M_2	$\frac{M_1}{M}$	$\frac{M_1}{N}$	K	K_1	K_2	$\frac{K_1}{K}$	$\frac{M1 + K_1}{N}$
0.5	15,704	13,741	1,963	85.7	65.4	5,296	4,565	731	86.2	87.17
0.6	14,872	12,968	1,904	87.2	61.7	6,128	5,460	668	89.1	87.75
0.7	13,457	12,461	996	92.6	59	7,543	7,068	475	93.7	92.99
0.8	11,241	10,690	551	95.1	50.9	9,759	9398	361	96.33	95.65
0.9	8,816	8,816	239	97.3	40.8	12,184	12013	171	98.6	98.05

As the threshold value increases, the rule-based tagger corrects more words. As a result of this, the hybrid tagger scores the highest performance as the threshold value goes up, as it is indicated in Figure 5.2.

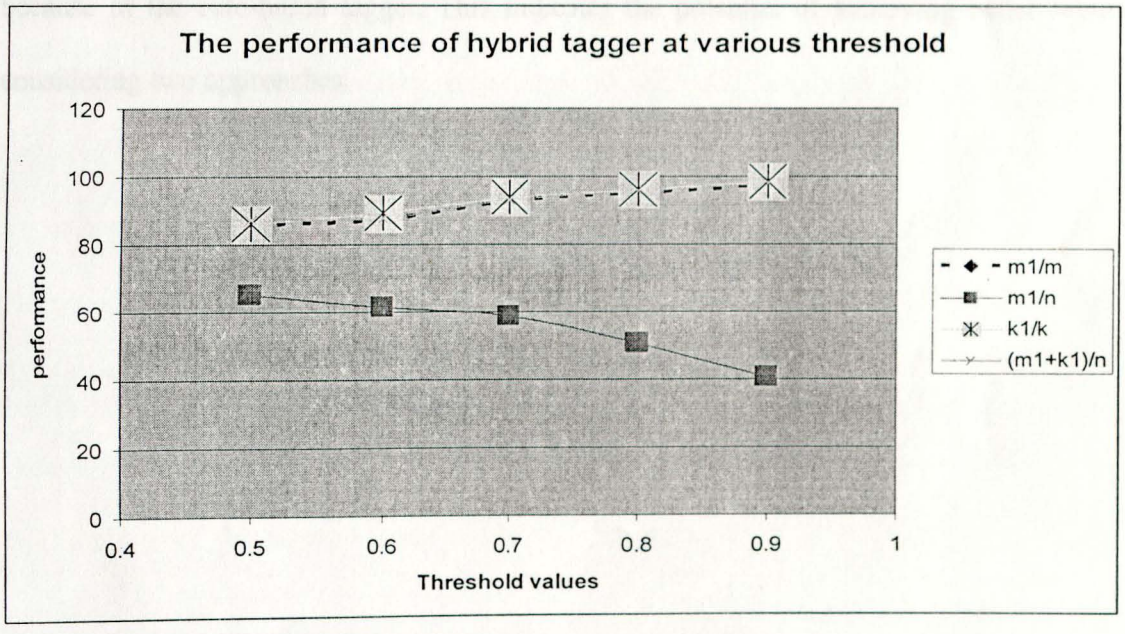


Figure 5.2 The performance of hybrid tagger at various thresholds.

In conclusion, the hybrid system achieves better performance than the NN, 94.4%, and the rule-based tagger, 91%. The performance increases because of the combination of the two approaches.

5.3 Summary

In this chapter, an MLP-tagger is presented for Amharic language as initial annotator. The result of neuro tagger is given as input to rule-based tagger when it is below a given threshold value. The current work uses neighborhood context for tagging of words in a large tagged corpus. The MLP-tagger combined with representation scheme for inputs and outputs could be a promising result for further research in the same direction for Amharic language corpora. Changing the default values of different parameters did not give significant improvement. Similarly enlarging the context more than (1_T_1) did not give improvement. The MLP-part of speech tagger which is modeled on the scenario 1_T_1 is selected and gave average accuracy of 94.4 % on words in test data that are not seen during the training process. But this accuracy is more than 98 % this because of the rule-based tagger. This indicates the potential of achieving better accuracy by considering two approaches.

Chapter Six

Conclusions and Recommendation

6.1 Conclusions

Part-of-Speech tagging is one of the research areas in Natural Language Processing (NLP) which annotates natural language sentences or texts with part-of-speech. In this thesis work, an Amharic tagger that scores better accuracy using ANN and Rule-based approaches, hybrid approach, has been designed.

Grammatical knowledge of the language is very useful in designing the POS tags and tagsets. POS tags have also been presented from Amharic language perspective. These tagsets do not indicate features like gender, number, person, tense, definiteness, etc. A total of 30 tags have been set in this thesis work.

The other core part of this thesis is the issue of designing Amharic POS tagger. MLP Neural Network and Transformation-Based Learning Rule-based type have been taken into account for the design of the Amharic tagger. In order to design the POS tagger two algorithms, Back-propagation and Transformation-based learning, have been adopted. Moreover, architecture of the system has been presented.

Data is one of the basic inputs in conducting research. About 210,000 words have been used in this thesis work. Using such large size corpus increases the performance and efficiency of the system.

Experiments using Rule-based, Neural Network, and hybrid approaches have been conducted. The achieved results of the above approaches on average are 91%, 94%, and 98%, respectively.

Though Neural Network approach provides better result than Rule-based approach, the latter one corrects most of the errors generated by the neural network. As a result, the hybrid approach became more efficient than both approaches used separately. Hence, hybrid approach has been chosen to develop Amharic POS tagger in this thesis work.

In conclusion, the combination of two approaches for developing a tagger results in a better accuracy than a single approach. Furthermore, this thesis work indicates the potential of rule-based approach to minimize the limitation of the neuro tagger.

6.2 Recommendations

There are many research areas on computational linguistic concerning languages spoken in Ethiopia. To avoid obstacles on this aspect, some points are recommended.

- Language resource center has to be set up by concerned bodies that will be easily accessible to researchers.
- NLP is now a research area so that NLP team should be organized under faculty or departmental level in collaboration with linguistics department. This will benefit to practically implement what has been done before and to bring continuity and integrity on research conducted at this or higher level.

As a future work of this research, the following suggestions are forwarded.

- Extend this work by acquiring well-annotated and large size data so as to increase the efficiency.
- Conduct researches for other Ethiopic languages by adopting this work.
- This type of tagging system can be used for the construction of multilingual corpora. This also one potential area to researches.

- Comparison study on the three approaches (statistical, rule-based, and ANN) for other Ethiopic languages based on their language family should be done.

References

1. Abiyot Bayou., *"Developing Automatic Word Parser for Amharic Verbs and Their Derivation"*, Master's Thesis at the School of Information Studies for Africa, 2000.
2. Allen James, *"Natural Language Understanding"*, 2nd ed., the Benjamin/Cummings Publishing Company Inc., California, 1995.
3. András Kuba, László FelfÖldi, András Kocsor, *"POS Tagger Combinations on Hungarian Text"*, for the 2nd International Joint Conference on Natural Language Processing, IJCNLP, Korea, 2005.
4. Atelach Alemu, Lars Asler and Mesfin Getachew, *"Natural language Processing For Amharic:* Overview and Suggestions for a Way Forward", In Tritement Automatique des Language Naturules Batt-sur-Mer11-14 jui, 2003.
5. Axay J. Mehta, Hema A. Mehta, T. C. Manjunath, C. Ardil, *"A Multi-layer Artificial Neural Network Architecture Design for Load Forecasting in Power Systems"*, International Journal of Applied Mathematics and Computer Sciences Volume 4, Number 4, 2007.
6. Barbora Vidová-Hladká, Kiril Ribarov, *"PoS Tags for Automatic Tagging and Syntactic Structures"*, In: E. Hajičová (ed.): Issues of Valency and Meaning. Studies in Honour of Jarmila Panevová, Karolinum, Charles University Press, Prague, Czech Republic, 1998, pp. 226-240.
7. Baye Yimam, *"የአማርኛ ሰዋሰው"*, Ethiopian Materials Production and Distribution Agency (E.M.P.D.A), Addis Ababa, 1986.

8. Ben Krose, Patrick van der Smagt, *"An Introduction to Neural Network"*, Eighth edition, November 1996.

9. Berhanu Aderaw, "*Amharic Character Recognition Using Artificial Neural Networks*", Masters Thesis Addis Ababa University, 1999.

10. Brill E., "*Transformation-Based Error-Driven Learning and Natural Language Processing: a Case Study in Part-of-Speech Tagging*", Computational Linguistics, Vol. 21, No. 4, pp. 543-565, 1994.

11. Brill Eric, "*A Simple Rule-Based Part of Speech Tagger*", 3rd Conference of Applied Computational Language (ACL), Trento, Italy,., 1992.

12. California Scientific Software Company, "*BrainMaker: Users Guide and Reference Manual*", Available at: http://www.calsci.com, 1998. Last visited on July 2008.

13. Carling Alison, "*Back Propagation Introducing Neural Networks*", Wilmslow: Sigma Press, pp. 147-154, 1992.

14. Chao-Huang Chang, Cheng-der Chen, "*HMM-Based Part-of-Speech Tagging for Chinese Corpora*" ACL Workshop on VLC, 1993.

15. Cheng-Yuan Liou and Yen-Ting Kuo, "Data Flow Design for the Back Propagation Algorithm", http://hdl.handle.net/2377/2325, 24-Oct-2006. Last visited on January 208.

16. Cutting, J. Kupiec, J. Pedersen, and P. Sibun., "*A Practical Part-of-Speech Tagger*", In Third Conference on Applied Natural Language Processing. ACL, 1992.

17. W. Daelemans, J. Zaurel, P. Berck, and S. Gillis, " *A Memory-Based Part-of-Speech Tagger Generator*", Proc.4th Workshop on Very Large Corpora, Copenhagen, Denmark, pp. 1-14, 1996.

18. Daniel Tianhang Hu, "*Development of Part of Speech Tagging and Syntactic Analysis Software for Chinese Text*", Bachelors of Science in Electrical Engineering and Computer Science and Masters of Engineering in Electrical Engineering and computer Science, 2001.

19. Diesenser Jana, "*Part of Speech Tagging for English Text Data*", Journal of Computer Science Pittsgurg USA, 1999.

20. S. Federici and V. Pirrelli, "*Analogical Modeling of Text Tagging*", Unpublished report, Institute di Lingustica Computaziords, Pisa, Italy, 1993.

21. Felipe Sánchez-Martínez, Juan Antonio Pérez-Ortiz, Mikel L. Forcada, "*Integrating Corpus-Based and Rule-Based Approaches in an Open-Source Machine Translation System*", in *Proceedings of METIS-II Workshop: New Approaches to Machine Translation, a workshop at CLIN 17 - Computational Linguistics in the Netherlands* (Leuven, Belgium), pp. 73-82, 11/01/2007

22. Getahun Amare, *"ዘመናዊ የአማርኛ ሰዋሰው በቀላል አቀራረብ"*, Commercial Printing Press, Addis Ababa, 1989.

23. Girma Awugechew, *Amharic Part-of-Speech*, Unpublished, 2006

24. Girma Awgichew & Mesfin Getachew, "*Manual Annotation of Amharic News Items with Part-of-Speech Tags and its Challenges*", ELRC Working Papers Vol. 2; number 1: pp 1-8, AAU Printing Press, March 2006.

25. B. Green and G. Rubin, "*Automated Grammatical Tagging of English*", Department of Linguistics, Brown University, 1971.

26. Hardie A., "*Developing a Tagset for Automated Part-of-Speech Tagging in Urdu*", Proceedings of the Corpus Linguistics 2003 conference. UCREL Technical Papers Volume 16. Department of Linguistics, Lancaster University, 2003.

27. Haykin S. "*Neural Networks*", Macmillan College Publishing Company, Inc., 1994.

28. Johan Carlberger, Viggo Kann, "*Implementing an Efficient Part-of-Speech Tagger*", John Wiley & Sons, Inc. New York, USA, Vol. 29, pp. 815-832, July 1999.

29. Jurafsky Daniel and Martin Janestl, "*Speech and Language Processing: An Introduction to Natural Language Processing Computational Linguistics and Speech Recognition*", University of Colorado Bouler Prentice Hall Inc., 2000.

30. Karel Oliva and Pavel Květoň, "*Linguistically Motivated Bigrams in Part-of-Speech Tagging of Language Corpora*", The Prague Bulletin of Mathematical Linguistics 78, 2002.

31. Kenneth Ward Church, "*Current Practice in Part of Speech Tagging and Suggestions for the Future*", in Simmons (ed.), Abornik praci: In Honor of Henry Kucera, Michigan Slavic Studies, pp. 13-48, 1992.

32. S. Klein and R. F. Simmons, "A Computational Approach to Grammatical Coding of English Words", JACM 10334-47, 1963.

33. Levent Altunyurt & Zihni Orhan, "*Part of Speech Tagger for Turkish*", Bogaziçi University, June 2006.

34. M. Civit and M.A. Martí, "*Design Principles for a Spanish Treebank*" In Proceedings of the Conference: Treebanks and Linguistic Theories. Sozopol, Bulgaria, September 2002.

35. M. Marchesi, N. Benvenuto, G. Orlandi, F. Piazza, A. Uncini, "*Design of Multi-Layer Neural Networks with Power-of-Two Weights*", Proceedings of ISCAS-90, IEEE

International Symposium on Circuit and Systems New Orleans (LA-USA), pp 2951-2954, May 1990.

36. Q. Ma and H. Isahara, "*A Multi-Neuro Tagger Using Variable Lengths of Contexts*", Proceedings of COLINGACL'98, Montreal, pp. 802-806, 1998.
37. Mao Yonghang, "*Natural Language Processing Model* (part of Speech Tagging and Sentences parsing)", Laboratory manual, 1997.
38. Megyesi Beata., "*Brill's Tagger Rule-based for Hungarian*", Course in Computational Linguistic Department of Linguistics Stockholm University, 1998.
39. Megyesi B., 1999b. "*Brill's POS Tagger with Extended Lexical Templates for Hungarian*", Workshop (W01) on Machine Learning in Human Language Technology", ACAI'99, Chania, Crete, July 5 - July 16, 1999.
40. B. Merialdo, "*Tagging English Text With a Probabilistic Model*", Computational Linguistics, 20(2), 155–172, 1994.
41. Mersehaizen Wolde Kirkos, "*የአማርኛ ሰዋስው*", Berihanena Selam Printing Press, Addis Ababa, 1934.
42. Mesfin Getachew, "*Automatic Part Of Speech Tagging for Amharic Language*: An Experiment Using HMM Approach", Master's Thesis AAU, 2001.
43. Michele BANKO and Robert C. MOORE, "*Part of Speech Tagging in Context*", In Proceedings of the 20th international conference on Computational Linguistics, Geneva, Switzerland, pp. 556-561, 2004.
44. Mitchell Tom, "*Machine learning*", The McGraw-Hill Companies Inc. New York, 1997.

45. M. Nakumura, K. Maruyama, T. Kawabata, and K. Shikano, "*Neural Network Approach to Word Category Prediction for English Texts*", In H. Karlghn Ed., COLING -90, Helsinki Universty, pp.213-218, 1990.

46. Qing Ma, Masaki Murato, Kiyotaka Uchimoto, and Hitoshi Isaharc, "*Hybrid Neuro and Rule-Based Part of Speech Taggers*", Communication research, Laboratory, Japan, 2000.

47. Quinlan J., "*Programs for machine Learning*", San Mateo, CA Morgan Kaufmann, 1993.

48. Saba Amsalu & Sisay Fissaha Adafre, "*Machine Translation For Amharic: Where We Are*", LREC-2006: Fifth International Conference on Language Resources and Evaluation, Genoa, Italy: pp.47-50, 23, May 2006.

49. Schmid H. "*Part-of-Speech Tagging With Neural Networks*", Proc. COLING'94, Kyoto, Japan, pp. 172-176, 1994.

50. Sisay Fissaha Adafre and J. Haller, "Application of Corpus-Based Techniques to Amharic Texts", MT Summit IX Workshop Machine Translation for Semitic Languages: Issues and Approaches, 2003.

51. Sisay fissaha. , "Part *of Speech Tagging for Amharic Using Conditional Random Fields",* Proceeding of the ACL 2005 Workshop on Computational Approaches to Semantic Languages: pages 47-54, June 2005.

52. T. Nakagawa, T. Kudoh, and Y. Matsumoto, "Unknown Word Guessing and Part-of-Speech Tagging Using Support Vector Machines", In Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium, 2001.

53. Tilli-Guiassa Yamina, "*Hybrid Method for Tagging Arabic Text*", Journal of Computer Science 2(3), pp. 245-248, 2006.

54. Yamina Tlili-Guiassa, "*Hybrid Method for Tagging Arabic Text*", Journal of Computer Science 2 (3): pp. 245-248, 2006.

55. Yenwondim Biadge, "*Application of Multi Layer Perceptron Neural for Tagging Part of Speech For Amharic Language*", Master's Thesis, AAU, 2006.

56. Guilder Linda V., "*Automated Part Of Speech Tagging: A Brief Overview*", At: Vanguill @ gusun.Georgetown.edu, 1995, *last visited April 2007.*

57. Hladka Barbora and Ribarov Kiril, "*Part Of Speech Tags for Automatic Tagging and Syntatic Structures*", http://ufal.ff.cuni.cz/Czech_tagging/haldka Ribarov/1998, *last visited April on 15, 2007.*

58. *"What are Artificial Neural Networks*?", http://Koti.mbnet.fi/neural Networks, Last visited June 25,2008.

59. Christos Stergiou and Dimitrios Siganos, "*Neural Networks*", http://www.doc.ic.ac.uk/journal/vol4/cs11/ , *last visited on August 30, 2007.*

Declaration

I, the undersigned, declare that this thesis is my original work, has not been presented for a degree in any other university, and that all sources of materials for the thesis have been duly acknowledged.

SOLOMON ASRES KIDANU

This thesis has been submitted for examination with my approval as an advisor.

SEBSIBE HAILEMARIAM

Addis Ababa, Ethiopia

September, 2008