ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
Faculty of Informatics
Department of Information Science

APPLICATION OF MULTILAYER PERCEPTRON NEURAL NETWORK FOR TAGGING PARTS OF SPEECH FOR AMHARIC LANGUAGE

BY

YENEWONDIM BIADGIE

2006
Addis Ababa

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
Faculty of Informatics
Department of Information Science

APLICATION OF MULTILAYER PERCEPTRON NEURAL NETWORK FOR TAGGING PARTS OF SPEECH FOR AMHARIC LANGUAGE

BY
YENEWONDIM BIADGIE

Name and Signature of Member of the Examining Board

Ato Workshet Lamenew , Chairman, Examining Board		
Prof. B.R.K. Rao, Advisor		
Prof. Deva Prasad Roy, External Examiner		

_____	Matthew Jennet	15/03/06
Chairman, Faculty	Signature	Date
_____	_____	_____
Chairman, Graduate Council	Signature	Date

ACKNOLEDGMENT

First and foremost, my thanks go to my advisor prof.B.R.K.Rao who helped me by giving his enlightening ideas and comments. My special thanks goes to Dr.Girma Awgchew who never hesitated to help me in any circumstances. I also wish to express my gratitude to Ato Yaregal Assabie, Zelalem Demlew, Mesfin Alamraw, Sebsebe Haylemariam, and Abeba Zeleke for their invaluable help.

Finally, I would like to thank all beloved friends whose encouragement, help and support in any aspect was very important which I could not forget.

DEDICATED

To

MY FATHER: BIADGIE SINSHAW

TABLE OF CONTENTS PAGE

CHAPTER ONE: INTRODUCTION

1.1 Background

Natural language is one of the fundamental aspects of human behavior and is a crucial component of our lives. In written form it serves as a long-term record of knowledge from one generation to the next. In spoken form it serves as our primary means of coordinating our day-to-day behavior with others (Allen, 1995).

The rapid growth of scientific and technical advances produced a scientific and technical revolution in 1990s. The most visible manifestation of this revolution was the phenomenon of information explosion, referring to the exponential growth of scientific publications and information records of all kinds (Saracevic, 1999). The difficulty of accessing the right information at the right time has been increased constantly starting from this event.

In order to fix this ever-worsening problem of information explosion, many fields are in the race. Automatic natural language processing using the emerging computing technology is one of them. Since most of the human knowledge is recorded in the form of natural language texts and utterance, only computer systems that have the ability to understand natural language can access this huge store of knowledge. In addition, natural language interfaces to computers allow effective human-machine communication so that complex systems can be accessible to every user. In other words a well-developed ability to handle natural language would revolutionize the way computers are used.

These facts are the major motivations for scientists and professionals in many fields to conduct research in automatic natural language processing (Allen, 1995).

Hence, the ultimate goal of natural language processing (NLP) is to design and build software products that will analyze, understand and generate languages that humans use naturally, so that eventually we will be able to address our computers as though we were addressing another person (Grishman, 1986). In short, by understanding natural language processes in terms of procedural and formal languages, we can give computer systems the ability to understand, generate and interpret natural language.

However, developing natural language understanding (NLU) systems is not an easy task. One major reason is that natural languages are ambiguous in their nature, i.e. there are words and sentences that are subjected to more than one linguistic structural interpretation (Jurafskg and Martin, 2000).

According to Megyesi (2002), ambiguity can occur at word level and at sense level. The two types of ambiguity that can occur at word level are *lexical category ambiguity* and *lexical semantics ambiguity*. Categorical ambiguity occurs when a word in a lexicon has more than one word category (part-of-speech) interpretations. For example, the English word "can" is categorically ambiguous between a verb and a noun in a lexicon.

Lexical semantic ambiguity arises when a word has more than one sense (meaning) in a lexicon. For instance, the Amharic word ***gena*** has two senses (meanings) in Amharic lexicon. As an adverb it refers to "future time" and as a noun it refers to "Christmas". On the other hand, the two types of ambiguity at sentence level are structural ambiguity and semantic ambiguity.

Structural or Syntactic ambiguity can occur when the grammar assigns more than one possible structure to a sentence. For example, the English sentence "***Rice flies like sand***" is ambiguous between two structures as shown below.

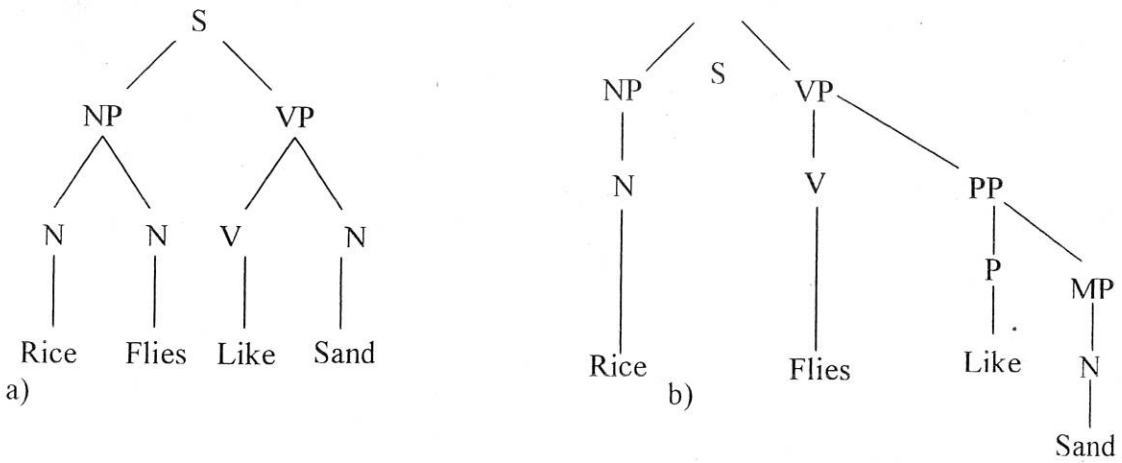


Figure 1 Two structural representations of the English sentence ***Rice flies like sand***.

In the first reading (a), the sentence is formed from a noun phrase (NP) describing a type of fly and a verb phrase (VP) that these flies like sand. In the second structure (b),

the sentence is formed from a noun phrase (NP) describing a type of substance called rice and a verb phrase (VP) stating that this substance flies like sand (say, if you throw it). The two structures also identify the part of speech for each word by analyzing the structure of Noun phrase and verb phrase more deeply (Allen, 1995:14).

The above type of ambiguities would inhibit NLU systems from making the appropriate inferences needed to model natural language understanding.

While people often resolve these ambiguities with out even being conscious of their existence in a sentence or text, computers lack this ability and computer programs must explicitly consider them one by one. People can do so because they have knowledge about the world and the language itself. People have, for example, lexical knowledge about what words mean, knowledge about syntactic structure, and general knowledge about the world. On the other hand, computers lack both our knowledge of the world and our experience with linguistic structures.

Although NLU by computers is a complicated problem, there are various approaches under investigation and some are succeeding to some extent in resolving the problem of ambiguity at different levels of language analysis (Megyesi, 2002).

For instance, *parsers* that resolve the structural ambiguity, *word-sense disambiguators* that resolve lexical semantic ambiguity and *part-of-speech taggers* that resolve lexical category ambiguity have been developed very well for popular languages like English. But the same story is not true for Amharic language. Thus, it is the purpose of this study to explore the possibility of developing an automatic part-of-speech tagger using ANN for Amharic language. A **part-of-speech tagger** (POST) is a system, which automatically assigns each word in a sentence a part-of-speech (Word class) from a pre-defined tag set. The format of a tagged sentence may be given in the form of **word/tag** followed by a single space between words as shown in the following example.

For instance, the English sentence "*I can light a fire and you can open a can of beans*" is untagged sentence. But the sentence I/PN can/VB light/VB a/DT fire/NN and/CN you/PN can/VB open/VB a/DT can/NN of/PP beans/NN./DL is a tagged sentence, where PN, NN, VB, DT, CN,PP and DL are Pronoun, Noun, Verb, Determiner, Conjunction, Preposition and Delimiter. Here every word is followed by a forward slash and a part of speech symbol (tag) taken from a predefined tag set.

Large text corpora, which are annotated with part-of-speech information, are useful in many areas of natural language processing applications. The following are some application areas of POS tagger.

1. POS tagging system is crucial for word-sense disambiguation (Lexical semantics ambiguity resolution). 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. For instance, the Amharic word ***gena*** has two senses (meanings) in Amharic lexicon. As an adverb it refers to "future time" and as a noun it refers to the "Christmas". If the output of a POS tagger is an input to a ward-sense disambiguator, the accuracy of word sense disambiguator increases.

2. A word's POS is also used to improve the level of precision in **Word-based indexing and retrieval systems** by disambiguating search terms. In these systems, the meaning of documents resides only in the words that are contained within them. For example, the sentences **"I see what I eat and I eat what I see"** are precisely the same in their sense (meaning) in the case of Word-based indexing and retrieval systems .In other words, the ordering of words that make the sentences play no role in determining their sense (meaning). Since they ignore syntactic information, these retrieval systems are called word-based (bag of words) systems. If the output of a POS tagger is an input to Word-based indexing and retrieval systems, their level of precision increases.

3. A Word's part-of-speech can tell us something about how the word is pronounced. For example, the word "content" can be a noun or an adjective. They are pronounced differently (the noun is pronounced as **CONtent** and the adjective as **conTENT**). Thus knowing the POS can produce more natural pronunciation in speech recognition and speech synthesis systems i.e. in order to pronounce a word correctly, a speech recognition system must know whether the word is being used as a noun or verb.

4. Parts-of-speech can also be used for stemming in information retrieval systems, since knowing a word's parts of speech can help tell us which morphological affixes it can take. For example in Amharic language, if a plural noun marker (tag) is attached to a word, then the suffix_**OC** can be stemmed.

5. Part-of-speech tagging may be the first process of a more full-fledged syntactic analysis usually involving a grammar-based parser. Parsing is the assignment of a syntactic description of a sentence. For instance, the English sentence "Rice flies like sand" is ambiguous between two structures as shown in Figure 1.1. If this sentence is tagged as Rice/N flies/N like/V sand/N, then the sentence parser will have the first structure (a). If this sentence is tagged as Rice/N flies/V like/P sand/N, then the parser will have the second structure (b). Thus, before a sentence is parsed syntactically, the sentence must be annotated with POS at word level. Then the sentence parser can take the out puts of a tagger as its input so that the sentence parser parses the sentence effectively and easily (Allen, 1995).

6. Part of speech tagging may be also primary step for semantic analysis in machine translation systems. The increasing use of the Internet intensifies the need to translate documents from foreign languages to local languages. To develop machine translation systems, a POS tagging system can be a component. Hence, the choice of the correct POS of a word may be crucial when translating to another language. For instance, if an adverb marker (tag) is attached to the Amharic word ***gena,*** it is translated in to *future time* and if a noun marker (tag) is attached to it, it is translated in to "*Christmas*".

7. POS tagger can also be a component for works related to spelling and grammar checking.

In short, POS tagger helps in parsing the sentence, which is in turn useful in various other natural language processing tasks such as machine translation, information retrieval, word sense disambiguation, phrase recognition, spelling checking etc. Thus, it is the purpose of this study to explore the possibility of developing part-of-speech tagger using Artificial Neural Network (ANN) for Amharic language.

1.2 Statement Of The Problem And Justification Of The Study.

According to Lesalu(1973) Amharic language has been the official language of the ruling class for centuries and still is the official language of the Federal Government of Ethiopia, and thus has a well established writing system and well standardized norms of spoken and written language. Amharic Language is also the widely spoken Semitic

5. Part-of-speech tagging may be the first process of a more full-fledged syntactic analysis usually involving a grammar-based parser. Parsing is the assignment of a syntactic description of a sentence. For instance, the English sentence "Rice flies like sand" is ambiguous between two structures as shown in Figure 1.1. If this sentence is tagged as Rice/N flies/N like/V sand/N, then the sentence parser will have the first structure (a). If this sentence is tagged as Rice/N flies/V like/P sand/N, then the parser will have the second structure (b). Thus, before a sentence is parsed syntactically, the sentence must be annotated with POS at word level. Then the sentence parser can take the out puts of a tagger as its input so that the sentence parser parses the sentence effectively and easily (Allen, 1995).

6. Part of speech tagging may be also primary step for semantic analysis in machine translation systems. The increasing use of the Internet intensifies the need to translate documents from foreign languages to local languages. To develop machine translation systems, a POS tagging system can be a component. Hence, the choice of the correct POS of a word may be crucial when translating to another language. For instance, if an adverb marker (tag) is attached to the Amharic word ***gena,*** it is translated in to *future time* and if a noun marker (tag) is attached to it, it is translated in to "*Christmas*".

7. POS tagger can also be a component for works related to spelling and grammar checking.

In short, POS tagger helps in parsing the sentence, which is in turn useful in various other natural language processing tasks such as machine translation, information retrieval, word sense disambiguation, phrase recognition, spelling checking etc. Thus, it is the purpose of this study to explore the possibility of developing part-of-speech tagger using Artificial Neural Network (ANN) for Amharic language.

1.2 Statement Of The Problem And Justification Of The Study.

According to Lesalu(1973) Amharic language has been the official language of the ruling class for centuries and still is the official language of the Federal Government of Ethiopia, and thus has a well established writing system and well standardized norms of spoken and written language. Amharic Language is also the widely spoken Semitic

language next to Arabic and Hebrew. It is used in word processing activities in different offices. This in turn has resulted in having a large amount of databases and repositories in electronic form. Moreover, there are also a large number of printed Amharic texts that are circulating among governmental, non-governmental and private sectors.

With the increasing availability of large-scale linguistic data (corpora) in electronic format, affordable computer resources and development in machine learning technology, corpus based NLP tasks for popular languages (like French, English) have been well studied with satisfactory achievements. So, Morphological analyzers, morphological synthesizers, part-of-speech taggers, phrase recognizers, indexers, stemmers, speech recognizers, speech synthesizer, machine translators, grammar and spelling checkers, etc have been developed very well. However, the same story is not true for Amharic language. This language is among the very few languages of the world being incorporated with the computer. In other words, corpus-based NLP tasks for Amharic language are at a deadlock state.

Studies in the development of different NLP tools for analyzing Amharic text have been started recently. Among the attempts made in this area are the development of Amharic word processing (Daniel and Yonnas, 1994), a stemming algorithm (Nega, 1999), word parser (Abiyot, 2000), sentence parser (Atelach, 2003), morphological analyzer (Tesfaye, 2002), Morphological synthesizer (Kubur, 2002), and part-of-speech tagger (Mesfin, 2001).

Mesfin (2001) conducted research to develop automatic part-of-speech tagger for Amharic language using stochastic (Probabilistic) Hidden Markov Model (HMM). Although corpus- based methods for part-of-speech tagging have been probabilistic to a great extent, recently there has been a great increase in the development of other types of corpus-based systems, especially within the machine learning community.. Artificial Neural Network (ANN) is one of them. Thus, a study with other approaches like ANN opens an alternative way of carrying the task of POS tagging for the following reasons:

1.One major disadvantage of rule-based and stochastic approach is that the inherent inability to deal with unknown words, i.e. words that are not part of the training set. But neural network based approaches not only learn the association (Word-to-tagg mapping)

from a representative training data set but can also generalize to unseen exemplars (Ma, 2002;Ahmed, 2002).

2. Stochastic approaches use only a small amount of information in the tagging process (e.g. the bi-gram method only looks at the parts-of-speech of the preceding word). But much more information is made available to the neural network program than bi-gram. Specifically, for each word for which a part-of-speech classification strategy is being learned, the training data will include the preceding and succeeding words.

Mesfin (2003), Atelach (2003) and Lars (2003) also suggested Artificial neural network approach initially, decision tree and constraint grammar using inductive logic programming secondly to develop part-of-speech tagger (POST) for Amharic language. Therefore, the current study is basically initiated to explore the potential application of ANN approach for developing Automatic POST for Amharic text.

1.3. Objective of The Study

1.3.1 General Objective

The general objective of this study is to explore the potential application of Artificial neural network (ANN) approach for developing POS tagger model for Amharic texts.

1.3.2 Specific Objectives

With the aim of achieving the above general objective, the study has attempted to address the following specific objectives.

- To review the application of artificial neural network in general and for part of speech tagging problem in particular.
- To review the word categories (part of speeches) of the Amharic language to identify part of speech tag set.
- To study the morphological property of Amharic words to identify the inflectional and derivational rules of word formation that are useful to develop part of speech tag set.
- To examine the type of lexicons required for neoro-part of speech tagger, and design the lexicons accordingly.

- To assess the various approaches (techniques) suggested for the development of neuro-tagger and select the convenient approach for this study.
- To prepare manually tagged training and testing data sets to be used in the experiment.
- To design or adopt machine learning algorithm for the development of neoro-tagger
- To draw conclusions based on experimental results.

1.4 Research Methodology

The following methods were employed in order to achieve the above stated objectives.

1.4.1 Literature Review: A number of resources including books, research reports, journal articles, manuals, and other published and unpublished documents (including those from the internet) have been used for the following purposes:

- ❖ To comprehend the word categories (part of speeches) of the Amharic language.
- ❖ To identify the various approaches suggested for the development of neuro-tagger and to select the convenient approach to the current research work

1.4.2 Discussion: Discussions with linguistic experts were made to understand the language and to get suggestions that are invaluable for this study.

1.4.3 Data Preparation and Database Design

In this study **Artificial Neural Network** approach that employs a supervised learning mechanism is selected to buid a neural network model for part of speech tagger. This approach requires a large data to function properly. Since such data is not available for Amharic language, manually tagged corpus consisting of 159 sentences was prepared. These sentences were randomly divided in to two sets. One with 136 sentences for training and the other with 23-sentences for testing. From this corpus a database consisting of three tables was prepared as a knowledge base for neuro-tagger. Microsoft Access has been used for the database design. The tables are Full form lexicon Table, Training Table and

Testing Table. The full form lexicon contains distinct words of the corpus with their corresponding lexical probabilities. The Training Table contains words for training and the Testing Table contains words for testing.

Algorithms are designed to generate training data (labeled as 1_T_0-TRN, 2_T_0_TRN, 1_T_1_TRN, 2_T_1_TRN and 2_T_2_TRN) that contain words with their corresponding features (input) and target tag type (output) for neural network. Testing data (labeled as 1_T_0-TST, 2_T_0_TST, 1_T_1_TST, 2_T_1_TST and 2_T_2_TST) are also generated that contain words with their corresponding features (input) but with out their corresponding target tag type (output) for neural network. **Microsoft Visual Basic version 6.0** programming language has been used to generate the inputs of neural network. This language is chosen for its user –friendly functionalities.

1.4.4.Neural Network Model Building

Neural Network model for part of speech tagger (MLP-tagger) has been developed using Brain Maker Neural Network application software. The reason of selecting this software is availability.

1.4.5 Training and Testing Procedures

In this study, supervised learning approach (MLP) was used to build a neural network model for part of speech tagger. The model developed depends on localized contextual features (patterns) of a words to identify their corresponding tags (labels). The quality of the outputs found is then tested by examining whether the model produce correct tags (labels).

For this purpose, a table consisting of 2429 words with their corresponding localized contextual features (input patterns) and target tag types (output pattern) for training and another table consisting of 392 words with their corresponding features but with out their target tag type were prepared. Based training, a number of models have been built. The testing data has been submitted to these models in order to select the best model. Here, **Microsoft Visual C++ programming** language has been used to activate the **trained BrainMaker Neural Network**

software. The performance of models is presented in percentage and tag-wise analysis form. Accordingly, the results from the selected model have been reported in sections **6.2 and 6.3** of chapter six.

1.5 Application Of Results And Beneficiaries

In order to survive in the information age, building foundation resources (such as balanced corpus, annotated corpus, basic lexical data base, morphological analyzer, parser, part of speech tagger) of Amharic language is the basic issue.

Hence, developing automatic part of speech tagger for Amharic language (one of the basic resources) and making it reusable by other high level applications (such as sentence parsing, phrase recognition, machine translation, information retrieval, grammar and spelling checker, generation of intonations in speech production systems) would be the major benefit of the study. Thus, researchers who are involved or who wants to be involved in increasing a computer's capability to analyze Amharic language at different level of analysis (parsing, information retrieval, word sense disambiguation, machine translation etc) are the beneficiaries of the results of this study.

1.6 Scope of The Study

The scope of this thesis is delimited to explore the potential application of Artificial neural network (ANN) approach for developing POS tagger model for Amharic texts.

1.7 Limitations of The Study.

As described earlier, MLP neural network model for part of speech tagger is developed. The neural network should have been trained and tested on huge data annotated with part of speech tags (labels) manually to see its actual performance. Unfortunately this was not done due to the unavailability of sufficient number of words and their target tag type and also preparing such type of corpora is laborious, expensive and time consuming.

CHAPTER TWO
PART OF SPEECH TAGGING

2.1 Introduction

Every language enables its speakers to exchange their ideas. In order to exchange their ideas, human beings should combine phones (sounds) in to a word, words in to a phrase and phrases in to a sentence.

Words are the physically definable units which one encounters in a stretch of writing (bounded by spaces) or speech (where identification is more difficult, but where there may be phonological clues to identify boundaries, like pause). In this sense, word is referred to as orthographic word (for writing) or the phonological word (for speech). Since studying the characteristics of all words one by one is a difficult task, the most efficient method is dividing them in to different word classes (parts of speech) and then studying the behavior of each word class.

In a lexicon (dictionary), words are often ambiguous in their part-of-speech. But each word in a sentence has only one part of speech. For instance, in a lexicon the word **can** is ambiguous for its part of speech (becuase it belongs to both verb and noun classes). But in the sentence" *you can open a can of beans"*, the first **can** is a verb and the second **can** is a noun. Hence, the main goal of part-of speech tagging is to assign each word in the sentence (corpus) the contextually correct part-of-speech tag. For instance, the Amharic sentence "alemu gobez temari new (alemu is a cleaver student)'' is not a labeled sentence (it is untagged sentence). But the sentence "Alemu/NN gobez/AJ temari/NN new/AUX./DL" is a labeled (tagged) sentence, where NN, AJ, AUX and DL are tags for Noun, Adjective, Auxiliary and Delimiter respectively. Here every word is followed by a forward slash and a part of speech symbol (tag).

The special codes or labels, which are attached to each word in the corpus, are called **tags**. Other tokens other than words like punctuation marks, symbols, abbreviations, etc often also receive a tag or a label in the corpus.

In short, each token in the text is annotated with some morpho-syntactic category. The morpho-syntactic category contains information about the parts-of-speech the word

belongs to, given some context and might also contain other relevant information about the morphological structure. For instance, it is common that the inflectional categories of the words (like number, person, case, etc) also are marked.

Morpho-Syntactic categories can be different from one language to another language. They can also different in a given language depending on the particular task for which the annotated text will be used.

2.2. Precise Definition of Part-Of-Speech Tagging.

Suppose there is a lexicon **L**= (W_1, W_2, …W_n), in which the parts-of-speeches that can be served by each word are listed and there is a tag set **T**= (T_1, T_2, T_3, …T_k) in a certain language. The part of speech tagging problem is, thus, to find a string of tags **S**=T_1 T_2…T_m($T_i \in T$,i=1,2,3…m) when sentence **W**=w_1 w_2 …w_m($w_i \in L$,i=1,2,3….m) is given. Here, **m** is the number of words in a sentence **W**, **k** is the number of tags in the given language and **n** is number of distinct words in the lexicon. Now a brief description of corpus (plural corpora) and tag set is given below.

A computer corpus is a large body of machine -readable texts which are taken as a representative sample of a given language and used both in the development of natural language processing (NLP) software and in NLP application areas like speech recognition, information retrieval, machine translation etc.

A corpus can be annotated or unannotated depending on the algorithm we wish to use. Unannotated corpora contain raw texts. On the other hand, annotated corpora can involve a repository of several types of linguistic information. The raw text is analyzed and annotated at some linguistic level of analysis (word, phrase and sentence levels). Labels or tags are attached to segments (tokens) of the text and describe their properties or the relations between them. Thus, a tag can be defined as a code representing some feature or a set of features of the segments in the text and can there by convey single or complex linguistic information such as part-of-speech, morphology, syntax and semantic structures.

The tags build up a tag set and the size, content, and the complexity of the tag set is different from language to language. In short, unannotated corpora indicate linguistic

information implicitly while annotated corpora are enhanced with linguistic information explicitly.

A Corpus can be labeled or tagged manually or automatically. A manually tagged corpus is annotated by one or more language experts while an automatically tagged or labeled corpus is annotated by using one or more algorithms whose out put may or may not be corrected by human beings. Manual annotation of texts is expensive, time consuming and monotonous.

Besides these, manual annotation can result in inconsistency because different experts may annotate the same corpus differently so that it is difficult to detect and correct the errors. The numbers of errors in automatic tagging may be more than the numbers of errors in manual tagging. But the errors are consistent, easy to detect and correct.

Automatic POS tagging is a difficult task and the accuracy of the tagger is always less than **100%** because of the following two major problems:

- The first problem is the **disambiguation of homograph** (ambiguous) words. Natural languages contain a large number of ambiguous words in their lexicon. Ambiguous (homograph) words (in the context of POS tagging) are words that are spelled the same and are ambiguous between two or more classes, i.e. words that belongs to two more parts of speech. For example, the English word **"can"** is ambiguous between the parts of speech verb, and noun. This means that even if we have a lexicon with a list of all the words in the lexicon with their possible parts-of -speech or word class, this does not give enough information to assign the correct tag to a given word in the context of a sentence. Therefore, we must exploit the information given in the context of occurrence to choose the correct tag for words that have more than one possible word class. An approach to determine the context of appearance of the word is to consider its **immediate neighborhood words.**
- The second problem is the tagging of **unknown words**, i.e. words not found in the lexicon. Practically, we never have a lexicon with a complete list of all words with their possible parts-of-speech for a given language. If the part-of-speech tagger is supposed to work on unrestricted text, we must always expect to come across words that are not in the lexicon and for which we have no information

about their possible POS. In this case the tagger has to predict the tag based on the morphological structure of the word.

Thus, most part-of-speech taggers relay on **lexical, contextual, and morphological** information to handle unknown words, and to disambiguate known words having several possible classes (Nivere, 2000).

Lexical information (in the context of POS Tagging) is information about what word forms exist in a given language, and what part-of-speech are possible for a given word form. For instance, the English word **"still"** is a word form, which can occur with the tags verb (VB), noun (NN), adverb (ADV) in a lexicon. Thus, the word "still" in the lexicon must be stored in such a way that the dictionary provides all the three possible parts-of -speech information. On the other hand, *Contextual information* is information concerning the tendency for different parts-of-speech and /or word forms to occur together. In English language, determiners ("a" and "the") are not followed by finite verbs (like "runs") is an example of contextual information for word forms that occurs together.

Most part-of-Speech taggers relay on the assumption that a word can be assigned a single POS tag in a given sentence.i.e. among the possible parts-of-speech, at least one of them is the most likely. Thus, it is assumed that the **syntax** of the language is the best clue to resolve the problem of ambiguity (Nivere, 2000).

2.3. Methods For Automatic POS Tagging

Generally, there are two major methods in automatic natural language processing: *Rule-based and corpus-based (or machine learning) methods*. Before 1980s, rule-based systems were dominant in the fields of computational linguistics and Natural language processing. After 1980s, there has been a paradigm shift with in the field from rule-based approaches to corpus-based (or machine learning) approaches as researchers have access to faster, more efficient and powerful computers.

Rue-based systems involve a large database of hand-written disambiguation rules. The rules are constructed based on the structure of a language being processed. On the other hand, corpus-based or machine learning methods automatically learn and build a model for a particular NLP task with out a lot of human intervention or expert knowledge. The

generalizations are acquired from a corpus. In these approaches, complicated language structures can be learned first by defining some general model and then by applying some machine learning method to large amount of data. Similarly, there are two basic methodologies to develop automatic part-of-speech tagger: *rule-based part-of-speech tagging and corpus-based part-of-speech tagging.*

2.3.1 Rule-Based Parts-of -Speech Tagging.

Rule-based taggers involve a large database of hand-written rules (contextual and morphological rules). In other words, the rules may contain morphological, lexical and /or syntactic information of the language being tagged to tag **unknown and ambiguous** words (homographs).

Many of these systems work on a two-stage structure (Jurafky, 2000). In the first stage, these systems use their lexicon to assign each word a list of potential parts-of-speech. In the second stage, they use their large lists of hand-written disambiguation rules to assign a single part-of-speech for each word by eliminating all tags, which are inconsistent with the context. The contextual rules revise the tag of a word based on the surrounding words or the tags of the surrounding words. Assume the tag of a word **w** in English language is **N** (noun) and **V** (verb) in a lexicon.

In the first stage, it is tagged as **W/N** or **W/V**. In the second stage, the tag N (noun) is selected when the preceding word is tagged with the tag **TO** (infinitive). According to this rule, if the phrase "to draw" is tagged as" to /**To** draw/ **N**" in the first-stage, then it can be tagged as "to /**To** draw/**V**" in the second-stage. Such type of contextual rules change tags that are wrongly assigned to words by lexical information in the first stage.

In addition to contextual rules for changing one type of tag in to another to solve the problems of ambiguous words, there are also contextual rules to tag unknown words.

In English language, for instance, **DT +X +N= X/ADJ** is a rule which states that if unknown word **X** is preceded by a determiner **(DT)** and followed by a noun **(N),** then the unknown word **X** will be tagged with an adjective **(ADJ)**.

In addition to the above types of contextual rules, there are also rules of morphological analyzer to guess the category of unknown words .For instance, if the English word

"**cats**" is not in the lexicon, it will be analyzed morphologically as "**cat**" +"**s**" and it will be tagged as **NPL** (a code for plural noun).

2.3.2 Corpus-Based Part-of-Speech Tagging

As mentioned above, there has been paradigm shift with in the field of **NLP** from rule-based methods to corpus-based or machine learning methods as the power of computers increases. Nowadays, many systems are not purely rule-based systems.

Although corpus-based methods for morphological and syntactic analysis have been **probabilistic** to a great extent, recently there has been a great increase in the development of other types of corpus-based systems especially with in the **machine learning community.**

Machine learning is the subfield of Artificial Intelligence and computer science that studies the automated acquisition of domain-specific knowledge (Megyesi, 2002). Mitchell(1997) also stated that Machine learning is the ability of a machine to recognize patterns and acquire knowledge from the environment that have occurred repeatedly and improve its performance based on past experience. The goal of these systems is to improve their performance as the result of experience. This capacity to learn from experience and analytical observation results in a system that can continuously self-improve and thereby offer increased efficiency and effectiveness.

In short, the main goal of this field is to develop algorithms that are able to automatically induce a model from data, i.e.to extract and describe structural patterns and regularities in the data. The model can be built of rules, concept descriptions, trees etc. In order to build a model, a machine-learning algorithm needs a training data set which is representative for the particular learning task. Natural language processing systems are difficult to build, but machine-learning methods can help automate their construction significantly. Machine learning techniques can be used for automatic construction of semantic parsers, information extraction systems, syntactic parsers, machine translators, word-sense disambiguators, and morphological analyzers. Examples of machine learning algorithms (data-driven algorithms) that have been successfully applied to these tasks for popular languages (like English) are decision tree, inductive logic programming, neural network,

genetic algorithms, Hidden Markov Model (HMM), memory-based learning, maximum entropy learning etc (Mitchell, 1997). In this thesis, Artificial Neural Network learning algorithm will be applied. This approach will be discussed in more detail in chapter three.

CHAPTER THREE: ARTIFICIAL NEURAL NETWORK

3.1. Introduction

Conventional computing programs are fast in arithmetic operation. They do precisely what the programmer programs them to do. They follow a set of instructions or rules in order to solve a problem. The way to solve a problem must be known and stated in small unambiguous instructions. Unless the specific steps that the computer needs to follow are known, the computer cannot solve the problem. This traditional programming approach restricts the problem solving capability of computers to problems that we already understand and know how to solve. But computers would be so much more useful if they solve problems that we do not exactly know how to do (Haykin, 1999).

It has been recognized that the human brain computes in an entirely different way from the conventional computers. The brain has the capability to organize its structural constituents, known as neurons, so as to perform computations (like patter recognition, perception and motor control) many times faster than the fastest digital computers in existence today. Even simple animal brains are capable of functions that are currently impossible for conventional programs. These programs are poor in recognizing even simple patterns and in generalizing those patterns of the past in to actions of the future.

In order to address problems that are often very complex for conventional programs to solve, **Biological Neural Networks (BNNs)** have been modeled artificially. Since the motivation of *Artificial Neural Network (ANN)* arise from the computational and pattern analysis of the biological neural network, it is important to have an insight about how a biological neuron network learns to understand ANN (Mitchell, 1997).

3.2. Natural/Biological Neural Network (BNN)

The exact working of the human brain is still a mystery, i.e. much is still unknown about how the brain itself learns to process information. Yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body cell, does not appear to regenerate. Because this type of cell is the only part of the body that is not slowly replaced, it is

assumed that these cells are what provide us with our abilities to remember, think, and apply previous experience to our every action. These cells are known as *neurons.*

It is known that the neuron, or nerve cell, is the fundamental unit of all nervous system tissue, including the brain. It is estimated that the human brain contains over 100 million nerve cells. In the brain, there are many variations on this basic type of neuron, further complicating the man's attempt at electrically replicating the process of thinking .Yet, all natural neurons have the same four basic components as shown in figure 3.1

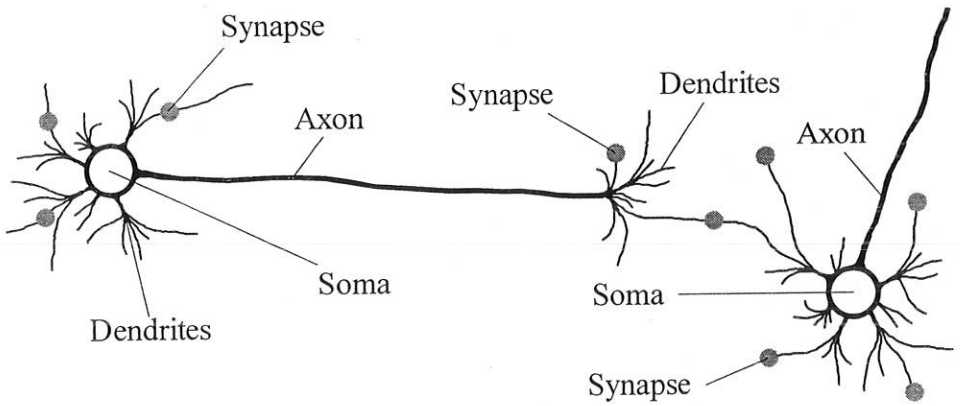


Figure 3.1.The structure of a natural/biological neuron

Each neuron consists of a **cell body, or soma** that contains a **cell nucleus**. Branching out from the cell body are a number of fiber like structures called **dendrites**, and a single long fiber called the **axon.** Dendrites branch into a bushy network around the cell, where the axon stretches out for a long distance. Eventually, the axon also branches in to strands and substrands that connect to dendrites and cell bodies of other neurons. The connecting junction is called **synapse**. Each neuron forms synapses with thousands of other neurons, i.e. number of connections or synapse per neuron is **10,000**. Signals are propagated from neuron to neuron by a complicated electrochemical reaction. **Chemical** transmitter substances are released from synapses and enter the dendrites (input channels), raising or lowering the electrical potential of the cell body (soma). The soma then combines these incoming signals in some way and performs a general non-linear operation on the result. When the potential reaches a threshold, an electrical pulse or action potential is sent down the axon. The pulse spreads out along the branches of the axon, eventually reaching

synapses and releasing transmitters in to the body of other cells. Synapses that increase the potential are called **excitatory,** and those that decrease it are called **inhibitory**. Perhaps the most significant finding is that the strength of synaptic connections can be changed in response to the pattern of stimulation. These mechanisms are taught to perform the basis for learning in the bran.

Currently, ANN researchers are seeking an understanding of nature's capabilities for which people can design solutions to problems that have not been solved by traditional computing. To do this the basic units of ANN (the artificial neurons) simulate the above four basic functions of natural neurons (Haykin, 1999).

3.3 Artificial Neural Network

Artificial Neural Networks are statistical models of real world systems comprising of large number of natural neuron-like processing elements and a large number of weighed connections between the elements (Haykin, 1999). They resemble the brain in two respects. First, the network acquires knowledge from its environment through a learning process. Second, inter-neuron connection strengths, known as synaptic weights, are used to store the acquired knowledge, i.e. the weights on the connections encode the knowledge of a network.

The "electrical" information is simulated with specific values stored on those weights that make these networks have the capacity to learn, memorize and create relationships among data. Learning in biological systems involves adjustments to the strength of synaptic connections that exist between the neurons so that the influence of one neuron on another changes. The connectivity of the brain is continually changing as an organism learns different functional tasks. This is true of **ANNs** as well. **ANNs** take a different approach to problem solving than that of conventional computers. Rather than using programs that are written as series of instructions, as do all traditional computers, ANNs are trained with a set of training examples. The network is then able to "learn" from the initial examples to respond to information sets that it has never encountered before. This feature of ANN is called learning capability of the system.

Generally, a very important feature of these networks is their adaptive nature where "*learning by example*' replaces "*programming*" to solve problems. These features make these computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved, but where training data are available.

Recently these techniques have begun to emerge as a novel approach for the modeling of various complex and non-linear phenomena and the **multiplayer feed-forward network** with **back propagation** algorithm is designed to function well in such a non-linear phenomena.

3.4 Mathematical Model For Different Types Of Artificial Units (Neurons)

3.4.1 Mathematical Model For A Single Perceptron Unit (Treshold Unit, Step Unit).

One type of ANN system is based on a unit called **a perceptron**, illustrated in Figure **3.2.**A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputa **1** if the result is greater than some treshold and **-1** otherwise.

Mathematically, given inputs x_1 through x_n, the output $O(x_1,x_2,x_3,\ldots,x_n)$ computed by the perceptron unit is :

$$O(x_1,x_2,x_3,\ldots,x_n)=\begin{cases} 1 & \text{if } W_0+x_1w_1+x_2w_2+\ldots+x_nw_n > 0 \\ -1 & \text{if } W_0+x_1w_1+x_2w_2+\ldots+x_nw_n \leq 0 \end{cases} \text{----------(3.1)}$$

Where each w_i is a real valued constant (weight) that detemines the contribution of input x_i to the perceptron out put.

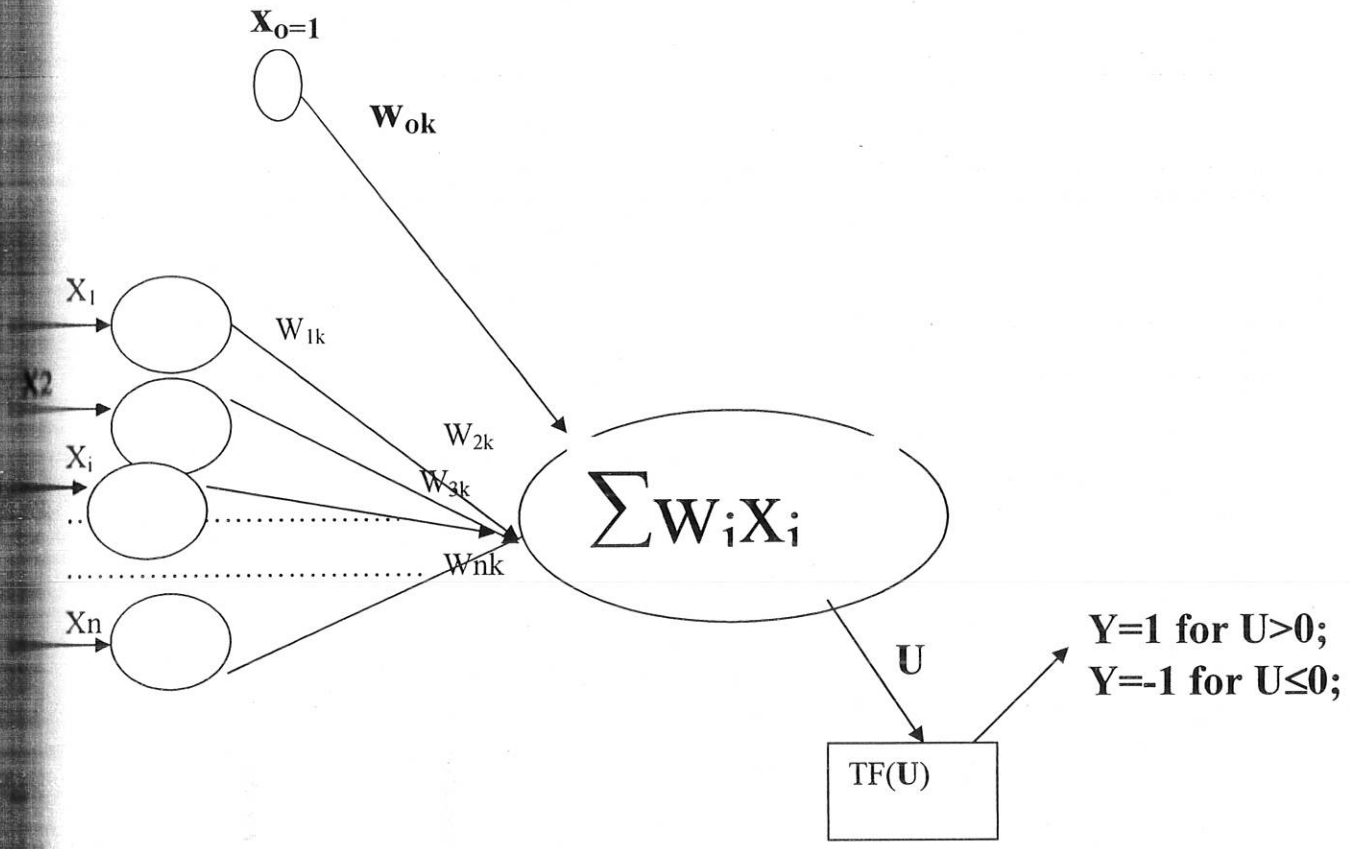


Figure 3.2. Computations Within A Single Perceptron Unit

As shown in figure 3.2, $X=(X_1, X_2 \ldots X_n)$ is the input vector with its corresponding weight vector $W=(w_1, w_{2k}, \ldots w_n)$ to the out put node. The bias W_0 serves to allow one to change bias of the output independently of the inputs. The summing function, shown by the summation $\sum$ sign, adds the dot product of the inputs x_i and the corresponding weights w_i, the result of the sum is shown as U ($U=\sum x_i\ w_i$) and passed to transfer function indicated by **TF (U)=Y and Y =1 if U > 0. Otherwise -1**.

3.4.2 Mathematical model for a Single Linear Unit

For a linear unit the output $O(X_1,X_2,X_3 \ldots X_n)$ is given by :

$$O(X_1,X_2,X_3 \ldots X_n) = W_0 + x_1w_1 + x_2w_2 + \ldots + x_nw_n = U = Y \text{----------} (3.2).$$

Thus, a linear unit corresponds to the first stage of a perceptron, without the treshold.

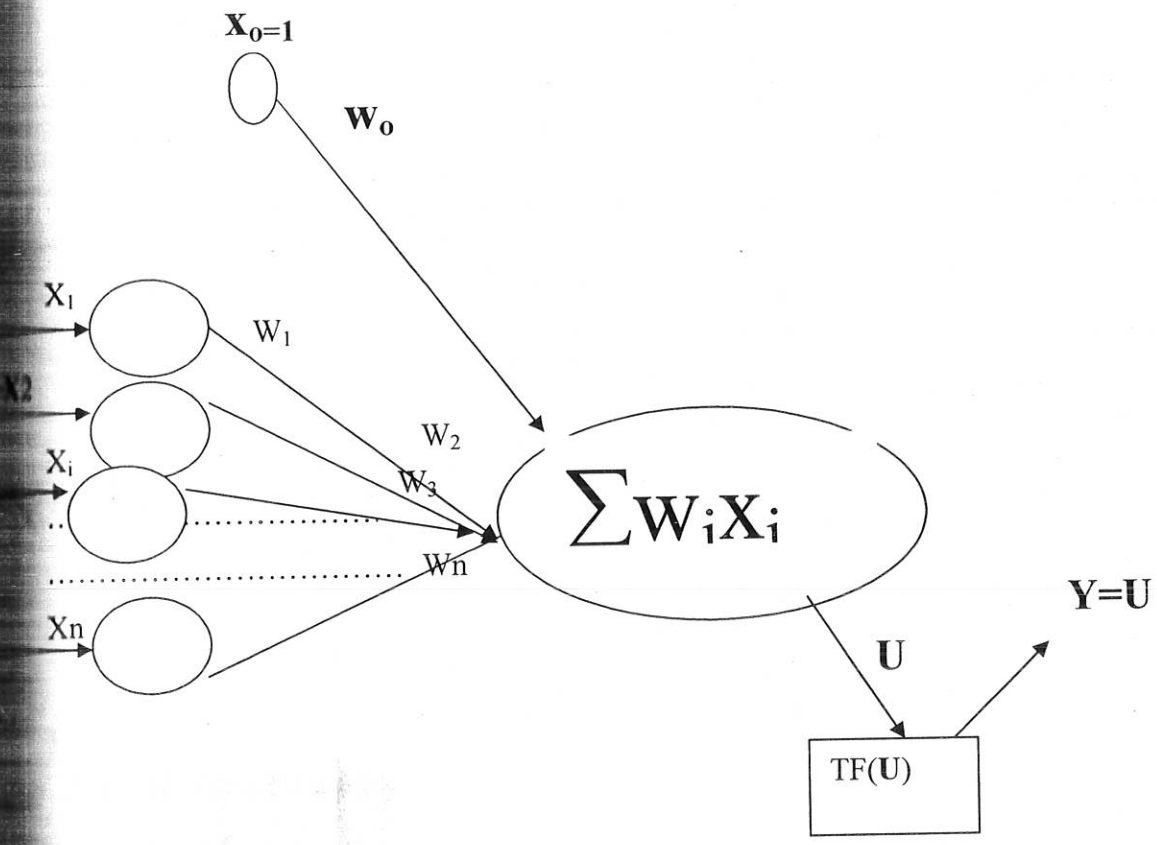


Figure 3.3. Computations Within A Single Linear Unit

3.4.3 Mathematical Model For A Single Sigmoid Unit (A Differentiable Unit).

Like the perceptron, the segmoid unit first computes a linear combination of its inputs, and then applies a threshold to the result. However, in the case of the sigmoid unit the treshold output is a continous function of its inputs. Mthematically, the sigmoid unit computes its output O as:

$$O(x_1, x_2, x_3 \ldots x_n) = W_0 + x_1 w_1 + x_2 w_2 + \ldots + x_n w_n = U \text{ and}$$

$$TF(U) = Y = 1/1 + e^{-U} \text{----------(3.3)}$$

Thus, a linear unit corresponds to the first stage of a perceptron, without the treshold.

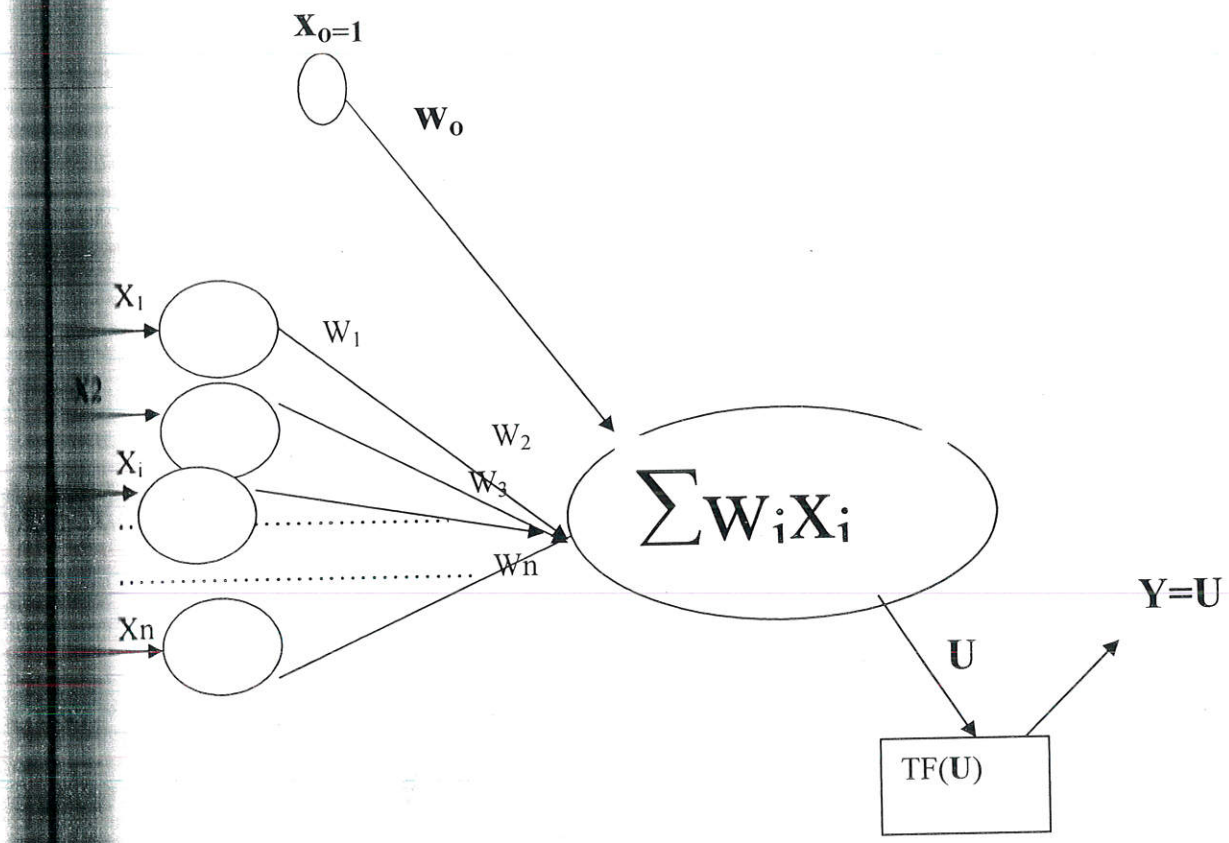


Figure 3.3. Computations Within A Single Linear Unit

3.4.3 Mathematical Model For A Single Sigmoid Unit (A Differentiable Unit).

Like the perceptron, the segmoid unit first computes a linear combination of its inputs, and then applies a threshold to the result. However, in the case of the sigmoid unit the treshold output is a continous function of its inputs. Mthematically, the sigmoid unit computes its output O as:

$$O(x_1, x_2, x_3 \ldots x_n) = W_0 + x_1 w_1 + x_2 w_2 + \ldots + x_n w_n = U \text{ and}$$

$$TF(U) = Y = 1/1 + e^{-U} \text{----------(3.3)}$$

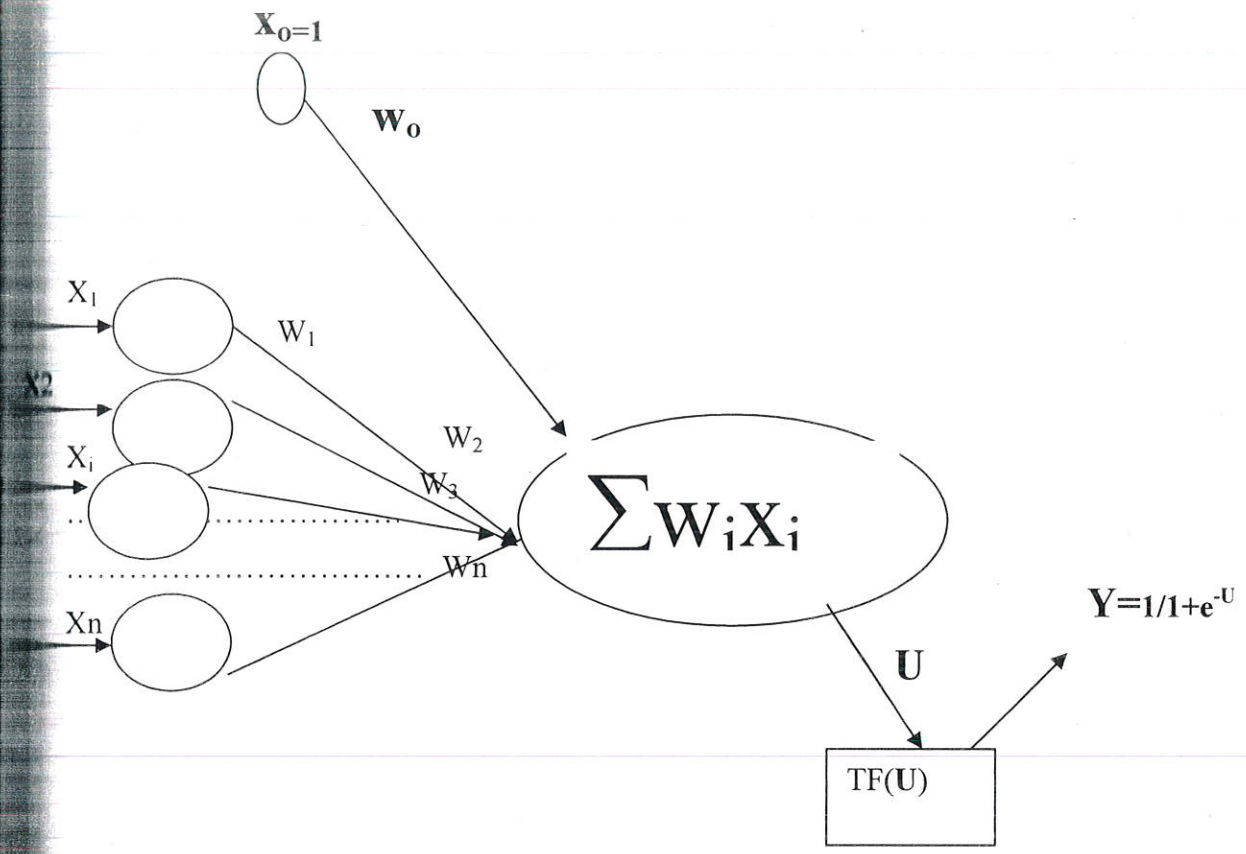


Figure 3.4. Computations Within A Single Sigmiod Unit

Generally, the following basic types of transfer functions are identified.

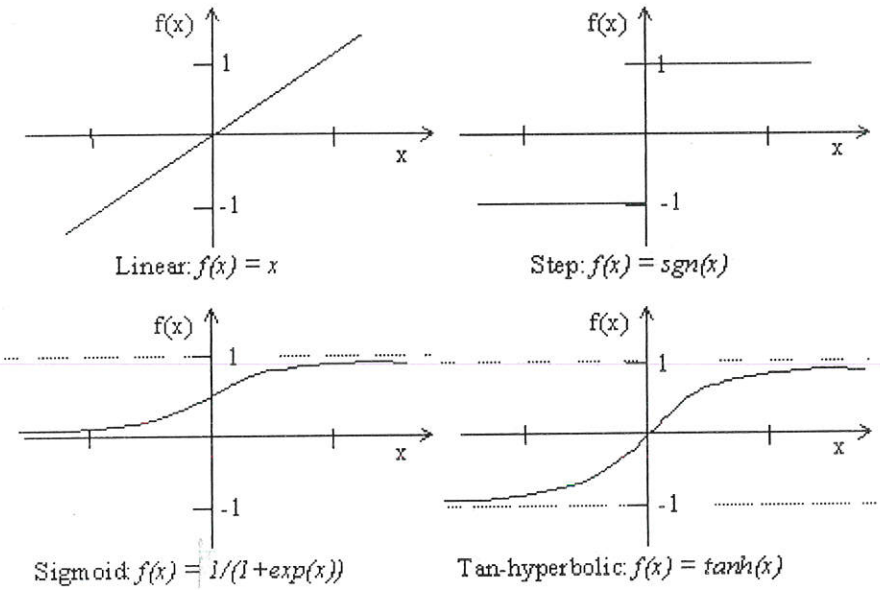


Figure 3.5. Transfer Function

3.5 When Artificial Neural Networks are used?

fArtificial Neural Networks (ANNs), with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an 'expert' in the category of information it has been given to analyze. It can respond to the input patterns given to it after training. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

There are some sets of real world problems that can robustly be solved by ANNs. These problems are kinds with data corresponding to noisy, complex, and subject to high degree of distortion. According to Mitchell (1997), ANN is appropriate for problems when:

- Instances are represented by many attribute value pairs. The target function to be learned can be defined over instances that can be described by a vector of predefined features. These input values may be highly correlated or independent of one another. Input values can be any real values.
- The target function out put may be discrete-valued, real-valued or a vector of several real or discrete valued attributes.
- Training as well as test examples may contain errors or noisy data.
- Long training times are acceptable.
- Fast evaluation of the trained or learned target function may be required.
- Understanding the learning and decision function of the target by human is not important.

In summary, ANNs offer a completely different way to analyze data, and to recognize patterns within that data, than traditional computing methods. They try to provide a tool that both programs itself and learns on its own. ANNs are structured to provide the capability to solve problems without the benefits of an expert and without the need of programming. They can seek patterns in data that no one knows are there. Yet, despite the advantages of neural networks over traditional computing in some specific areas, neural nets are not a solution for all computing problems. Traditional computing methods work well for problems like balancing checkbooks, keeping ledgers, and keeping tabs of inventory and do not require the special characteristics of neural network. Hence, ANNs and conventional algorithmic computers are not in competition but complement each other. A large number of tasks require systems that use a combination of two approaches (normally a conventional computers is used to supervise the ANN) in order to perform at maximum efficiency

3.6 Designing Artificial Neural Network Models

The design issues in ANNs are complex and are the major concern of the system developer. The developer must go through a period of trial and error in the design

decisions before coming up with a satisfactory model in few shots. The process of designing ANN model is an iterative course of action that achieves an optimal result by adjusting the parameters of the network .In general, designing a model of ANN consists of the following decisions and activities.

- Deciding network topology which involves deciding the number of layers in the network, deciding the number of neurons in each layer, and deciding 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. Learning algorithm is a prescribed set of well defined rules for the solution of learning problem, like Hebb's learning rule, delta learning rule, perceptron learning rule, back propagation learning rule etc.
- Encoding scheme (deciding the way a neuron receives input and produces output). How the input and output data is represented, or encoded, is a major component to successfully instructing a network. Artificial networks only deal with numeric input data. Therefore, the raw data must often be converted from the external environment. Additionally, it is usually necessary to scale the data, or normalize it to the network's paradigm.

3.6.1 Layers of An Artificial Neural Network

Although there are useful networks, which contain only one layer, or even one element, most applications require networks that contain at least the three normal types of layers - input, hidden, and output as shown in figure 3.6. The layer of input neurons receives the data either from input files or directly from electronic sensors in real-time applications. The output layer sends information directly to the outside world, to a secondary computer process, or to other devices such as a mechanical control system. Between these two layers, there may be one or more hidden layers. The hidden layer extracts relevant features or patterns from the received signals. Those features or patterns that are considered important are then directed to the out put layer.

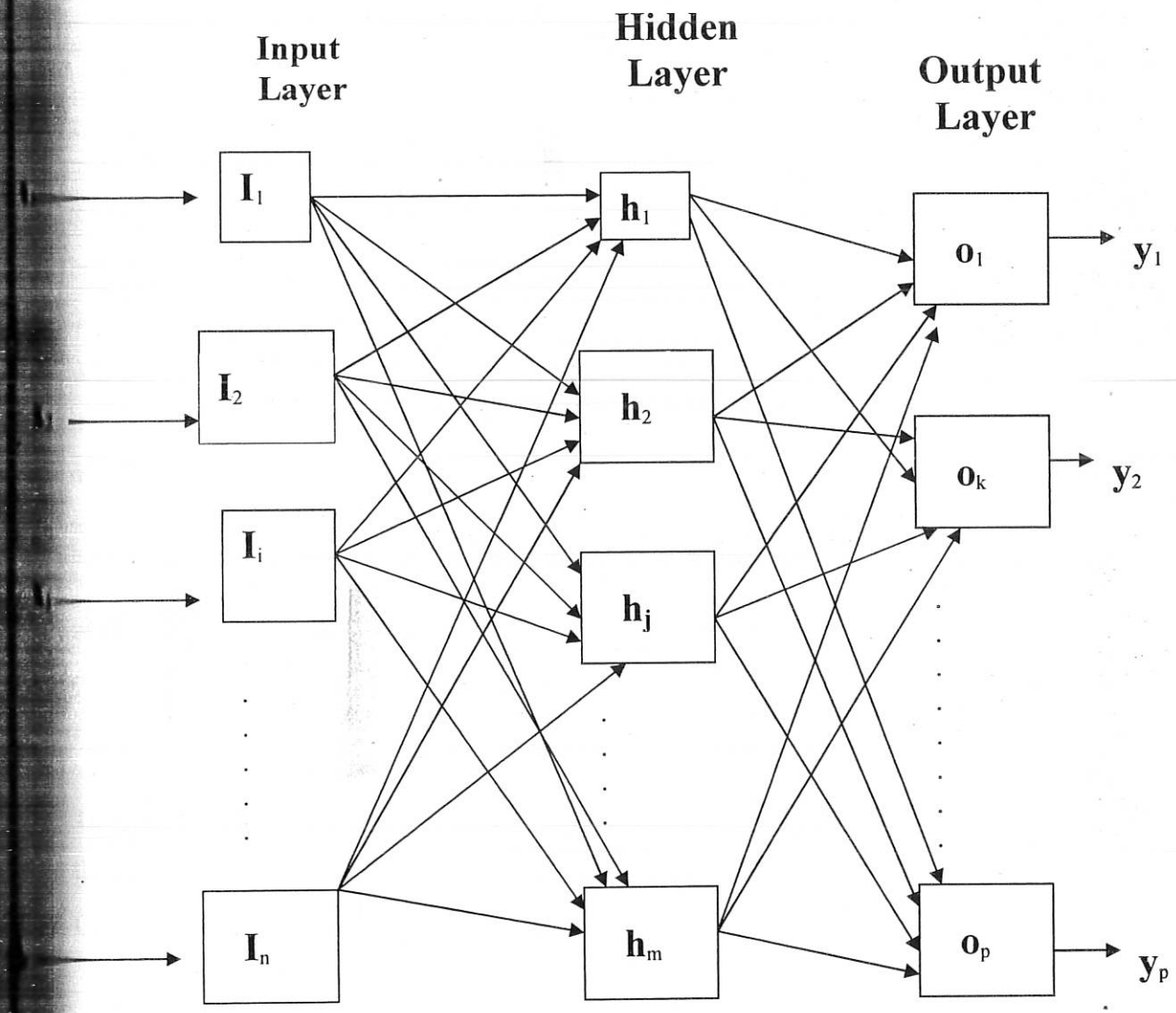


Figure 3.6.Three Layer Artificial Neural Network Architecture

In most networks each neuron in a hidden layer receives the signals from all of the neurons in a layer behind it, typically an input layer. After a neuron performs its function it passes its output to all of the neurons in the next layer, providing a feed forward path to the output. These lines of communication from one neuron to another are

important aspects of neural networks. The hidden units are free to construct their own representations of the input. To mimic the way in which biological neurons learn patterns, the connections between artificial neurons in a neural network are given adjustable connection weights, or measures of importance. This value of the interconnection will keep changing during the training session to hold the optimal value so as to make the best approximation to new patterns.

3.6.2 Communication and Type of Connections.

Since Artificial Neural Network is the interconnection of many neurons, the connection behavior of the neurons needs to be studied to have good understanding on how the network works. In human brain, neural networks are constructed in a three dimensional way from microscopic components. These neurons seem capable of nearly unrestricted interconnections. This is not true in any man-made network. Each input and hidden layer neurons are connected via a network of paths carrying the output of one neuron as input to other neuron. The path may be unidirectional or a two-way communication, The neuron in a layer may communicate with each other, or they may not have any connections within their layer. But it is mandatory that a neuron of one layer must be connected to at least a neuron of another layer. The way the neurons are connected to each other has a significant impact on the operation of the network.

The connection of neurons within the same layer is called intra-layer connection while the connection between neurons in different layer is called **inter-layer** connection. The type of **inter-layer** connection could be:

- **Fully connected**. Each neuron in the first layer is connected to every neuron on the second layer.
- **Partially connected**. A neuron of the first layer does not have to be connected to all neurons on the second layer.
- **Feed forward connection or uni-directional**. Feed-forward ANNs (shown in figure 3.4) allow signals to travel only from input to out put. There is no feedback (loops), i.e. the output of any layer does not affect that the same layer or previous

layer. Feed-forfard ANNs tend to be straight networks that associate inputs with outputs.

- **Resonance or feedback connection**. The layers have Bi-directional connections, and they can continue sending messages across the connections a number of times until a certain condition is achieved. Feedback networks are dynamic; their state is changing continually until they reach an equilibrium point. They remain at equilibrium point until the input changes and a new equilibrium needs to be found.

3. 7 The Learning Processing in ANNs

It is pointed out the excellence of ANN is its ability to "learn "from examples in the real world and its capability to deal with new patterns. In the context of ANN," learning" is a process by which the free parameters of a neural network are adapted through a process of simulation by the environment in which the network is embedded. Unlike telling the network how to react to each data vector separately, as conventional programming, the network itself is able to find properties from the presented data.

Therefore, "learning" in ANN is adaptive. The learning ability of neural network is determined by its architecture and by the learning algorithm chosen for training or" learning". Depending up on the environment in which the neural network operates, there are two basic approaches or paradigms to train an ANN: **supervised and unsupervised** training.

Unsupervised learning is training with out an external teacher and is based upon only local information. In this type of training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This learning by doing while supervised learning is learning by example. Unsupervised learning is sometimes referred to as self-organization since the network learns the pattern by itself.

layer. Feed-forfard ANNs tend to be straight networks that associate inputs with outputs.

- **Resonance or feedback connection**. The layers have Bi-directional connections, and they can continue sending messages across the connections a number of times until a certain condition is achieved. Feedback networks are dynamic; their state is changing continually until they reach an equilibrium point. They remain at equilibrium point until the input changes and a new equilibrium needs to be found.

3. 7 The Learning Processing in ANNs

It is pointed out the excellence of ANN is its ability to "learn "from examples in the real world and its capability to deal with new patterns. In the context of ANN," learning" is a process by which the free parameters of a neural network are adapted through a process of simulation by the environment in which the network is embedded. Unlike telling the network how to react to each data vector separately, as conventional programming, the network itself is able to find properties from the presented data.

Therefore, "learning" in ANN is adaptive. The learning ability of neural network is determined by its architecture and by the learning algorithm chosen for training or" learning". Depending up on the environment in which the neural network operates, there are two basic approaches or paradigms to train an ANN: **supervised and unsupervised** training.

Unsupervised learning is training with out an external teacher and is based upon only local information. In this type of training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This learning by doing while supervised learning is learning by example. Unsupervised learning is sometimes referred to as self-organization since the network learns the pattern by itself.

At the present time, unsupervised learning is not well understood. Life is filled with situations where exact training sets do not exist. Because of this unexpected aspect to life, there should be continuous research into this field.

Yet, the vast majority of artificial neural network solutions have been trained with supervision. In this mode, both the inputs and the desired outputs are provided. The network then adjusts weights, which are usually randomly set to begin with, so that the next iteration, or cycle, will produce a closer match between the desired and the actual output. The learning method tries to minimize the errors of all processing elements. This global error reduction is created over time by continuously modifying the input weights until acceptable network accuracy is reached.

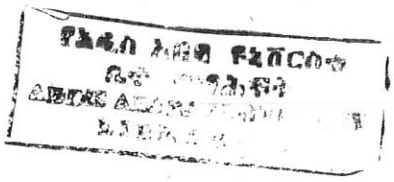
. This training is considered complete when the neural network reaches a user defined performance level. This level signifies that the network has achieved the desired statistical accuracy as it produces the required outputs for a given sequence of inputs. When no further learning is necessary, the weights are typically frozen for the application.

Training sets need to be fairly large to contain all the needed information if the network is to learn the features and relationships that are important. If the network is trained just one example at a time, all the weights could be drastically altered in learning the next fact. The previous facts could be forgotten in learning something new. As a result, the system has to learn everything together, finding the best weight settings for the total set of facts.

After a supervised network performs well on the training data, then it is important to see what it can do with data it has not seen before. If a system does not give reasonable outputs for this test set, the training period is not over. Indeed, this testing is critical to insure that the network has not simply memorized a given set of data but has learned the general patterns involved within an application.

If a network can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and even the initial weights themselves.

3.8 The Training Rule (Algorithm)



3.8.1 The Perceptron Training Rule (Algorithm)

Here the precise learning problem is to determine a weight vector that causes **a single** perceptron to produce the correct **+1** or **-1 Output** for each of the training examples. To solve this learning problem, the **Perceptron training rule** (algorithm) and **delta rule** (algorithm) are designed.

In the **Perceptron training rule** (algorithm), at each iteration, weights are modified as follows:

$w_i \leftarrow w_i + \Delta w_i$ ----------3.4 and

$\Delta w_i = \eta(T-Y)x_i$.----------3.5

Here, **T** is the target output for the current training example, **Y** is the output generated by the perceptron (as described in section **3.4.1**) and η is a positive constant called a **learing rate.** If the training examples are **not linearly separable**, convergence is not assured in the above training procedure.

3.8.2 The Delta Training Rule (Algorithm)

The Delta Training rule (algorithm) is designed to overcome the shorcoming of Perceptron training rule (algorithm). If the training examples are **not linearly separable,** Delta Training rule (algorithm) convergences a best-fit approximation to the target concept. The key idea behind the Delta Training rule is to use gradient descent to search the hypothesis space of possible weight vectors to find the weights that best fit the training examples.

The delta-training rule is best understood by considering the task of training **a linear unit** as descibed in section **3.4.2**. In order to derive a weight learing rule for linear units, first

define the training error of a hypothesis (weight vector) relative to the training examples as follows:

$$\vec{E}(W) \equiv 1/2 \sum_{d \in D} (T_d - Y_d)^2 \text{----------} (3.6) \text{ and } Y_d = W_0 + x_1 w_1 + x_2 w_2 + \ldots + x_n w_n,$$

Where D is the set of training examples, T_d is the target output for training example d, and Y_d is the output of the linear unit for trainig example d. Hence, **E (W)** is simply half the squared difference between the target output $\mathbf{T_d}$ and the linear unit output $\mathbf{Y_d}$,summed over all trainig example.By using Partial differentiation technique and by gradient Descent rule, we have the following:

$$w_i \leftarrow w_i + \Delta w_i \text{ ---------- and}$$

$$\Delta w_i = \eta X_{id} \sum_{d \in D} (T_d - Y_d) \text{----------} (3.7)$$

3.8.3 Multi-layer Perceptron (MLP) with Gradient Descent Back-propagation Algorithm (GDBPA)

As described above, single perceptron can only express linear decision surfaces. In contrast, the multi-layer perceptron (MLP) with a Gradient Descent Back-propagation Algorithm (GDBPA) is capable of expressing non-linear decision. This type of network remains a useful standard by which others are compared. **What type of unit is used as a basis for constructing MLP network**?

Since multiple layers of *linear units* produce only **linear functions** and since the **discontinous** naure of treshold perceptron makes it undifferentiable and unsuitable for ***gradient descent***, **sigmoid** unit is

selected as a basis for constructing **MLP** network as described in section **3.4.3.** The following figure shows **MLP** with **BPA**.

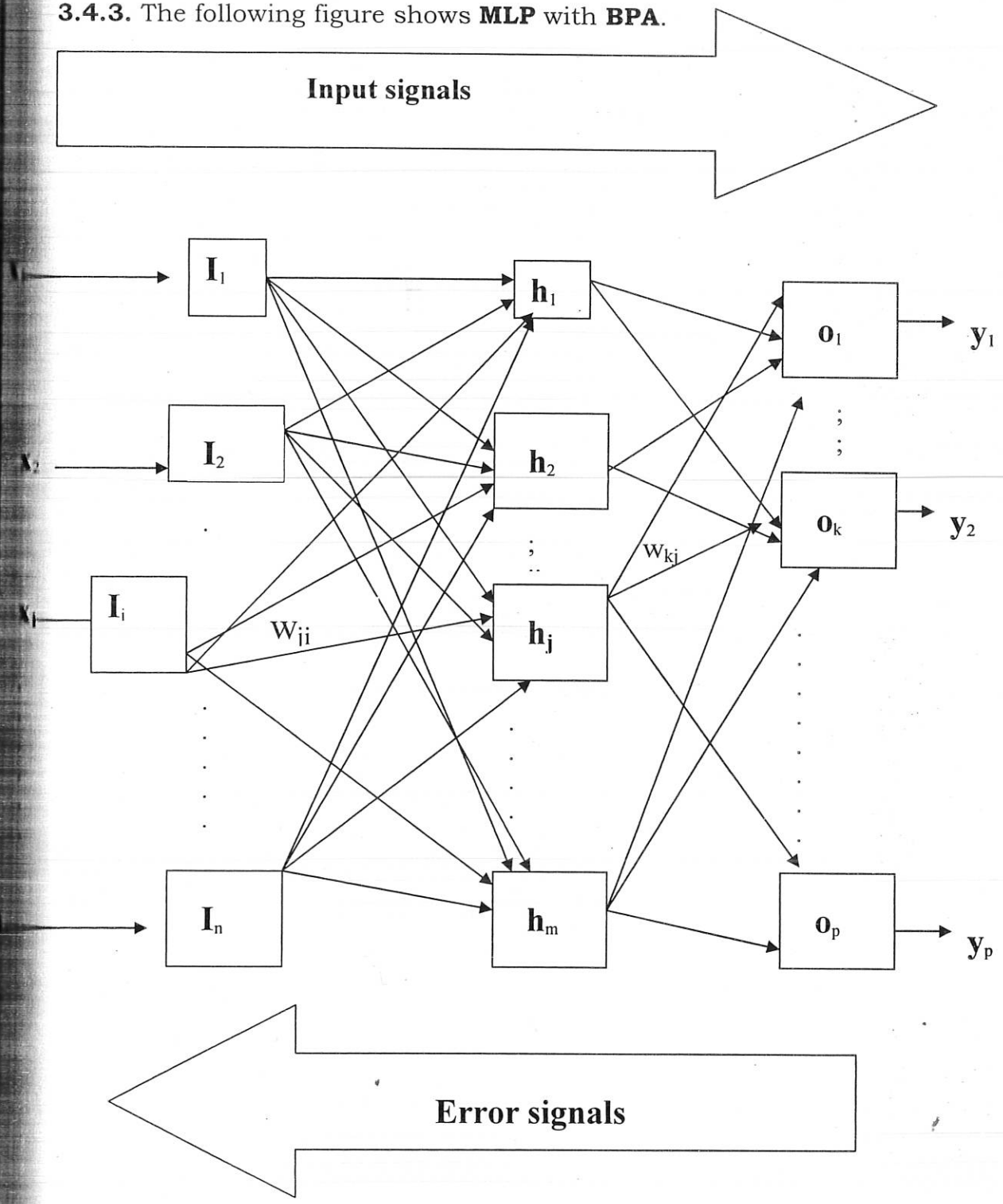


Figure 3.7. Typical MLP with GDBPA

As shown in figure **3.7**, a **MLP** with **GDBPA** has an input layer of source nodes and an output layer of neurons (i.e., computation nodes); 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, which are so called because 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 historical data so that the model can then be used to produce the output when the desired output is unknown.

The **MLP** with **GDBPA** involves two phases: forward Phase and backward Phase.

During *forward Phase*, the free parameters of the network are fixed, and the input signal is propagated through the network layer by layer as shown in figure **3.7.** The *forward phase* finishes with the computation of an error signal $\mathbf{e_i} = \mathbf{T_i} - \mathbf{Y_i}$ where T_i is the desired response and $\mathbf{Y_i}$ is the actual output produced by the network in response to the input $\mathbf{X_i}$.

During backward Phase, the error signal $\mathbf{e_i}$ is propagated through the network in the backward direction as shown in figure **3.7.** The adjustment (modification) of weights starts from the back (weights between the output and last hidden layer), and propagates to the front (weights between the first hidden layer and the input layer). During this phase, the adjustments are applied to the free parameters of the network so as to minimize the ***sum of square error*** between the network output and the desired output during training.

In other words, the error back-propagation is the way of using known input-output pairs of a target function to find the coefficients of a certain mapping function, which approximate the target function as closely as possible, i.e. given a set of input vectors and associated target vectors, the objective is to learn a rule that captures the underlying functional relationship between the input vectors and the target vectors. Mathematically, each target vector $\mathbf{Y_i}$ is a function of the input vector $\mathbf{X_i}$. i.e. $\mathbf{Y_i} = \mathbf{F}(\mathbf{X_i})$,

The task of the back-propagation network is to learn the function **F**. This is achieved by finding regularities in the input patterns that correspond to regularities in the output patterns. The network has a weight parameter vector, whose values are changed to modify a function computed by the network to be as close as possible to **F.** Back-propagation learning may be implemented in one of two basic ways: *sequential mode and batch mode*. In sequential mode (also referred to as the on-line mode) of BP learning,

weight adjustments are made to the free parameters of the network example-by example. On the other hand, in batch mod of BP learning, weight adjustments are made to the free parameters of the network epoch- by- epoch, where each epoch consists of the entire set of training examples.

In sequential mode, each example is analyzed one by one and the network estimates the outputs based on the values of the inputs. For every example, each input node passes a value of an independent variable $\mathbf{X_i}$ to all the nodes of the hidden layer. Each hidden node computes a weighted sum of the input values based on its weights. The weights are determined during the learning mode. Finally, from this value of the weighted sum, the hidden nodes compute a sigmoid output $\mathbf{h_i}$ of the hidden nodes. The sigmoid function provides a bounded output of the hidden node. Each of the output nodes receives the outputs of the hidden nodes $\mathbf{h_i}$, computes a weighted sum of the inputs based on the weights and finally, determines the sigmoid output $\mathbf{Y_i}$ of the node. The output from the output node is compared with the target output and the error is propagated back to adjust the connecting weights $\mathbf{W_{ik}}$ as well as b_{ik} as shown in figure **3.7.** The cycle is repeated until the overall error values drop below some predetermined threshold. At this point we say that the network learned the problem "well-enough". Of course, the network would never exactly learn the ideal function, but rather it will asymptotically approach the ideal function.

Since a **MLP** with **GDBPA** involves multiple output units rather than single unit,the **sum of mean squared error**, at each iteration, will be redefined as follows:

$$E(\mathbf{W}) \equiv 1/2 \sum_{d \in D} \sum_{k \in p} (T_{kd} - Y_{kd})^2 \text{----------} \quad \textbf{(3.8)}$$

Where:

- **E (W)** is the sum of man square error over over the network.
- **P** is the set of output units in the network
- $\mathbf{T_{kd}}$ is the target value associated with the K^{th} output unit and training example d
- $\mathbf{Y_{kd}}$ is the output value associated with the K^{th} output unit and training example d.

Backpropagation learning algorithm For Feed-forwardnetworks containig three layers of sigmoid units is presented as follows:

INPUT: Backpropagation (training examples,η, n_i, n_h, n_o)

- **Each trainig example is a pair of the form (X,T),where X is the vector of network input values ,and T is the vector of target output values. η is the learning rate (e.g. 0.5), n_i is number of nodes in input layer, n_h is number of nodes in the hidden layer,and n_o number of nodes in the output layer.**

OUTPUT: a Neural Network Trained to Classify.

PROCESS:

1.Create a feed-forward network with n_i input units, n_h hidden units, and n_o output units.

2.Initialize all network weights to small random numbers (e.g.b/n -0.5 and +0.5)

3. While terminating condition is not satisfied, Do

For each (X,T) in the training examples,Do

I. Propagate the input forward throught the network:

For each hidden layer unit **j or** for each output layer unit **j**

a. $I_j=\sum w_{ij} + b_j$ ---------- **(3.9)**---net input to unit j with respect to the previous layer i and

b. $Oj = 1/1+e^{-Ij}$ --------- **(3.10)**- -- the output from unit **j.**

II. Propagate the error backward through the network:

c. For each unit k in the output layer, comput the error **term** $\delta_k= O_k(1-O_k)(T_k-O_k)$---------(3.11)

d. For each unit j in the first hidden layer from the output layer, compute the error term δ_j:

$$\delta_j = Oj(1-Oj) \sum_{K \in p} \delta_k \, wjk \text{----------(3.12)}$$

e. For each weight W_{ij} in the network, calculate weight **increment** and update the weight:

$\Delta w_{ij} = \eta \, \delta_j \, X_{ij}$ **----------(3.13) and**

$w_{ij} = w_{ij} + \Delta w_{ij}$**----------(3.14)**

f. For each bias bj in the network, calculate bias increament and update the biase:

$\Delta bj = \eta \, \delta j$ **---------- (3. 15) and**

bj= bj+Δbj----------(3.16)

3.9 Properties and Performance of MLP with GDBPA

The back-propagation algorithm has a number of properties that make it highly attractive.

1. Learning, not programming. What the network does is learning not programming.

2. Generalization. **MLP** with **GDBPA** networks can generalize. This is an essential property of intelligent systems that have to function in the real world. Generalization in this sense means that similar inputs lead to similar outputs.

3. Noise and fault tolerance. The network is noise and fault tolerant. Note that this property is not explicitly programmed into the model. It is a result of the massive parallelism

4. Re-learning. The network shows fast re-learning. If the network is distorted to a particular performance level it re-learns faster than a new network starting at the same performance level.

CHAPTER FOUR

WORD CLASSES AND TAG SETS FOR AMHARIC LANGUAGE

4.1. Introduction.

In order to exchange their ideas, speakers of any languages should combine phones (sounds) in to a word, words in to a phrase and phrases in to a sentence in a specified manner. So there are four major different branches of linguistics (Phonology, Morphology, syntax and Semantics) to study every language at sound, word, phrase and sentence levels of analysis.

Phonology concerns with sounds (phones) and it tries to identify rules that govern combination and co-occurrence of phones to form ordinary types of words (free morphemes). **Morphology** deals with components of a word (morphemes) and the rules that govern their combination and co-occurrence to form complex types of words. At next higher level, **syntax** deals with the combination of words in a phrase, the combination of phrases in a sentence and the rules that govern their combinations. **Semantics** is concerned with the meaning of words and sentences (Allen, 1995). **This thesis** focuses on **word level analysis**, particularly on **word classes or parts of speech**.

4.2. Why Words Are Categorized.

Words are fundamental units in every sentence. The grammar of a language dictates the way words are arranged and combined during communication. String of words arranged based on the grammar of the language will form a sentence. The meaning of a sentence can be understood from the meaning of individual words and the way they are arranged. But all words do not have equal contribution to the meaning of a sentence.. The contribution of a word to the meaning of a sentence depends on the **category** and feature of the word. The category of the word also determines the possible syntactic rules to be applied on the word. Hence, to determine the contribution of a word to the meaning of a sentence and the rules that can be applied to a particular word, identifying the category of the word should be the first step. More over, studying the characteristics of all words one by one is a difficult task. So the most efficient method is dividing them in to different

categories (word classes or parts of speech) and then studying the behavior the whole class.

For example, in English language nouns, pronouns, verbs, adverbs, adjectives, prepositions, conjunctions and interjections are generally recognized (Jurafsky, 2000). Like English language, Mersehazen (1934) categorized words of the **Amharic language** in to the above eight word classes (parts of speech). But Baye(1987) treated pronouns and conjunctions as a subclass of nouns and prepositions respectively. He did not also considered interjections as one part of word classes. So he reduced the above eight classes in to five classes (nouns, verbs, adjectives, adverbs and prepositions)

A difference like this refers to the fact that the boundaries between the word classes are not absolutely fixed. Many word classes share characteristics with others, and there is considerable overlap between some of the classes. In other words, the boundaries are "fuzzy", so different grammars draw them in different places. So how can we distinguish word classes in general?

4.3.Criteria For Word Classes

Consider the words in the Amharic sentence **wendmE tlq mekina ynedal** "my brother drives a big car". We began by grouping words more or less on the basis of our instincts about Amharic language. We somehow "feel" that **wondim** Ina **mekina** (brother and car) belong to the same class, and that **wondm** Ina **mendat** (brother and drives) belong to different classes. However, in order to conduct an informed study of grammar, we need a much more reliable and more systematic method than this for distinguishing between word classes. We use a combination of three criteria for determining the word class of a word: The meaning of the word, the form or `shape' of the word and the position or `environment' of the word in a sentence.

4.3.1 The Meaning of a word.

Using this criterion, we generalize about the kind of meanings that words convey as in:

- Words that denote actions (Verbs)
- Words that denote entities (Nouns)
- Words that denote states (Adjectives)

- Words that denote the manner in which something is done (Adverbs)

For example, we could group together the words brother (wondm) and car (mekina), as well as kebede, house (bet), and London (lendèn) on the basis that they all refer to people, places, or things. In fact, this has traditionally been a popular approach to determining members of the class of nouns. It has also been applied to verbs, by saying that they denote some kind of "action", like cook, drive, eat, run, shout, and walk.

This approach has certain merits, since it allows us to determine word classes by replacing words in a sentence with words of "similar" meaning. For instance, in the sentence **yene lg beyeqenu Irat yseral (my son cooks dinner every day),** we can replace the verb ***yseral*** (cooks*)* with other "**action**" words like: **prepares** (yazegajal), **eats** (ybelal) etc…

- My son cooks dinner every day **(yene lg beyeqenu Irat yseral)**
- My son prepares dinner every day (**yene lg beyeqenu Irat yazegajal)**
- My son eats dinner every day **(yene lg beyeqenu Irat ybelal)**

On the basis of this replacement test, we can conclude that all of these words belong to the same class, that of "action" words, or verbs.

However, this approach also has some serious limitations. The definition of a noun as a word denoting a person, place, or thing, is wholly inadequate, since it excludes abstract nouns such as **death** (mot), ***life*** (hywet), ***knowledge*** (Iwqet), ***chance*** (idl) etc. Similarly, The definition of a verb as a word denoting an "action" excludes a verb like ***be*** (mehon), as in **I want to *be* happy** (desteNa mehon Ifelgalehu). What "action" does ***be*** (mehon) refer to here? So although this criterion has certain validity when applied to some words, we need other, more stringent criteria as well.

4.3.2. ***The form or `shape' of a word.*** In order to solve the limitations of the above criteria, some words can be assigned to a word class on the basis of their form or `shape'. Morphological categorization depends on the fact that certain types of affixes are

If we really need to, we can also apply a replacement test, based on our first criterion, replacing cook in each sentence with "similar" words. Notice that we can replace verbs with verbs, and nouns with nouns, but we cannot replace verbs with nouns or nouns with verbs like I **car** dinner every Sunday.

It should be clear from this discussion that there is no one-to-one relation between words and their classes. **Cook** can be a verb or a noun. It depends on how the word is used. In fact, many words can belong to more than one word class. However, they only belong to one word class at a time, depending on how they are used. So it is quite wrong to say, for example, "**cook** is a verb". Instead, we have to say something like **"cook"** is a verb in the sentence **I cook dinner every Sunday**, but it is a noun in the sentence **cook is on holiday.** In short, the category of words can be determined semantically, morphologically, or syntactically.

4.4. Category of Words in Amharic

In the early works of word categorization, words of the Amharic language were categorized in eight parts of speech: nouns, pronouns, verbs, adverbs, adjectives, prepositions, conjunctions and interjections (Mersehazen, 1934). In the early works of word categorization, the major criterion to categorize words was the meaning of the word. Since categorizing words based on their meaning is mental activity, it is subjective and it is also affected by cultural, social and psychological factors. In other words, it is not objective, tangible and observable criteria. It seems that the above eight word classes were obtained simply from the English language with out studying the behavior of the Amharic language in detail (Getahun, 1989).

Nowadays, ***The form or `shape' of a word*** is one of the tangible, objective and observable criterion to categorize words. Words that are similar in their shape can be categorized in one category. For instance, if we compare the Amharic words **bela**(he ate) and **lam**(cow), the word **lam** can take a bound morpheme **{-oc}** and it can have the shape **lamoc** while the word **bela** cannot do this. The other tangible, objective and observable criteria to categorize words is the position or `environment' of a word in a sentence.

Words that can take similar positions in a sentence can be categorized in one category. If we see the above words from this point of view, the word **bela** can be found at the end of a sentence while the word **lam** cannot do this. On the contrary, the word **lam** can be found at the beginning of a sentence and at the middle of the sentence before a verb while the word **bela** cannot do the same. Although the major criteria to categorize words are the shape of a word and the position of a word in a sentence, the meaning of a word can also support.

By examining the Amharic words using these two major criteria, Baye (1987) categorized Amharic words in to five classes. These are nouns, verbs, adverbs, adjectives, and prepositions. He also presents justification why pronouns, conjunctions and interjections are not treated as major categories of words in Amharic as follows. Since pronouns are a subclass of nouns and characteristics of conjunctions is similar to prepositions, they are categorized under the category of **nouns and prepositions** respectively. In Amharic there are words (like goS, zeraf, gud etc) that are used to express feelings such as pleasure, surprise, annoyance etc. These words do not change their shape for grammatical categories (for gender, person, number etc). They can stand-alone by themselves outside a sentence or they can appear any where in a sentence as shown below.

- goS !
- goS ! hekimu meTa.
- hekimu meTa gos !

These words are not combined with any word in a sentene. But they are combined with the whole sentence. From the above sentence, the word **goS** is not combined with the word **hekimu** or **meTa.** It is combined with the sentence **hekimu meTa**.

Since it is impossible to identify interjection words from their shape and Position in a sentence, Baye(1987) does not consider them as one part of speech category in Amharic. Below characteristics that are unique for each of the above five word categories in Amharic language is given.

4.4.1. Noun

Words in this category have the following morphological (shape of a word) and syntactical (the position of a word in a sentence) characteristics.

- They can take the bound morpheme {-oc} to make plural number .
- They can be found at the beginning or at the middle of a sentence being the lexical head of the phrase that functions as a subject or an object.
- They may change their shape not only for the grammatical category **number** but also for other grammatical categories such as definiteness, gender, and case.

Although there are some words (like gobez "clever", rejm "tall" etc) which can take the bound morpheme {-oc} for their plural marker, they cannot be categorized as noun because their syntax role is different from that of nouns. Pronouns (like Isu "he", Isua "she" ante "you' etc) are similar in their position with nouns. But they take a different bound morpheme {In-}to mark plural noun. Like pronouns, proper nouns also take the bound morpheme {In-} for their plural marker as shown below.

- Abebe →Inabebe, kebede→Inkebede.
- Isu"he"→Inersu"they", Isua"she".

Besides pronouns and proper names ,there are infinitive and abstract nouns that do not take the plural marker **–oc.** However,all such words are categorized under noun along with nouns that take the regular plural marker **–oc** because they have similar role in syntax. i.e. they can be a lexical head of an object or a subject. They can also take prepositions to form prepositional phrases. Thus pronouns are treated as subcategory of nouns instead of considering as a separate word category.

4.4.2. Adjective

Amharic words in the adjective class usually precede the nouns they modify. For instance, in the sentence **tlq beg new** "it is a big sheep", the adjective **tlq** "big" precedes the noun **beg** "sheep" it modifies. But every word that comes immediately before a noun cannot be considered as an adjective. For instance, in the phrase **yh beg** "this sheep", although the word **yh**"this" shares the function of an adjective (modifier), it is a demonstrative pronoun. This concept can be justified by adding the word **beTam** "very" to the above phrase as follows.

- beTam **yh** beg (not acceptable in Amharic language)
- beTam **tlq** beg (acceptable in Amharic language

Thus any word that can be accepted between the words **beTam** and **beg** can be considered as an adjective (Baye, 1987). Adjectives also change their shape for gender, number and definiteness (for detailed information see Baye, 1987 and, Getahun, 1997).

4.4.3. Verb

There are two major criteria to identify verbs from other word categories. First, syntactically, verbs function as predicates and in unmarked simple sentence, they are found at the end of a sentence. Second, morphologically, they reflect for the grammatical categories such as aspect, mood agreement. They also suffix bound morphemes that indicate the subject of a sentence as shown below.

- Isua msawan **bela-c** (she ate her lunch).
- Ante beg **geza-h** (you bought a sheep).
- InE wetet **TeTah-u** (I drunk milk).

In the above sentence, the words **bela-c, geza-h, and TeTah-u** are verbs and the bound morphemes {-c}, {-h}, {-u} indicate 3rd person, 2nd person and 1st pronouns, which are used as the subject of a sentence. Generally, verbs change their shape for person, gender,

number and time by attaching prefixes and suffixes (for detailed information see Baye, 1987 and, Gtahun, 1997).

4.4.4. Adverb

An adverb is a word that modifies a verb. Since the number of words in this category is very small, noun phrases, prepositional phrases and subordinate clauses do the function of adverbs as shown below.

- Kasa **ZarE** beg ygezal (kasa will buy sheep today).
- Kasa **wede gojam** hEde(kasa went to gojam).
- Kasa **bednget** hEde (kasa went suddenly).
- **Kasa wede gojam slehEde** aster aleqesec (Aster cried because kasa went to gojam).

In the early works, **zarE, wede gojam, bednget** and **Kasa wede gojam slehEde** were treated as adverbs of time, place, manner and reason respectively. But the word **zarE** is a noun which is used as an adverb of time in the given sentence. **wede gojam** and **bednget** are **not** words. They are **prepositional phrases** which are used as an adverb of place and manner respectively. **Kasa wede gojam slehEde is a** subordinate clause, which is used as an adverb of reason.

4.4.5. Preposition

In the early works of word categorization, prepositions and conjunctions were treated as two different categories. But nowadays, conjunctions are treated as a subclass of prepositions because words in both categories have the following major characteristics in common (Baye, 1987).

- Any bound morpheme cannot be attached to them (they do not change their shape for number, person, gender, tense, case etc.
- They are not used as a base for the formation of new words and they are not formed from other words.

- ❖ They do not have meaning by their own. They have meaning when they are attached with other words (noun, verb, adjective, adverb).

Generally, since inflectional and derivational morphology are not applied, prepositions and conjunctions are closed words while nouns,verbs, adjectives and adverbs are open words.

4.5. Tag Set For The Amharic Word Classes.

Most of the tag sets presented in this section are drawn from the data compiled by Mesfin(2001) with some modification. In this study, two new tag sets are added (**NVP and JN**) and one tag is also reduced **(ITJ),** where **NVP** stands for a preposition that cannot be separated from verbal noun (e.g. **lemamTat** "to bring", **lemegemer** "to start", **bemaseraCet** "by distributing", **bemefTer** "by creating" etc...) and **JN** stands for a noun which functions as an adjective (e.g. the word **yh** in the phrase **yh tarik** "this history, the word **Inihn** in the phrase **Inihn aynet lamoc** " these kinds of cows", etc), and **ITJ** stands for interjections (e.g. **goS!**(wonderful),**wa!**(take care) etc).In this study, the following tag sets are used (For the detailed information refer to mesfin (2001).

Table 4: Tag set for Amharic language

No.	TAG	Brief Description
1	N	N stands for proper noun, common noun, all types of pronouns (personal pronoun, reflexive pronoun, demonstrative pronoun, Interrogative pronoun, relative pronoun, Indefinite pronoun, Possessive pronoun) which are invariant of gender (male, female, neutral), number (plural, singular), case (nominative, accusative, dative, genitive), and definiteness (definite, indefinite). Examples for this category are **ityoPya** "Ethiopia", **heger** "country", **bahloc** "cultures", **etc**
2	NV	NV stands for a noun derived from verb(verbal noun). Examples for this category are **menqesaqes** (moving), **meqemeT** (sitting) etc...
3	NVP	NVP stands for a word from which a preposition cannot be separated from the

		verbal noun. Examples for this category are **bemaseracet** (by distributing), **lemamtat** (to bring) etc…
4	NB	NB stands for a word from which the prefix bale cannot be separated from a noun. Examples for this category are **bale-mekina**(the car owner),**bale-coat** (the owner of coat) ,etc
5	NP	NP stands for a word from which a preposition cannot be separated from the noun. Examples for this category are **bemerkeb** (by ship),**sletmhrt**(about education) etc…
6	NC	NC stands for a word from which a conjunction cannot be separated from the noun. Examples for this category are **begs** (what about sheep), **raym/**NC, etc…
7	V	V stands for main verbs, which are in variant for gender (male, female, neutral), number (singular, plural), person (1st, 2nd, 3rd), Tense (the six forms of tense) and polarity (affirmative and negative). Examples for this category are **bela** "he ate", **sebere** "he broke", etc…
8	AUX	AUX stands for auxiliary verbs. Like main verbs, they are in variant for gender, number, person, Tense and polarity. Examples for this category are **alec**(she is present), **neber**(he was present),etc..
9	VCO	VCO stands for compound forms of the words **ale, aderege,and aseNe** which are invariant for morphosyntactic features like No.7 and 8. Examples for this category are **quCale**(he sits),**qeded aderege**(he tears),etc…
10	VP	VP stands for a main verb or auxiliary verb from which prepositions cannot be separated. Examples for this category are **slemimar** (since he learns), **kemeTa** (if he comes) …
11	VC	VC stands for a verb to which conjunctions are prefixed or suffixed. Examples for this category are medresm,yemetuTm etc
12	J	J stands for adjectives that are invariant to gender, number, definiteness. Examples for this category are **gobez**(clever), **aCr**(short),etc
13	JN	JN stands for nouns that are used as adjectives. Examples for this category are the word **yh** in the phrase **yh tarik** "this history, the word **Inihn** in the phrase **Inihn aynet lamoc** " these kinds of cows", etc

14	JPN	JPN stands for a noun from which prepositions cannot be separated but used as an adjective. Examples for this category are the word **yeqoda** in the phrase **yeqoda coat** "leather coat", the word **yebuna** in the phrase **yebuna mankiya** "spoon for coffee",etc .
15	JNU	JNU stands for a word that represents a numerical value and used as an adjective. Examples for this category are **sost kilogram** "three kilogram,etc…
16	JP	JP stands for a word from which prepositions cannot be separated from an adjective. Examples for this category are **yeteleyayu**(different), **yekll**(region's),**begara(**together with) etc…
17	JC	JC stands for a word from which conjunctions cannot be separated from an adjective.e.g. **hulum**(all of them),etc..
18	ADV	ADV stands for an adverb. E.g. **zarE**(today),**qedemblo**(previously),**qesbeqes**(gradually)etc…
19	ADVC	ADVC stands for an adverb that has a conjunction suffixed to it. e.g. **negem**(even tomorrow),**tnantnam**(even yesterday),etc…
20	C	C stands for a coordinating conjunction that cannot be attached to other words.e.g. **Ina**(and),**weym**(or),etc
21	PRE	PRE stands for a preposition that cannot be attached to other words. e.g. **gar** (with), **wede**(in to),**Inde**(like),etc…
22	REL	REL stands for a relative verb. e.g.**yejemerubet, yetenesubet, yetewesenebet** etc.
23	ORD	ORD stands for ordinal number.e.g. **sosteNa**(third),**amsteNa**(fifth) etc..
24	CRD	CRD stands for cardinal number. e.g.**sost**(three),**amst**(five)etc…

CHAPTER FIVE: CORPUS AND LEXICON PREPARATION FOR MLP_TAGGER.

5.1 Sampling Technique.

In this research work, the problem of part of speech tagging was undertaken by using Artificial Neural Network (ANN). ANN needs a large corpus size of words. However, such a large Amharic corpus is not readily available .The required corpus is thus prepared from scratch in this study. The size, however, is very small compared to the corpus size used in the experiment of other languages (like English). This is mainly due to the absence of manually tagged corpus in the Amharic language. It is also laborious, expensive, and time consuming to tag manually a large corpus.

A corpus of 159 sentences (2,826 words-a word by word count with out including punctuation marks) is prepared for the experiment. The corpus is prepared from nes'anetn yemayawk nesa awC book by Andargacew Tsege(1977). This book is suggested as a source of sample by a linguist (Dr.Girma Awgchew, department of linguistics, AAU) because it is a narrative text and easy to tag manually. More over, most of the sentences in this book do not contain interjections, which are not treated as one part of word classes (part of speeches) in this study. He also suggested that the book consists of all word classes of the language in sufficient number. In short, this book is selected by using **judgment-sampling** technique by a linguist. Since this book is subdivided in to 12 chapters, **stratified sampling** technique is used to include pages from each chapter. Three pages are selected from the total number of pages in each chapter by using **systematic sampling technique**, i.e. if a chapter has 30 pages and if 3 pages are taken from these pages, first divide 30 by 3 to get 10.then the 1st page, the 11th page, the 21th page, the 31th page ,the 41st page etc are taken. The corpus is transcribed in to its Latin version and then this version is tagged manually by a post-graduate student from the department of linguistics.The correctness of its transcription and tagging is also checked by Dr.Girma Awgchew. The sample text and its transcribed version are found in the appendix by the name **sample corpus in Amharic alphabet and sample corpus in Latin alphabet** respectively. The manually tagged corpus is also found in the appendix

by the name corpus tagged manually. Since lexicon and lexical probability are essential to represent input to the MLP-tagger, a brief description of them is given below.

5.2. Lexicon Preparation

From the sample corpus which is tagged manually, a lexicon was prepared as shown in the table 5.1.This lexicon is again used to prepare the matrix of lexical probabilities as shown in table 5.2.

Table 5.1: Part of the entry of the entire lexicon that is designed for this study (a lexicon of distinct words and their corresponding frequency in each tag).

Distinct words	N	NV	JN	.....	NP	V	AUX	VP	J	ADV	.....	PRE	TOTAL
ityoPya	12	0	1	.....		0			0	0	.....	0	1 3
beSwoc	0	0	0	.....	2	0			0	0	.....	.....	2
yemiqoteru	0	0	0	.....	0	o		1	0	0	.....	.....	1
ametat	4	0	0	.....	0	0		0	0	0	.....	....	4
.....													
.....													
.....													
neber	0	0	0	0	0	0	25	0	0	0	.....	.....	20

From the above lexicon, the horizontal total provides information on the use of each word in the corpus (the frequency of each word in the corpus). For example, there are **thirteen** uses of the word **ityopya** 'Ethiopia" and **four** uses of the word **ametat** "years. Each entry also indicates how many times each word occurred in each part of speech category. For instance, the word **ityopya** was used twelve times as a noun and one times as adverb.

5.3. Preparation Of Lexical Probabilities

From the lexicon presented in table 5.1, the lexical probabilities of each word for each of the 24 categories (excluding punctuation marks) was obtained. A sample extracted from the full corpus is presented in the table 5.2 below.

Distinct words	N	NV	JN	....	NP	V	AUX	VP	J	ADV	....	PRE
ityoPya	0.923	0	0.077	.....	0	0			0	0	.....	0
beSwoc	0	0	0	.....	1	0			0	0	.....	.....
yemiqoteru	0	0	0	.....	0	o		1	0	0	.....	.....
ametat	1	0	0	.....	0	0		0	0	0	.....	....
.....												
.....												
.....												
neber	0	0	0	0	0	0	1	0	0	0	.....	.....
tark	**0.359,**	**0.0256**	.....	.....	.....	..		....				
yalat	**0.037**	**0.037**	.....	....	....	.....						
heger	**0.2632**	**0.2632**	.....	.....	.....	.....	.....	.....				

Table 5.2. The lexical probability matrix. Each word in a text document is represented in a form of vector with 24-element.

5.4. Database Design

A database consisting of three tables was prepared as a knowledge base for neuro-tagger. Microsoft Access has been used for the database design. The tables are Full form lexicon Table, Training Table and Testing Table. The full form lexicon contains distinct words of

the corpus with their corresponding lexical probabilities. The Train Table contains words for training and the Test Table contains words for testing.

5.4.1 Design Of Full Form Lexicon

Each word in a full form lexicon has the following fields:

- Name of the word: string of characters.
- Lexical probability corresponds to each tag type: the type of tag the word belongs to; which can be N, V, ADJ, ADV, etc.

Based on this analysis a table is designed, which has the folowing structure.

Table 5.3 Structure of the full form lexicon table.

Field name	Data type	Description
Word name	Text	Primary key
Lexical probability corresponds to each tag type	Number	A four digit decimal number

The relational schema of the *full form* table is given as follows.

FullFormTable

WordName	Lexical probability corresponds to a tag type.

5.4.2 Design Of Training And Testing Tables. These tables have the following fields:

- Decimal code as a primary key.
- Name of the word.

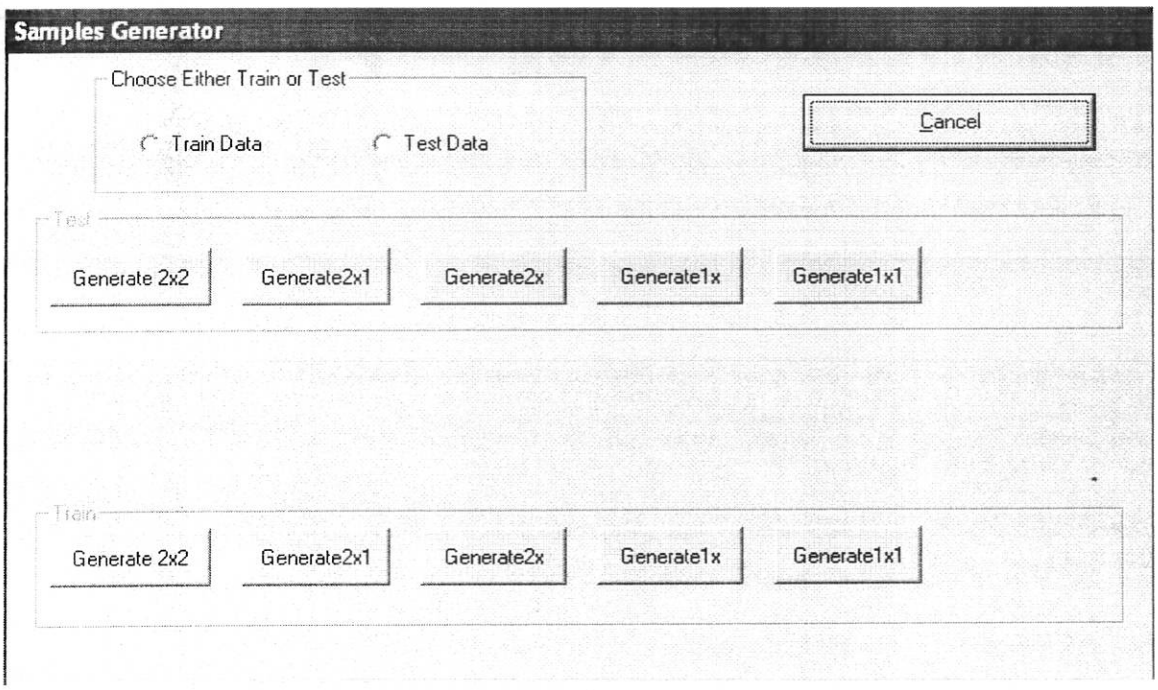
Table 5.4 structures of training and testing tables.

Field name	Data type	Description
Decimal code	Number	Primary key.
Name of the word	Text	String of characters.

The relational schema of the training table and testing tables is given as follows.

Decimalcode	Word name

Algorithms are designed to generate five samples for training purpose (labeled as 1_X_0-TRN, 2_X_0_TRN, 1_X_1_TRN, 2_X_1_TRN and 2_X_2_TRN) that contain words with their corresponding features (input) and target tag type (output) for neural network. The corresponding samples for testing (labeled as 1_X_0-TST, 2_X_0_TST, 1_X_1_TST, 2_X_1_TST and 2_X_2_TST) are also generated that contain words with their corresponding features (input) but with out their corresponding target tag type (output) for neural network. Microsoft Visual Basic version 6.0 programming language has been used to generate the inputs of neural network. This language is chosen for its user –friendly functionalities as shown below.



The lexical probability values in figure 5.2 were computed from the lexicon (figure 5.1) by using the following formula:

P(word/ tag)= P_K=P(w/ T_k)=n(T_K,W)/n(W),----------5.1

Where n(T_K,**W**) is the number of occurrences of the word **W** labeled as the tag T_k in the corpus and n(**W**) is number of occurrences of the word W in the corpus. For example, P(ityoPya/N)=12/13=0.923 and P(ityoPya/JN)=1/13=0.077 because n(ityopya,N)=12,n(ityopya,JN)=1,and n(ityopya)=13.

5.5 Representation Of Input For Mlp_Tagger

The problem of part of speech tagging can be tackled at two levels: word level and sentence level. In the case of word level tagging, the problem can be posed as a classification problem. Where as, in the case of sentence level tagging, a series of tags corresponding to the sequence of words in the sentence need to be found and in this case the context provided by the whole sentence influences the tag assignment. In this study, the MLP-tagger with error 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 in to 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:

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(label) of the target word **yalat** in the Amharic sentence **ityoPya tarik yalat heger nat,** the words **ityoPya, tarik, heger,** and **nat** can be taken as context words. i.e. F (ityoPya_ tarik_**yalat**_ heger_ nat})=tag.

Each word **W** from the corpus is encoded as an **n-element** vector W=(P_1, P_2, P_3, . . .P_k... P_n), where **n** corresponds to the total number of tags and P_k is the prior probability (lexical probability) that the word W corresponds to the tag T_K .P_k is estimated by using

the formula **(1)** which is indicated above. In this study, each word in a text document is processed and represented in a form of vector with 24-elements (since we have 24 tags). Generally, The input **IN** to the MLP tagger can be represented as follows:

$IN=(W_{T-L} \ldots W_{T-2}, W_{T-1}, \mathbf{W_T}, W_{T+1}, W_{T+2}, \ldots W_{T+R})$ **----(5.2)**

Each element $\mathbf{W_k}$ ($W_k \in W_{T-L}, \ldots W_{T-2}, W_{T-1}, \mathbf{W_T}, W_{T+1}, W_{T+2}, \ldots W_{T+R}$) 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. For instance, from the above sentence, the input for the target word **yalat** can be represented as:

IN=(tarik_ yalat_ heger)=(0.359, 0.0256, …, 0.037,…,0.2632).This is an input vector with **72-elements** to determine the tag of the target word **yalat.**

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.

For this study, a generic three layer MLP Network with windowed input is shown below in figure 5.1

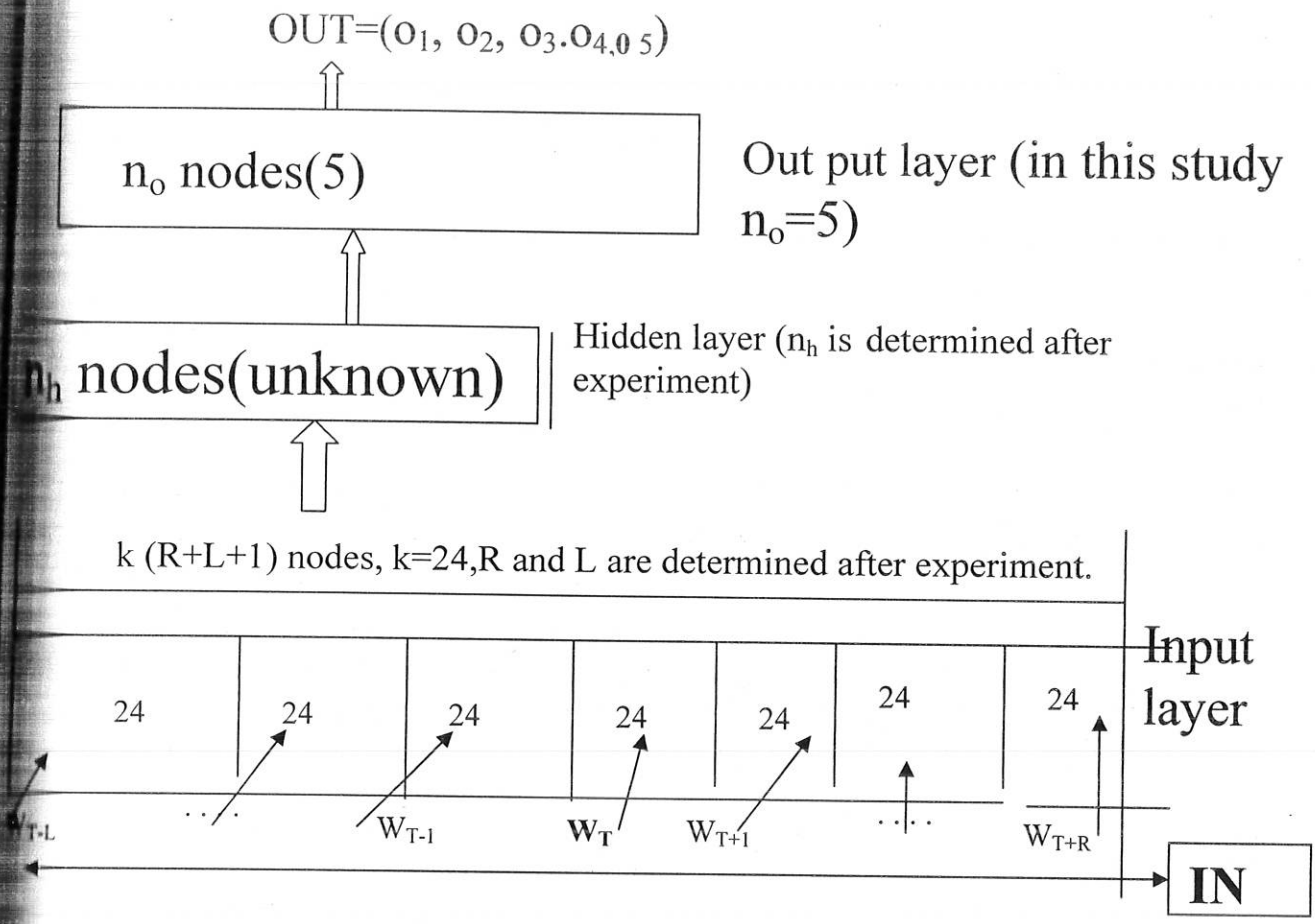


Figure 5.1 A generic three layer MLP Network with windowed input.

Legend: $\mathbf{n_i}$, $\mathbf{n_h}$ and $\mathbf{n_o}$ indicate the number of nodes in the input layer, hidden layer and output layer respectively. **K** is the number of tags that can be used as an attribute to describe a word in a corpus, $\mathbf{n_i = k(R+L+1)}$ where R and L are lengths of the left and right context for the target word $\mathbf{W_T}$. In this study, $\mathbf{k}$ =24, $\mathbf{n_o}$= 5 but $\mathbf{n_h}$, **R and L** are determined after the experiment. So in this study, the following five samples are treated to determine the appropriate value of **R and L**.

- **Sample 1**: An input vector with **48**-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) .**

- **Sample 2**: An input vector with **72**-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).**
- **Sample 3**: An input vector with **72**-elements for a target word and two words immediately before it as its context. This can be represented as **previous word_ target word _ Next word(1_T_1; L=1 and R=1).**
- **Sample 4**: An input vector with **96**-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).**
- **Sample 5**: An input vector with **120**-elements for a target word and two words immediately before it and two words immediately after it as its context. This can be represented as **second previous word_ first previous word_ target word_ first next word_ second next word (2_T_2; L=2 and R=2).**

5.6 Representation of Out Put for MLP_Tagger

Since the problem of tagging at word level can be posed as a classification problem, there are 24 classes (tag sets) in this study. All tags are given numeric codes (decimal number) and the corresponding numeric code for a tag is changed in to 5-digit binary number (since the number of tags in this study is 24,they cab be represented by 5-digit numbers). These classes (tags) are represented using binary numbers and decimal numbers as **shown below in table** 5.5

NO.	Type of tag	Binary representation	Decimal representation
1	N .	11111	31
2	NV	11110	30
3	NVP	11101	29
4	NP	11100	28
5	NB	11011	27

6	NC	11010	26
7	CRD	11001	25
8	ORD	11000	24
9	J	10111	23
10	JN	10110	22
11	JPN	10101	21
12	JP	10100	20
13	JC	10011	19
14	JNU	10010	18
15	ADV	10001	17
16	ADVC	10000	16
17	REL	01111	15
18	C	01110	14
19	PRE	01101	13
20	V	01100	12
21	VCO	01011	11
22	VP	01010	10
23	VC	01001	9
24	AUX	01000	8

Table 5.5 Binary and decimal number representation of tags.

The output **OUT** for MLP_Tagger can be represented as follows:

OUT=(O_1, O_2, O_3, O_4,O_5)------(5.3) and this can also be illustrated as follows.

1. F (target word, context)=N if OUT=(11111)=31.
2. F (target word, context)=NV if OUT=(11110)=30.
3. F (target word, context)=NVP if OUT=(11101)=29.
4. F (target word, context)=NP if OUT=(11100)=28.
5. F (target word, context)=NB if OUT=(11011)=27.
6. F (target word, context)=NC if OUT=(11010)=26.

7. F (target word, context)=CRD if OUT=(11001)=25.
8. F (target word, context)=ORD if OUT=(11000)=24.
9. F (target word, context)=J if OUT=(10111)=23.
10. F (target word, context)= JN if OUT=(10110)=22.
11. F (target word, context)=JPN if OUT=(10101)=21.
12. F (target word, context)=JP if OUT=(10100)=20.
13. F (target word, context)=JC if OUT=(10011)=19.
14. F (target word, context)=JNU if OUT=(10010)=18.
15. F (target word, context)=ADV if OUT=(10001)=17.
16. F (target word, context)=ADVC if OUT=(10000)=16.
17. F (target word, context)=REL if OUT=(01111)=15.
18. F (target word, context)=C if OUT=(01110)=14.
19. F (target word, context)=PRE if OUT=(01101)=13.
20. F (target word, context)=V if OUT=(01100)=12.
21. F (target word, context)=VCO if OUT=(01011)=11.
22. F (target word, context)=VP if OUT=(01010)=10.
23. F (target word, context)=VC if OUT=(01001)=9.
24. F (target word, context)=AUX if OUT=(01000)=8.
25. Other Wise F (target word, context)=UNW (unknown category).

7. F (target word, context)=CRD if OUT=(11001)=25.
8. F (target word, context)=ORD if OUT=(11000)=24.
9. F (target word, context)=J if OUT=(10111)=23.
10. F (target word, context)= JN if OUT=(10110)=22.
11. F (target word, context)=JPN if OUT=(10101)=21.
12. F (target word, context)=JP if OUT=(10100)=20.
13. F (target word, context)=JC if OUT=(10011)=19.
14. F (target word, context)=JNU if OUT=(10010)=18.
15. F (target word, context)=ADV if OUT=(10001)=17.
16. F (target word, context)=ADVC if OUT=(10000)=16.
17. F (target word, context)=REL if OUT=(01111)=15.
18. F (target word, context)=C if OUT=(01110)=14.
19. F (target word, context)=PRE if OUT=(01101)=13.
20. F (target word, context)=V if OUT=(01100)=12.
21. F (target word, context)=VCO if OUT=(01011)=11.
22. F (target word, context)=VP if OUT=(01010)=10.
23. F (target word, context)=VC if OUT=(01001)=9.
24. F (target word, context)=AUX if OUT=(01000)=8.
25. Other Wise F (target word, context)=UNW (unknown category).

CHAPTER SIX. THE EXPERIMENT

In this research work, the problem of part of speech tagging is undertaken by using **Artificial Neural Network** technique. Moreover, The specific neural network software that is used for model building and testing purposes in this study is **BrainMaker** software. Thus, before going to the details of specific steps that are carried out in this study, an overview of **BrainMaker** soft ware is presented below.

6.1. Brain Maker Neural Network Software

Brain maker Neural Network software is developed by California Scientific Software (http://www.calsci.com). This software uses back propagation algorithm in developing neural network model. The network is trained by presenting a set of facts (records) over and over again. Brainmaker goes through all the training lists (records) addressing each fact (record) in turn and making necessary corrections. After the entire list of facts has been presented (when one epoch or one run is completed), BrainMaker starts again from the beginning of the list. The training process is repeated until the network gets all facts (records) correct or until training is interrupted.

Brain maker Neural Network software has two programs called NetMaker and BrainMaker. Netmaker can import data file from any of the popular modes (like Lotus, ASCII ,Excel,Text,dBase,Binary,etc). Both numeric data and text data can be accepted by netmaker and converted in to a representation that the neural network can understand.i.e. In the range of **[0, 1].** The-imported files are seen on netmaker as a spreadsheet as shown below

Maker - sampledta.txt

Column Row Label Number Symbol Operate

	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed	NotUsed
	words	P1	P2	P3	P4	P5	N	NV	NVP	NP	NB	NC	V	AUX	VCO
1	ityoPya/	1	1	1	1	1	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
2	beSwoc/	1	1	1	0	0	0.3158	0.0263	0.0263	0.0263	0.0263	0.0263	0.0263	0.0263	0.0263
3	yemiqoT	0	1	0	1	0	0.0357	0.0357	0.0357	0.1071	0.0357	0.0357	0.0357	0.0357	0.0357
4	ametat/N	1	1	1	1	1	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037
5	wedehua	1	0	0	0	1	0.1667	0.0333	0.0333	0.0333	0.0333	0.0333	0.0333	0.0333	0.0333

Figure 6.1 NetMaker Screen with sample Data

After importing the data set in to NetMaker, the input fields (independent variables) and pattern fields (dependent variables) were determined and labeled. For this research, the independent variable is features of a target word and its context while a pattern(dependant variable) is the classification field that indicates whether the tag is **N**(Noun),**V**(verb),etc.In addition, a field can be labeled as annotations to represent fields that are not used as input or pattern but as an identification of a particular record. In this study the field which is labeled as words is used as annotation. The above-prepared file is then saved with a ***.dat extension as shown below.***

NetMaker - sampledta.DAT

File Column Row Label Number Symbol Operate

	Annote	Pattern	Pattern	Pattern	Pattern	Pattern	Input	Input	Input	Input	Input	Input	Input	Input	Input
	words	P1	P2	P3	P4	P5	N	NV	NVP	NP	NB	NC	V	AUX	VCO
1	ityoPya/	1	1	1	1	1	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
2	beSwoc/	1	1	1	0	0	0.3158	0.0263	0.0263	0.0263	0.0263	0.0263	0.0263	0.0263	0.0263
3	yemiqoT	0	1	0	1	0	0.0357	0.0357	0.0357	0.1071	0.0357	0.0357	0.0357	0.0357	0.0357
4	ametat/N	1	1	1	1	1	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037	0.037
5	wedehua	1	0	0	0	1	0.1667	0.0333	0.0333	0.0333	0.0333	0.0333	0.0333	0.0333	0.0333

Figure 6.2 Preparing BrainMaker Network file using NetMaker Tool.

The BrainMaker program has the facility where data are classified in to training and testing sets.. So, The above ***. dat** file is basis to create the three brain maker files. The first BrainMaker 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 taining. The last one is running fact file **(*.in)** created to predict future records.

After the above files are created the model developer can move from Netmaker program to the Brain Maker Program.

The **Brain Maker** Program has different parameters with different possible values for each parameter that are essential in neural network training and testing. The most important parameters are training tolerance, learning rate, smoothing factor (momentum), number of hidden layers, number of neurons in hidden layer, type of function, etc). After determining these parameters, training is started by using the **Operate/Train Network** command. While training progresses statistical information are provided on the screen such as which fact the BrainMaker is processing at a specific time, the number of facts which met and did not meet the training tolerance, the number of run (epoch) etc. There are also two graphs that display the progress of training. The first is a histogram that shows the distribution of error over an entire run. The horizontal axis represents the error level and the vertical axis signifies the number of out put values at that particular level. As training progresses and fewer facts are classified as incorrect, the bars (solid boxes) move to the left. The second graph 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 represent the over all error level (root mean square error, RMS). For a good training, the value of RMS would decrease as the number of runs increases as shown below.

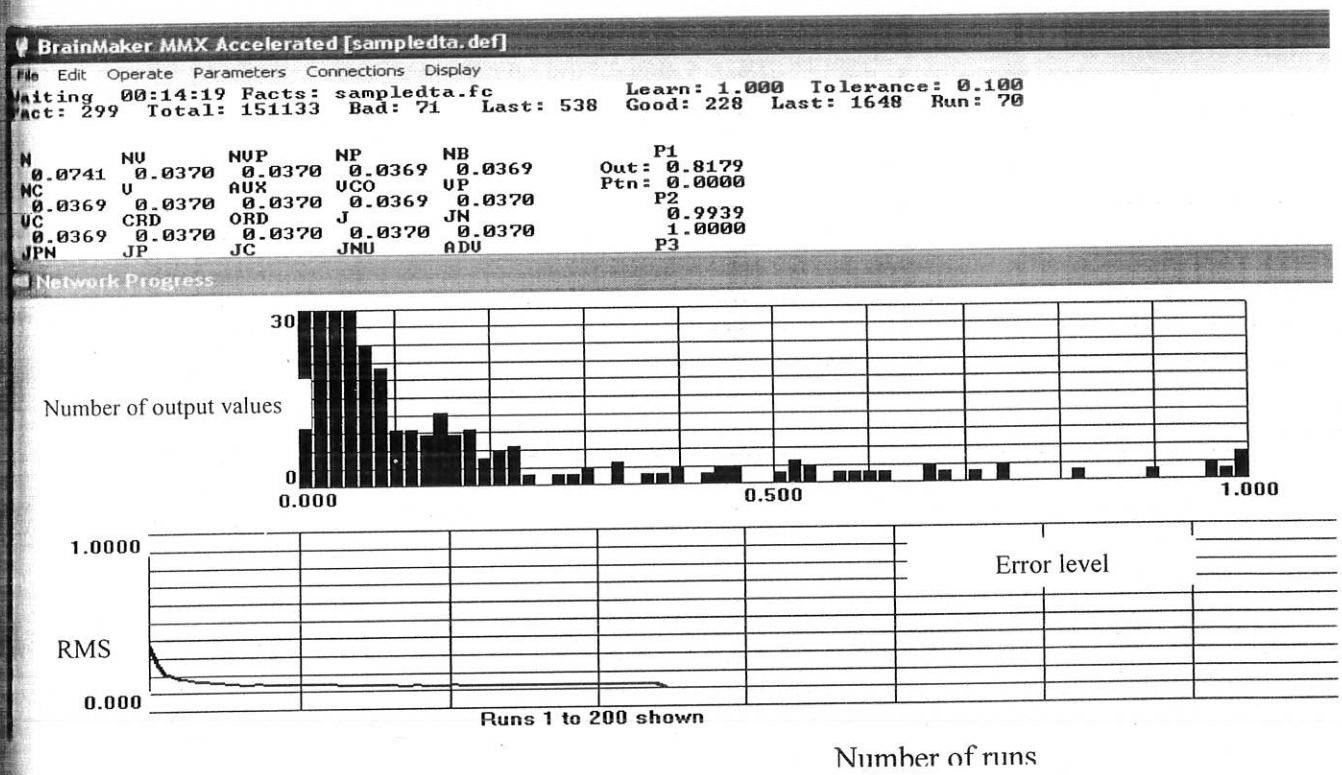


Figure 6.3 BrainMaker screen while training.

Training can be stopped at any time before the instruction to stop training is met. The default for stopping training process is when the incorrect classification of facts in a single run (epoch) is zero. So better network model can be obtained before the criteria for stopping training are met. Hence, it is advisable for model developer to test and save a network model periodically.

6.2 Experiments With Brain Maker Software

This section discusses the experiment conducted using **Brain Maker** neural network software for the purpose of predicting the type of tag for each word in a given sentence. In the discussion emphasis is given to assess the outputs produced and the test results obtained. In this experiment a corpus with 159 sentences is used. These sentences were randomly divided in to two sets. One with 136 sentences for training and the other with 23-sentences for testing. The traning set contained 2429 words without including

punctuation marks. The testing set contained 392-words without including punctuation marks. The procedures to train and test a neural network model for part of speech tagger (MLP) can be represented in the following flow chart (Berhanu, 1999).

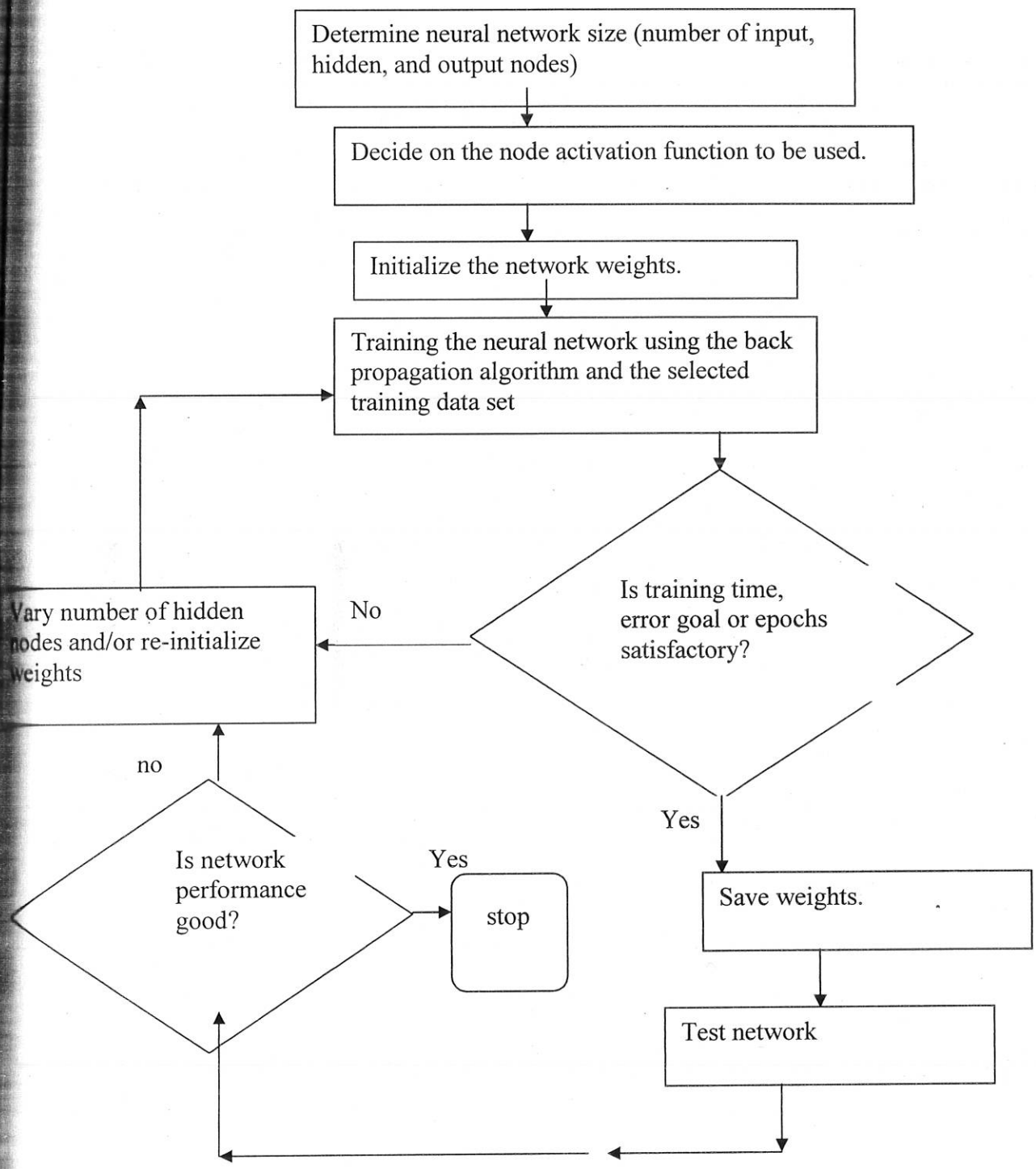


Figure 6.4 Neural Network Training And Testing Procedure

Using this procedure, the number of input nodes was determined as **48, 72, 96, and 120** for samples one, two, three, four, and five respectively as described in section **5.2**. The number of output nodes was **5** for each type of sample.

The first training trial is conducted by using **sample one** (an input vector with **48-**elements to represent a target word and one word immediately before it as its context **(Previous word_ Target word).**

To build the neural network model, first the data set which is in text format is imported in to **NetMaker** program. This prepared data set contains 2429 sample records. After importing the data set in to NetMaker, the input fields (features of a target word) , pattern fields (parts of speech tags) and the annotated field (words) were determined as shown in figure 6.2 above.

After labeling the fields as input, pattern and annotation, the file is saved with a ***.dat*** *extension. Then* running fact file **(*.in)** , definition file **(*.def**)and training file or fact file **(* .fct)** are created. After these files are created, the model developer moves from **Netmaker** program to the **Brain Maker** Program.

Initially, training was 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 layer=**48** which is the same as number of nodes in the input layer by default; type of neuron function=**sigmoid**; network weights=small randomly chosen

umbers and gavethe following progress.

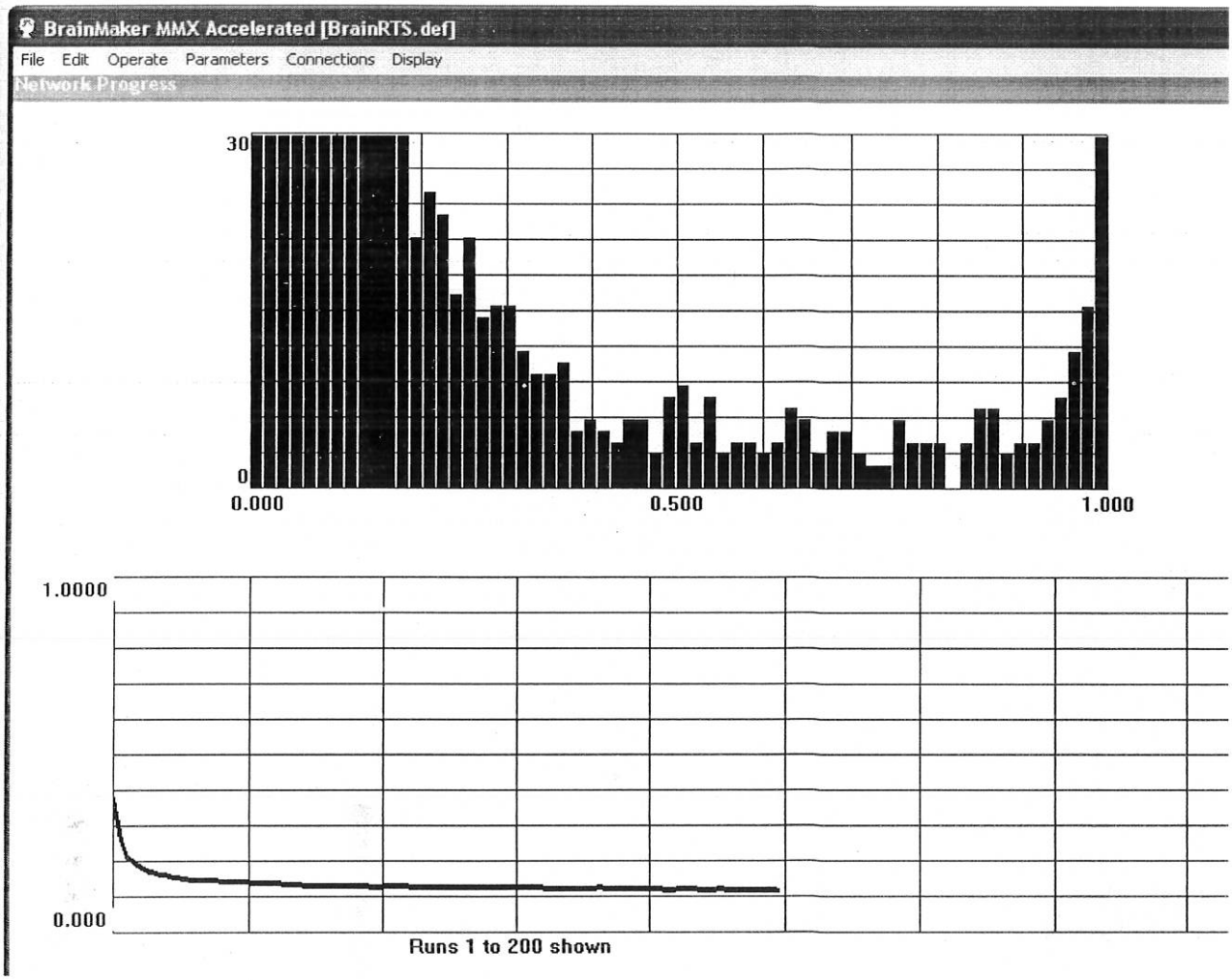


Figure 6.5 Progress Of The Neural Network On The Training Data Set Using The Default Values For Different Network Parameters.

The training Process can be interrupted when the distribution of errors and the root mean square error value stopped to decrease as shown in the figure 6.6 above. The above model constructed using the default values predicts 361 tags correctly out of 392 tags. Thus, the model works with an accuracy of **92.09%**. Since the test result of the above network model was encouraging, the researcher continued the experiment 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 performing fairly well in testing process, try to build

another network model by changing number of nodes in the hidden layer, by adding noise to the network, by changing number of hidden layers, by changing training tolerance and learning rate etc (California Scientific Software, 1998).

Thus, **first** it was 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 (i.e. it predicts 355 tags correctly out of 392 tags and the model works with an accuracy of **90.56%**).

Second, by keeping the default value for other parameters constant, the default values of training tolerance was changed in to **0.01.** But in this experiment, the default value of 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 (i.e. it predicts **366**tags correctly out of **392** tags and the model works with an accuracy of **93.37%). Again when the training tolerance was changed in to 0.001,** it resulted in a little bit better performance than the network model under default values (i.e. it predicts 365 tags correctly out of **392** tags and the model works with an accuracy of **93.11 %).**

6.3. Other Experimental Results.

Generally, numerous experiments were carried out by varying the values of parameters on the basis of suggestions made by the vendors of this software and workers in this area. From all the above experiments for sample one (**1_T_0**), it became clear that varying of default values was not resulting in a significant change to the performance of the network model. This is also true for samples two, three, four and five as shown in the following table.

Sample Typ	HL=1 HN=D TT=D=0.1	HL=2 HN=D ; HL=2; TT=D=0.1	HL =D=1 HN=D TT=0.01	HL=D=1 HN=D TT=0.O01
1_T_0	92.09%	90.560%	93.37%	93.11%
2_T_0	92.35%	90.31%	92.60%	91.32%
1_T_1	93.88%	93.620%	93.38%	94.39%
2_T_1	93.62%	92.86%	92.60%	92.09%
2_T_2	91.84%	78.06%	82.91%	85.97%

Table 6 Performances of Network Models For Different Values of Parameters.

Legend: HL=number of hidden layer; **HN**=number of nodes in the hidden layer; **TT**=training tolerance; **D**=default value.

6.4. Conclusion

In this thesis, an MLP-tagger was presented for Amharic language. The current work uses neighbourhood context (localized information) for tagging of words in a small tagged corpus .The MLP-tagger combined with the representation scheme for inputs and outputs could be a promising result for further research in the same direction for Amharic language corpora. Changing the defualt values of different parametres 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 sample **1-T-1** is selected and gave an average accuracy of **93.880%** on words in test data that are not seen during the training process. Hopefully, the accuracy will increase if it is tested with a large data. The next chapter closes this thesis by providing summary and recommendation.

CHAPTER SEVEN

SUMMARY AND RECOMMENDATION.

7.1. Summary

Text corpora that are tagged with part of speech information are useful in many areas of linguistic research including speech recognition, speech synthesis, machine translation and information retrieval. The purpose of this study is also to test the potential application of Multilayer Perceptron Network for tagging parts of speech for Amharic Language. Rule-based, stochastic and Artificial Neural networks approaches for part of speech tagging problem were briefly reviewed. The purpose of this study is also to test the potential application of Multilayer Perceptron Network for tagging parts of speech for Amharic Language. A discussion on Amharic word classes was done to design the tag set. The tags are general because they do not show morpho-syntactic feature like gender, number, person, case etc. The step-by-step procedures followed to prepare lexical probability matrix were discussed and a database of lexical probability is designed to generate an input vector in to MLP-tagger to determine a tag for a target word. Experiments were conducted using BrainMaker Software by varying the number of words which precede and succeed a target word and also by varying the default values of network parameters .The percentage of correct tag assignment was used to measure the performance of the MLP-tagger.

Even though the accuracy of the MLP-tagger in this study is somewhat acceptable, it may not have an immediate practical application because the MLP-tagger was not trained on large quantities of data. This is due to lack of standard corpora annotated with part of speech tags in the language. The researcher did not generate such quantities of data because of the following reasons.

- Manual tagging is laborious, expensive and time consuming.
- Financial constraint to prepare such type of huge date.
- The scope of the research is to demonstrate the application of MLP-tagger for Amharic language.

The research has indicated the possibility of developing an automatic MLP-tagger for Amharic language provided that a standard corpus for training is available.

7.2 Recommendation.

This research has some shortcomings. Thus, the following are suggested for future research work.

- Replicate the research for other local languages by adopting the procedures followed in this study.

- The problem of part of speech tagging can be done at two levels: word level and sentence level. In the case of word level, the problem can be posed as a classification problem and the context provided by neighbouring words influences the tag assignment for a target word. Where as, in the case of sentence level tagging, the context provided by the whole sentence influences the tag assignment for a target word. In this study, the MLP-tagger addresses the classification problem posed by word level part of speech tagging. Since the context provided by the whole sentence has more information than the context provided by neighbouring words, the efficiency of MLP tagger can increase at sentence level tagging. Hence, it is recommended to do further research at sentence level of tagging for Amharic language.
- Since a MLP network model for part of speech tagging was built on a small training and testing data, the accuracy reported is valid only for this data. Therefore, replicate this work using a large corpus, so that it is possible to estimate the correct effectiveness of the MLP-tagger and its potentiality for applying it for Amharic language.
- Experts in the area of Amharic language should be encouraged to tag large quantity of data manually.
- Because of convergence and over training problems, it is impossible and also not advisable to train neural nets to an accuracy of 100%. The training should be stopped at an appropriate level of accuracy. Consequently, neural network

may not acquire some useful rules. Hence, to solve this problem, replicate this work by incorporating a rule-based corrector module as a post processor (a hybrid system of neuro-tagger and rule-based corrector. First the neuro-tagger out put a tagging result for the target word and then the rule-base corrector corrects the output of the neuro-tagger and gives the final tagging result).

- Replicate this work using other machine learning approaches (like Recurrent Neural Networks, decision tree, constraint grammar using inductive logic programming) and stochastic approaches (bi-gram and thr-gram) to compare their performance.
- In order to see the impact of tag set size on the performance of MLP-tagger, make part of speech tags very fine by incorporating morpho-syntactic features like number, gender, person, case, tense, and so on.
- Part of speech tagger is not an end by itself. it is a way to an end. Therefore, further studies in the language like parsing, speech recognition and synthesis, machine translation and information retrieval should incorporate part of speech tagger as a module to use the outputs of it.

REFERENCE

Abiyot Bayu (2001). *Developing Automatic Word Parser for Amharic Verbs and and Their Derivation*, Master Thesis. Addis Ababa University.

Allen, Jemes (1995). *Natural language understanding .2nd ed.* The Benjamin/Cummings publishing Company, Inc; California.

Ahmed(2002).*Application of Multilayer Perceptron Network For Tagging Parts of speech.* Language Enginnering Conference (57-63);IEEE, Inc; California.

Atelach Alemu (2002). *Automatic Morphological Analyzer for Amharic Text: An Experiment Using Probabilistic context free Grammers,* Masters Thesis, Addis Ababa University.

Atelach Alemu, Lars Asker, and Mesfin Getachew(2003).*Natural Language Processing for Amharic: Overview and suggestions for a way forward.* In Tritement Automatique des langues naturelles, Batz-sur-Mer, 11-14 juin 2003.

Baye Yimam (1986).*YeamareNa sewasew*; E.M.P.D.A, Addis Ababa.

Bender, M.L (1974). *Amharic Verb Types From Text and lexicon. Folia Oriental* 15, (23-46).

Bender, M.L and Hailu Fulas (1978). *Amharic Verb Morphology.* Michigan State University.

Berhanu Aderaw (1999). *Amharic Character Recognition Using Artifial Neural Networks*. Masters Thesis Addis Ababa University.

Dawkins, C.H (1969). *The fundamentals of Amharic*, Addis Ababa Sudan Interior Mission.

Dereje Teferi (1999). *Optical Character Recognition of Type Written Amharic Text.* Master Thesis. Addis Ababa University.

California Scientific Software (1988). BrainMaker, Users Guide and Reference Manual.

Forcada, Mikel L. and P'erez-Ortiz, Juan A. (no year). *Part of speech tagging With Recurrent Neural Networks*. AT http://www.dlsi.va.es/~japerez/pub/pdf/ijcnn2001.pdf.

Getahun Amare (1989). *Zemenawi yeamareNa sewasew beqelal aqerareb*. Commercial Printing Press, Addis Ababa.

Girmay B. (1992). *Word Formation in Amharic*. Journal of Ethiopian Language and Literature.No.2,50-74

Guilder, Linda V. (1995). *Automated Part of Speech Tagging: A Brief Overview*. At:

Vanguill@gusun.georgetown.edu.

Haykin, Simon (1999). *Neural Networks, a Comprehensive Approach*. Prentice-hall, Inc. USA.

Leslau, Wolf (1965). *An Amharic Text Book of Every Day Usage*. University of California, Los Angles.

Leslua, Wolf (1967). *Amharic Text Book*. Otto Harrassowitz, Wiesbaden, Belgium.

Ma, Qing (2003). *Natural Language Processing With Neural Networks*.Language Enginnering Conference(45-56);IEEE,Inc;California.

Megyesi, Bea'ta(2002).*Data-Driven Syntactic Analysis Methods and Applications for Swedish*. Doctoral Dissertation.Universitetsservice US AB. Sweden

Mersehazen Woldeqiros (1934). *yeamareNA sewasew*. Birhan Selam Prining Press. Addis Ababa.

Mesfin Getachew(2002).*Automatic Part of Speech Tagging for Amharic Language:An Experiment Using HMM Approach.* Masters Thesis, Addis Ababa University.

Mitchell,Tom(1997).*Machine Learning*. The McGraw-Hill Companies,Inc.New York

Nakagawa, Tetsuji and etal (no year). *Unknown Word Guessing and Part of Speech Tagging Using Support Vector Machine*. At: http://www.afnlp.org/nlprs2001/pdf/0053-01.pdf

Nega Alemayehu (1999). *Development of Stemming Algorithm For Amharic Texts Retrieval.* PHD Dissertation at university of Shifeld.

Roberts, Andrew (2003). *Machine Learning in Natural Language Processing* At: http://www.com.leeds.ac.uk/andyr/misc/latex/sessions/bibtex/bib-example.pdf.

Russell,S.J.and Norving,Peter(2003).*Artificial Intelligence:A Modern Approach.2nd* Edition. Prentice Hall.USA.

Salton, G. (1983). *Introduction to Modern Information Retrieval*. McGraw-hill Company. New York

Saracevic,Tefko(1999).*Information Science*. Journal of the American Society for Information Science.Vol.50 (12): 1051-1063.

Schmid, Helmut (no year). *Part of speech Tagging With Neural Networks*. At: http://acl.ldc:upenn.edu/c/c94/c94-1027.pdf

Worku Alemu (1997). *The application of OCR Techniques to Amharic Script*. Master Thesis. Addis Ababa University.

Yaregal Assabie (2002). Development of Versatile Character Recognition System for Amharic Text. Master Thesis. Addis Ababa University.

Appendix 1: Character Representation Used in the Transcription

ሀ	he	ሁ	hu	ሂ	hi	ሃ	he	ሄ	hE	ህ	h	ሆ	ho
ለ	le	ሉ	lu	ሊ	li	ላ	la	ሌ	lE	ል	l	ሎ	lo
ሐ	he	ሑ	hu	ሒ	hi	ሓ	he	ሔ	hE	ሕ	h	ሖ	ho
መ	me	ሙ	mu	ሚ	mi	ማ	ma	ሜ	mE	ም	m	ሞ	mo
ሠ	se	ሡ	su	ሢ	si	ሣ	sa	ሤ	sE	ሥ	s	ሦ	so
ረ	re	ሩ	ru	ሪ	ri	ራ	ra	ሬ	rE	ር	r	ሮ	ro
ሰ	se	ሱ	su	ሲ	si	ሳ	sa	ሴ	sE	ስ	s	ሶ	so
ሸ	Se	ሹ	Su	ሺ	Si	ሻ	Sa	ሼ	SE	ሽ	S	ሾ	So
ቀ	qe	ቁ	qu	ቂ	qi	ቃ	qa	ቄ	qE	ቅ	q	ቆ	qo
በ	be	ቡ	bu	ቢ	bi	ባ	ba	ቤ	bE	ብ	b	ቦ	bo
ተ	te	ቱ	tu	ቲ	ti	ታ	ta	ቴ	tE	ት	t	ቶ	to
ቸ	ce	ቹ	cu	ቺ	ci	ቻ	ca	ቼ	cE	ች	c	ቾ	co
ነ	ne	ኑ	nu	ኒ	ni	ና	na	ኔ	nE	ን	n	ኖ	no
ኘ	Ne	ኙ	Nu	ኚ	Ni	ኛ	Na	ኜ	NE	ኝ	N	ኞ	No
አ	a	ኡ	u	ኢ	i	ኣ	a	ኤ	E	እ	I	ኦ	O
ከ	ke	ኩ	ku	ኪ	ki	ካ	ka	ኬ	kE	ክ	k	ኮ	ko
ወ	we	ዉ	wu	ዊ	wi	ዋ	wa	ዌ	wE	ው	w	ዎ	wo
ዐ	a	ዑ	u	ዒ	i	ዓ	a	ዔ	E	ዕ	I	ዖ	O
ዘ	ze	ዙ	zu	ዚ	zi	ዛ	za	ዜ	zE	ዝ	z	ዞ	zo
ዠ	Ze	ዡ	Zu	ዢ	Zi	ዣ	Za	ዤ	ZE	ዥ	Z	ዦ	Zo
የ	ye	ዩ	yu	ዪ	yi	ያ	ya	ዬ	yE	ይ	y	ዮ	yo
ደ	de	ዱ	du	ዲ	di	ዳ	da	ዴ	dE	ድ	d	ዶ	do
ጀ	je	ጁ	ju	ጂ	ji	ጃ	ja	ጄ	jE	ጅ	j	ጆ	jo
ገ	ge	ጉ	gu	ጊ	gi	ጋ	ga	ጌ	gE	ግ	g	ጎ	go
ጠ	Te	ጡ	Tu	ጢ	Ti	ጣ	Ta	ጤ	TE	ጥ	T	ጦ	To
ጨ	Ce	ጩ	Cu	ጪ	Ci	ጫ	Ca	ጬ	CE	ጭ	C	ጮ	Co
ፀ	s'e	ፁ	s'u	ፂ	s'i	ፃ	s'a	ፄ	s'E	ፅ	s'	ፆ	s'o
ጰ	Pe	ጱ	Pu	ጲ	Pi	ጳ	Pa	ጴ	PE	ጵ	P	ጶ	Po
ፈ	fe	ፉ	fu	ፊ	fi	ፋ	fa	ፌ	fE	ፍ	f	ፎ	fo
ፐ	pe	ፑ	pu	ፒ	pi	ፓ	pa	ፔ	pE	ጩ	p	ፖ	po
ሏ	lua	ሟ	mua	ሯ	rua	ሷ	Sua	ሿ	Sua	ቧ	bua	ቷ	tua
ቿ	cua	ኗ	nua	ኟ	Nua	ኧ	Tua	ቯ	Cua	ዟ	zua	ዧ	Zua
ዷ	dua	ጇ	jua	ፏ	fua								

Appendix 2: Part of Sample Corpus

ItyoPya/N beSwoc/CRD yemiqoTeru/VP ametat/N wedehuala/ADV temelsen/V hulacnm/N yehegeritu/JPN zegoc/N lnaTEnew/VP,/PUC lnmeremrew/VP,/PUC kezam/NP alfen/V lnkorabet/REL Ina/C lntaferbet/REL yemnclbet/REL tarik/N yalat/V heger/N nat/AUX ./PUF yh/JP tarik/N zarE/ADV ityoPya/N bemnlat/VP heger/N wsT/PRE yetesebesebu/VP yeteleyayu/VP quanqua/N tenagariwoc/VP Ina/C yeteleyayu/VP bahloc/N telabaS/J yehonu/CP hzboc/N bemulu/NP beandnet/NP,/PUC begara/NP Ina/C beqeTaynet/NP siserut/VP yenore/VP tarik/N aydelem/AUX./PUF # lihon/VP yemiclm alneberem/AUX./PUF # beTam/J beqelalu/JP lilewaweT/VP yemicl/VP yegzat/JPN Ina/C yekll/JNP dnber/N beneberebet/REL,/PUC beTam/J beqelalu/JP liTenaker Ina/C lidakem/VP yemicl/VP yenegestat/JPN,/PUC yemesafnt/JPN Ina/C yebalabat/JPN astedader/N sefno/V benorebet/REL yetarik/JPN zemen/N ,/PUC beSwoc/NP lemiqoTeru/VP ametat/N bequaminet/NP sayqeyer/VP yenorebet/VP yegzat/JPN kll/N Ina/C yehzb/JPN sbTr/N yemiTebq/VP kale/VP ,/PUC betarik/JPN fit/N yeqome/VP yewah/J bCa/J new/AUX ./PUF # yihunna/C ,/PUC hzboc/N bequaminet/NP Ina/C beqeTaynet/NP lebzu/JNU ametat/N yemizelq/VP yegara/JP yegzat/JPN Ina/C astedaderawi/J Tla/N sr/PRE alneberum/AUX ./PUF # beSih/JNU ametat/N wede/PRE huala/N yemiqoTer/VP yegara/JP tarik/N linoracew/VP ayclm/V yemil/VP medemdemiya/N lay/PRE medresm/VC yannuyahl/J qilaqilnet/N yhonal/AUX./PUF #

DECLARATION

esis is my original work, has not been presented for a degree in
er university and all sources of material used for the thesis have
ly acknowledged.

Yenewondim Biadgie Sinshaw

March, 2006

esis has been submitted for examination with my approval as
sity advisor

Prof. B.R. Krishna Rao

March, 2006