ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

WORD SENSE DISAMBIGUATION FOR TIGRIGNA LANGUAGE USING SEMI-SUPERVISED MACHINE LEARNING APPROACH

By

TEKLAY MURUTS WELEMICHAEL

JUNE, 2018

ADDIS ABABA, **ETHIOPIA**

ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

WORD SENSE DISAMBIGUATION FOR TIGRIGNA LANGUAGE USING SEMI-SUPERVISED MACHINE LEARNING APPROACH

TEKLAY MURUTS WELEMICHAEL

A Thesis Submitted to School of Graduate Studies of Addis Ababa University in Partial Fulfillment of the Requirements for the Degree of Master of Science in Information Science

By: TEKLAY MURUTS WELEMICHAEL

Advisor: Solomon Teferra (PhD)

June, 2018

Addis Ababa, **Ethiopia**

ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

WORD SENSE DISAMBIGUATION FOR TIGRIGNA LANGUAGE USING SEMI-SUPERVISED MACHINE LEARNING APPROACH

By:

TEKLAY MURUTS WELEMICHAEL

Name and signiture of Members of the Examining Board

Name	**Signtaure**	**Date**
Solomon Teferra (PhD) Advisor	_____	_____
_____ Examiner	_____	_____
_____ Examiner	_____	_____

Declaration

This thesis has not previously been accepted for any degree and is not being concurrently submitted in candidature for any degree in any university.

I declare that the thesis is a result of my own investigation, except where otherwise stated. I have undertaken the study independently with the guidance and support of my research advisor. Other sources are well acknowledged by citations giving explicit references. A list of references is appended.

Signature: _____

Teklay Muruts

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

Advisor's Signature: _____

Solomon Teferra (PhD)

Acknowledgment

First and foremost, I would like to thank my Almighty God and St. Marry for giving me health, patience, and make everything possible.

Next, I would like to express my sincere gratitude to my advisor Dr. Solomon Teferra for his guidance, understanding, patience, punctuality, excellent supervision, and fatherhood treatments throughout this study. In addition to his academic advises, I have learnt what humanity means from him.

Beside to my advisor, I would like to thank Ato Getahun Wassie and Tsegaye Weledemariam for supporting me both in moral and material. I would also like to thank Hafte Abera and Mulugeta Atsbeha for helping me in collecting my corpus and Dureti Siraj for giving me some direction on how to collect corpus.

My special thanks go to my family, specially my mother Berihu Weldegebrieal, my uncle Abrha Weldegebrieal, and my uncle's wife Senbetu Gebre (Rocket), for helping me in everything. My uncle is my inspiration for being the person that I am now. Without you I wouldn't be the person that I am now. You become both as a father and as an uncle. You teach me to become a hard worker in everything. I don't have any words to express my feeling about you, I can only say thank you so much and long live for you.

I also want to say thank you for my friend, my classmate, and my staff member Melaku Yenew for being with me at any time. It is better to say you my brother rather than my friend.

The last but not the least, I would like to say thank you for my friends Teklay Hailu, Tesfay Alemayehu, Haftom Berhe, Kibrom Kidanu, Fikru Tafesse, Alema Gebru, Yibrah Goitom, Gebreyesus Gebrerufael, Tewelde Welu, Tesfay Gebreslasie, Teklay Abrha, Tekeste Araya, and all of my classmates.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI	Artificial Intelligence
BNC	British National Corpus
EM	Expectation Maximization
IE	Information Extraction
IR	Information Retrieval
KNN	K-Nearest Neighbor
MRD	Machine Readable Dictionary
MT	Machine Translation
NER	Named Entity Recognition
NLP	Natural Language Processing
NLU	Natural Language Understanding
SERA	System for Ethiopic Representation in ASCII
OALD	Oxford Advanced Learner's Dictionary
POST	Part of Speech Tagging
QA	Question Answering
SQL	Standard Query Language
SVM	Support Vector Machine
TRTO	Tigray Radio and Television Organization
VOA	Voice of America
WSD	Word Sense Disambiguation

Abstract

Developing different natural language applications can be used to make the communication between humans and computers easy. Most natural languages have words having different meanings depending on different contexts. Those words make the communication between computers and humans difficult because computers cannot differentiate (identify) proper meanings of ambiguos words. Therefore, Word Sense Disambiguation (WSD) enables computers identify the proper meaning of the ambiguous words depending on the surrounding contexts.

In this study, we tried to build WSD prototype model for Tigrigna language because WSD is an intermediate task for other NLP tasks like Machine Translation, Information Retrieval systems, Information Extraction, Speech Processing, etc. WSD can be developed by using corpus-based, knowledge based, and hybrid approaches. From those approaches, we used a corpus-based approach to build WSD prototype model. Corpus-based approach is supported by machine learning methods. Corpus-based approaches can be classified as supervised, semi-supervised, and unsupervised machine learning methods. Since semi-supervised machine learning method narrows the weakness of both supervised and unsupervised machine learning methods by exploiting many unlabeled datasets with small labeled datasets, we worked on semi-supervised machine learning method to build WSD prototype model of Tigrigna Language.

We conducted our experiments on five ambiguous words of Tigrigna by collecting a total of 1250 sentences of the language. Those five ambiguous words of the language are: - kefele(ክፈለ), OareQe (ዓረቐ), seOare (ሰዓረ), Halefe (ሓለፈ), and medeb (መደብ). The first three words have two senses of each, and the fourth and fifth words have four and three senses respectively. We applied four clustering algorithms (EM, Simple K-Means, FarthestFirst, and HierarchicalClusterer) and five classification algorithms (ADTree, AdaBoostM1, Bagging, SMO, and Naïve Bayes) for clustering and classification purposes of the sentences into their senses respectively. Since those algorithms are available in WEKA 3.8, we used this tool in our study. We compared the three machine learning methods; and found out that semi-supervised machine learning achived the best performance. We achieved an average performance of 93.6636%, 91.9224%, 85.35%, 79.1917%, and 70.8968% using ADTree, SMO, AdaBoostM1, Bagging, and Naïve Bayes algorithms respectively. Window size of 1-1 became the optimal window size to identify the meaning of the selected ambiguous words of Tigrigna language using all of the selected classification algorithms.

Chapter One

1. Introduction

1.1 Background

Nowadays, the growth of information technology has lead the way for a large volume of information to be available for the society. Discussion about importance of a language for using the available information is not far from obvious since it serves as a medium of communication among the races. Language has a potential to express a wide range of ideas and to convey complex thoughts. In particular, natural language is now being used to exchange information among humans and has now reached to the extent of being evolution criteria for technology. In order to make available information useful for the society, an interest has emerged to make use of technology to process natural language. In response to such a need, NLP has come up with an indispensable focus of natural language computations [1].

Natural Language processing is a field of computer science which deals with the interaction among computers and humans using natural language that aims to improve human to human communication and human to computer communication [2]. It is normally used to describe the function of software or hardware components in a computer system which analyzes or synthesizes spoken or written language [3]. One of the challenges inherent in natural language processing is teaching computers to understand the way humans learn and use languages [14]. Natural Language Processing is a theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications. There are in fact two distinct focuses of NLP: language processing and language generation. The first one refers to the analysis of language for the purpose of producing a meaningful representation, while the latter refers to the production of language from a representation [4].

The field of NLP was originally referred to as Natural Language Understanding (NLU) in the early days of AI. It is well agreed today that while the goal of NLP is true NLU, that goal has not yet been accomplished. A full NLU System would be able to: paraphrase an input text, translate the

text into another language, answer questions about the contents of the text and Draw inferences from the text [4]. The aim of NLP is studying problems in the automatic generation and understanding of natural languages. Natural language is understood as a tool that people use to express themselves and has specific properties that reduce the efficiency of textual information retrieval systems. These properties are linguistic variation and ambiguity. Natural language processing (NLP) is also a subfield of artificial intelligence and linguistics [5]. NLP refers to Human-like language processing which reveals that a discipline within the field of Artificial Intelligence (AI). Moreover, NLP is the means for accomplishing different types of tasks and/or applications; such tasks include Part of Speech (POS) Tagging, Named Entity Recognition (NER), Information Retrieval (IR), Speech Recognition, Machine Translation, Question Answering, Word Sense Disambiguation, etc. [4]. Ambiguity, one of the greatest challenges in NLP, is the term used to refer for understanding of something in two or more possible ways or something which has more than one meaning. It can appear in sentence or clause level (called structural or syntactic ambiguity) or at a word level (called lexical ambiguity) and phonological ambiguity. Ambiguity is a universally recognized linguistic phenomenon which arises from the structure of the language and can be explained in terms of the analysis at different levels [6].

When humans process natural language, they have the capability to overlook the ambiguities which is hardly possible for machines or computers because machines run according to the instructions. In natural language, most words have more than one lexical meaning. Those words which involve multiple meanings or senses are called polysemous words. Polysemous words can be described in several different ways based on the context in which they occur. The context of polysemous words will be determined by the other neighboring words which is called local or sentential context [7]. For example, let's take Tigrigna word **መደብ (medeb)** which has three different meanings i.e. "plan", "traditional bed" and "assigning or grouping". Look the following three sentences: -

1. ኣብ ጥቅምቲ ንኽምርያ **መደብ** ነበሮ ።
2. ብዕራይ ሓሪዶም እቲ ስጋ **ብመደብ** መዲቦም እዮም ተኻፊሎም ዝውዕሉ ።
3. እማሆይ ወዲቀን እግረን ምስተሰበራ ኣብ **መደብ** ንክልተ ሰሙን ደቂሰን።።

In the above sentences the meaning of the word "**መደብ" is "plan"," grouping or Assigning**", and "**traditional bed**" respectively. Here the meaning of the word "**መደብ"** was determined

depending on the words that surrounds it. Therefore, Tigrigna is one of the natural languages which contains a lot of its own words having more than one lexical meanings or senses.

To solve this ambiguity, Word Sense Disambiguation is essential in natural language processing. In the field of computational linguistics, word sense disambiguation is defined as the problem of computationally determining which "sense" of a word is activated by the use of the word in a particular context. Lexical disambiguation in its broadest definition is nothing less than determining the meaning of every word in context, which appears to be a largely unconscious process in people. Most words in natural language have more than one lexical meaning or sense, but usually only one of them is active in a given context. Fundamentally, WSD deals with choosing the correct sense (i.e., meaning) of a word in a given text from a list of possible senses based on the content. The correct sense of the word is made clear based on the context in which the word has been used. The process of disambiguation for the different meanings of polysemous word relies on the context of the target word and also analysis of the properties exhibited by that context [8]. Using WSD as an intermediary step for other natural language applications is better than using as end by itself. Machine translation (MT) is challenging task without some form of disambiguation; WSD is also helpful for information retrieval (IR), information extraction (IE) QA systems, etc. [9] [10].

Word sense disambiguation can be developed by using three main approaches. These approaches are knowledge based, corpus-based, and hybrid approaches. Knowledge based uses external sources as otology information [11]. Corpus-based approach is machine learning approach that is farther subdivided into unsupervised, semi-supervised and supervised machine learning algorithms. Supervised learning method is a classification technique which requires a predefined human annotated data with class whereas unsupervised learning paradigms which require unlabeled data to cluster based on data similarity. Both supervised learning and unsupervised learning paradigms have their own limitations to achieve good classification and clustering results, but their limitations are complementary. Supervised method requires external teachers to label the training data which is laborious (expensive), subjective and time taking. On the other hand, the unsupervised method yields significantly lower accuracy and produce results that are not satisfying for many applications and often drive sets of word senses that are not intuitive to human's due to the absence of human intervention for label annotation. Semi-supervised learning method falls in

the middle between unsupervised learning and supervised learning that make use of both a small amount of labeled and a large amount of unlabeled data for training [9]. Hybrid approach is the combination of corpus-based approach and knowledge-based approach [12].

Due to the importance of WSD for understanding semantics and many real-world applications; researchers have been interestingly trying to tackle that problem. Therefore, the goal of this research is to develop a WSD prototype model for Tigrigna language using semi-supervised learning method to extract training sets which minimizes amount of the required human intervention and to produce considerable improvement in learning accuracy of the data sets by preparing large number of unannotated datasets and small number of annotated datasets.

1.2 Statement of the problem

Tigrigna language is a Semitic language widely spoken in the Tigray region of Ethiopia and in the whole nation of Eritrea. Tigrigna is spoken by large immigrant communities around the world, including Sudan, Saudi Arabia, the United States, Germany, Italy, the United Kingdom, Canada and Sweden. Tigrigna is written in the Ge'ez script which these days is called Ethiopic script and originally developed for Ge'ez language. This language has more than six million speakers worldwide. As a result, Tigrigna documents are increasing in size from time to time [13]. This shows that there are large collections of Tigrigna documents available in web, in addition to hard copy document in library, and documentation centers. Conducting researches for this language can be mostly important for Tigrigna language speakers.

Word Sense Disambiguation remains one of the most complex problems facing computational linguists to date. Despite the advances in natural language processing (NLP), WSD is still considered one of the most challenging problems in the field [14]. Developing word sense disambiguation is crucial for latter development of other natural language applications such as Information Extraction, Machine Translation, Speech Recognition, Information Retrieval, Question Answering, Text Summarization and others. It helps them to overcome the problem of ambiguity. Absence of WSD is one of the most challenging tasks in Machine Translation system and other WSD based applications [9]. Natural language dependent applications have been developed by understanding words, statements, phrases and etc. WSD has been applied for foreign languages to develop applications like machine translation, information retrieval, text

summarization, text classification, and question answering, speech processing, information extraction and text mining [1]. WSD was also developed for local languages like Amharic, Afaan Oromo by different researchers by using different approaches.

However, a limited number of natural language applications (i.e. tasks) have been developed for Tigrigna language such as, A Two Step Approach For Tigrigna Text Categorization by Gebrehiwot Assefa [15], Design and Development of Tigrigna Search Engine [16], Hidden Markov Model Based Tigrigna Speech Recognition by Hafte Abera [17], Phonetic Based Keyboard mapping For Tigrigna by Okbeab Tewelde [18], Part of Speech Tagging For Tigrigna Language by Teklay G. [19], Development of Stemming Algorithm for Tigrigna Text by Yonas Fisseha [20], etc. WSD prototype model for Tigrigna language has also been developed by Meresa Mebrahtu [21] using unsupervised machine learning methods. The researcher tested only unsupervised machine learning method that deals with grouping of contexts for the given word that express the same meaning without proving explicit sense labels for each group. Since Meresa M. [21] avoided human intervention by using unsupervised machine learning method, he could not achieve high performance. The reason behind the low performance of his system is mainly the use of no sense annotation.

Since, Tigrigna language has a number of ambiguous words; absence of automatic word sense disambiguation can be a challenge for the development of WSD based applications of Tigrigna language. Therefore, solving such kind of problem for the language can help for the development of WSD based applications such as information retrieval systems, machine translation, etc. Even though WSD is important for developing other natural language applications, almost no WSD systems have been conducted for Tigrigna language except the one which is developed by Meresa M. [21]. The researcher conducted his research by using unsupervised machine learning methods for WSD of Tigrigna language and achieved encouraging results. But, we have learnt that Getahun W. [1] achieved attractive results for WSD system of Amharic using semi-supervised machine learning method compared to results achieved by Solomon A. [22] and Solomon M. [23] who used unsupervised and supervised approaches respectively with the same datasets. But Getahun W. [1] used different datasets from both Solomon A. [22] and Solomon M. [23]. However, Getahun W. [1] tried to compare the three machine learning methods by using the same datasets and he found that semi-supervised machine learning method improves the performance of WSD. By considering

the results achieved for Amharic language using semi-supervised machine learning method, we tried to investigate WSD prototype model for Tigrigna language using semi-supervised approach.

The conducted research answers the following questions: -

- Could semi-supervised machine learning method improve the performance of Tigrigna WSD prototype model compared to supervised and unsupervised learning methods?
- What would be the most effective algorithm from the selected algorthms that improve the performance of Tigrigna WSD prototype model?
- What would be the optimal window size for Tigrigna WSD systems using semi-supervised machine learning method?
- Could bootstrapping algorithms perform better for Tigrigna WSD prototype model using semi-supervised machine learning method compared to the supervised machine learning method?

1.3 Objective of the study

General Objective

The general objective of the study is to investigate word sense disambiguation model for Tigrigna language using semi-supervised machine learning approach.

Specific Objectives

To meet the above general objective, the study must address the following specific objectives:

- Studying ambiguities in Tigrigna texts in order to understand WSD issues in the language.
- Reviewing WSD researches conducted for other languages in order to understand the problem area and different methods that can be used to solve the problem.
- Collecting Tigrigna sentences with ambiguous words to prepare WSD corpus.
- Preparing feature sets from both labeled and unlabeled instances.
- Training WSD model using the selected clustering algorithms.
- Labeling unlabeled datasets using the best performing clustering algorithm.
- Training WSD model using the selected classification algorithms by using different window size.

- Measuring the performance of the developed model by conducting an experiment on the collected data sets.
- Making conclusions and forwarding recommendations for WSD on Tigrigna texts.

1.4 Scope and Limitation of the study

Word Sense Disambiguation is a complex and recent research discipline that requires the effective analysis and processing of ambiguous words in its textual context. There are no publicly available linguistic resources for Tigrigna language. Due to unavailability of words context database for Tigrigna language, the study was limited to train and evaluate using 1250 collected sentences with only five ambiguous words. Ambiguous words having two, three, and four senses were used in this study.

As we have discussed in the introduction part, Ambiguity in natural language processing can exist at word or sentences levels. However, this study focuses on a word level ambiguity. We concentrate on word level ambiguity due to its necessity to other levels as well as the indispensability of words for natural language processing systems. There are different types of ambiguities in Tigrigna language which are explained in section 3.5 of this study. But from those different ambiguities, the researcher focused only on lexical ambiguity because it is difficult and time taken to get enough Tigrigna datasets which are reffered to the other ambiguities.
The researcher is also focused on: -

- Textual information only. This means the system accepts data in text form only not sound or voice form.
- Semantic level analysis only. This means the system does not perform any kind of grammar and spelling correction or any other ambiguities in the language.

1.5 Significance of the study

The results of the study are expected to produce experimental evidences that demonstrate the applicability of semi-supervised machine learning methods to word sense disambiguation of Tigrigna texts. The datasets that we prepared can serve as Tigrigna dataset repository. It can also serve as the initiation for the preparation of part of speech tagged dataset repository for the development and evaluation of Tigrigna WSD by applying part of speech tagging. It can also have a contribution to future researches and development of other Natural Language Processing

applications specifically machine translation, Information Retrieval, grammatical analysis, content and thematic analysis as those areas require word sense disambiguation as complement.

1.6 Methodology of the study

Research methodology is a systematic way to solve a problem. It may be understood as a science of studying how research is done scientifically. Essentially, it is also the procedures by which researchers go about their work of describing, explaining and predicting phenomena [24]. Therefore, the general methodology of the study is Experimental research. To conduct experimental research, the researcher uses tools, techniques, and procedures by using the following methods.

Literature Review

Various literatures that are considered to be relevant for the research work are reviewed to get better understanding of the area and to have detailed knowledge on various techniques that are essential for WSD systems. Different literatures are reviewed in the following areas: -

- Researches conducted on Word sense disambiguation for both local and other languages.
- Different machine learning approaches with their advantage and disadvantage.
- Different natural language applications which are done previously for Tigrigna language and other local languages.
- Tigrigna Writing system, punctuation marks and its syntactic structure.
- Different documents about Tigrigna have been reviewed.

Data Source and Corpus Preparation

To achieve the objective of this study, corpus preparation is the major task. A text corpus is one of the resources required in natural language processing researches. A good-sized text can reasonably show the performance of the model that you develop. For the purpose of this research, the researcher collects sample texts or sentences of different disciplines which are collected from different sources (mekalah Tigray, dmtsi weyane Tigray, demhit, etc). Those sample texts or sentences contain ambiguous words of Tigrigna language. Before using those collected data directly to our system, they have to be passed through preprocessing tasks. Since, finding large

number of manually annotated datasets for Tigrigna language is so difficult; the researcher uses semi-supervised machine learning approach for conducting this study because semi-supervised machine learning approach uses a small number of manually labeled or annotated data to automatically label the unlabeled examples. The reason that makes difficult for finding large number of manually annotated datasets for tigrigna language is due to the absence of context-based database repository. As shown in table 1.1, the following Tigrigna words have been selected for conducting this study: መደብ (medeb), ሓለፈ (Halefe), ዓረቐ (OareQe), ሰዓረ (seOare), ከፈለ (kefele).

Table 1.1: Senses of selected ambiguous words

Ambiguous words	Sense 1	Sense 2	Sense 3	Sense 4
ሓለፈ(Halefe)	Pass	Promote	Pass Away(Die)	Lead
መደብ(Medeb)	Plan	Traditional Bed	Assigning or grouping	-
ዓረቐ(OareQe)	Negotiation	poverty	-	-
ሰዓረ(seOare)	Win	Not fasting	-	-
ከፈለ(Kefele)	Pay	Divide (Share)	-	-

The researcher used an average of 96 sentences for each sense of the ambiguous words. To conduct this research a total of 1250 sentences were used.

Tools and Techniques

Different tools and techniques have been used to achieve the goal of this research. Weka 3.8 and python programming language are used as a tool to develop word sense disambiguation. When we used Weka 3.8 as a development, we apply different classification and clustering algorithms to develop best word sense disambiguation model, and python programming language is also used to do preprocessing tasks like tokenization, stop word removal, normalization, stemming, etc. We choose those tools for conducting this study because of their familiarity with the researcher, and availability.

1.7 Evaluation Techniques

Since we used semi-supervised machine learning method in this study; two evaluation techniques were applied. Those are: - "classes to clusters evaluation" mode and 10-fold cross validation which are available in Weka. "Classes to clusters evaluation" mode is used to know the class of each cluster using clustering algorithms; whereas 10-fold cross validation is used to evaluate the performance of classification algorithms.

1.8 Thesis Organization

This thesis is organized in to six chapters. Chapter 1 presents overview and background of the study. The second chapter reviews related works on Word Sense Disambiguation that are done previously, approaches. In this chapter, different machine learning with different algorithms were also discussed. Chapter 3 presents the nature and structure of Tigrigna language, Tigrigna writing system, morphological structure of Tigrigna language and ambiguities in the language. The fourth chapter is about research methodology used for developing the proposed system. It contains data collection mechanism, corpus preparation, architecture of the proposed system, preprocessing tasks, and techniques. Experimentation and finding of the study were discussed in Chapter 5. Finally, in Chapter 6, conclusion and future works were presented.

Chapter Two

2. Literature Review

In this chapter Word Sense Disambiguation (WSD) is reviewed and discussed in detail. The chapter covers brief background and history of WSD, application areas and discussion on major approaches that have been employed for WSD research with special focus on a machine learning approach or corpus-based approach which is used in this study. Moreover, machine learning algorithms that are tested to perform well for WSD research including semi-supervised algorithms which is going to be employed in this research and others are discussed. The discussion on different approaches and algorithms would help the understanding of the central problem in WSD research and also facilitates the comparison of existing approaches to the specific solutions that are employed in this study.

2.1. Overview of Word Sense Disambiguation

In all the major natural languages around the world, there are a lot of words which denote meanings in different contexts. For example, Tigrigna word 'ከፈለ' have two different senses such as "divide", "pay". Such words with multiple senses are called ambiguous words and the process of finding the exact sense of an ambiguous word for a particular context is called Word Sense Disambiguation. Humans resolve such ambiguity by understanding the context of each ambiguous word in a document because they have inborn capability to differentiate the multiple senses of an ambiguous word in a particular context. But for a machine, it will be difficult to determine the meaning of such ambiguous words. Machines run only according to the instructions. So, different rules are fed to the system to execute a particular task. For machines to determine the meaning of such ambiguous words depending on context needs word sense disambiguation. To determine the correct sense of ambiguous words by using word sense disambiguation there are different WSD algorithms. According to [2] WSD algorithms take as input a word in a context along with a fixed inventory of potential word senses and return the correct word sense for that use. Both the nature of the input and the inventory of senses depend on the task. If speech synthesis is our task, the inventory might be restricted to homographs with differing pronunciations such as bass and bow.

If our task is automatic indexing of medical articles, the sense tag inventory might be the set of MeSH (Medical Subject Headings) thesaurus entries.

WSD is usually performed on one or more texts (although in principle bags of words, i.e., collections of naturally occurring words, might be employed). If we disregard the punctuation, we can view a text *T* as a sequence of words (*w*1, *w*2, . . . , *wn*), and we can formally describe WSD as the task of assigning the appropriate sense(s) to all or some of the words in *T*, that is, to identify a mapping *A* from words to senses, such that $A(i) \subseteq Senses\ D(wi)$, where *SensesD*(*wi*) is the set of senses encoded in a dictionary *D* for word *wi*,1 and *A*(*i*) is that subset of the senses of *wi* which are appropriate in the context *T*. The mapping *A* can assign more than one sense to each word $wi \in T$, although typically only the most appropriate sense is selected, that is, $|A(i)| = 1$ [14].

According to [14] WSD can be viewed as a classification task: i.e. word senses are the classes, and an automatic classification method is used to assign each occurrence of a word to one or more classes based on the evidence from the context and from external knowledge sources. Other classification tasks are studied in the area of natural language processing such as part-of speech tagging (i.e., the assignment of parts of speech to target words in context), named entity resolution (the classification of target textual items into predefined categories), text categorization (i.e., the assignment of predefined labels to target texts), etc. But an important difference between these tasks and WSD is that the former uses a single predefined set of classes (parts of speech, categories, etc.), whereas in the latter the set of classes typically changes depending on the word to be classified. In this respect, WSD actually comprises *n* distinct classification tasks, where *n* is the size of the lexicon.

According to [25] ambiguities can be divided into two main categories: Ambiguous words and ambiguous sentences. Each main category has different types of ambiguities. Ambiguous words can be further classified as homonymy and polysemy.

Ambiguous words

1. **Homonymy:** A case of homonymy is one of an ambiguous word, where different cases are related to each other in any way. Homonymy can be further divided into three types.
    a) **Homographs:** words that have the same spelling but differ in sound and meaning are called homographs.

b) **Homophones:** Words that are identical in sound but differ in spelling and meaning are called homophones.

2. **Full homonyms:** Words that are identical in sound and spelling are called full homonyms.
3. **Polysemy:** A case of polysemy is one where a word has several clearly related senses, e.g. mouth (of a river vs. of an animal), the two senses are clearly related by the concept of an opening from the interior of some solid mass to the outside, and of a place of issue at the end of some long narrow channel.

Ambiguous sentences

A sentence is ambiguous if it has two (or more) paraphrases which are not themselves' paraphrases of each other [25]. For example, the sentence "Visiting relatives can be boring" can be paraphrased into two ways:

- ✓ It can be boring to visit relatives.
- ✓ Relatives who are visiting can be boring.

According to Ide and Veroni [9] WSD involves two steps. The first step is to determine all the different senses for every word relevant to the text or discourse under consideration, i.e., to choose a sense inventory from the lists of senses in everyday dictionary, from the synonyms in a thesaurus, or from the translations in a translation dictionary. The second step involves a means to assign the appropriate sense to each occurrence of a word in context. All disambiguation work involves matching the context of an instance of the word to be disambiguated either by using information from external knowledge sources or with contexts of previously disambiguated instances of the word. For both of these sources, we need preprocessing or knowledge-extraction procedures representing the information as context features. However, it is useful to recognize that another step is also involved here: the computer needs to learn how to associate a word sense with a word in context using either machine learning or manual creation of rules or metrics. Unless the associations between word senses and context features are given explicitly in the form of rules by a human being, the computer will need to use machine learning techniques to infer the associations from some training material.

It is useful to distinguish two variants of the generic WSD task [2]. The first one is the **lexical sample** task, a small pre-selected set of target words is chosen, along with an inventory of senses for each word from some lexicon. Since the set of words and the set of senses are small, **supervised**

machine learning approaches are often used to handle lexical sample tasks. For each word, a number of corpus instances (context sentences) can be selected and hand-labeled with the correct sense of the target word in each. Classifier systems can then be trained using these labeled examples. Unlabeled target words in context can then be labeled using such a trained classifier.

The second one is **all-words** task; systems are given entire texts and a lexicon with an inventory of senses for each entry and are required to disambiguate every content word in the text. The all-words task is very similar to part-of-speech tagging, except with a much larger set of tags, since each lemma has its own set. A consequence of this larger set of tags is a serious data sparseness problem; there is unlikely to be adequate training data for every word in the test set. Moreover, given the number of polysemous words in reasonably-sized lexicons, approaches based on training one classifier per term are unlikely to be practical.

WSD is a long-standing problem in Computational Linguistics and has significant impact in many real-world applications including machine translation, information extraction, and information retrieval [26].

2.2. History of word sense disambiguation

In computational linguistics, **word-sense disambiguation** (WSD) is an open problem of natural language processing, which governs the process of identifying which sense of a word (i.e. meaning) is used in a sentence, when the word has multiple meanings (polysemy). The task of WSD is a historical one in the field of natural language processing. In fact, it was conceived as a fundamental task of Machine Translation (MT) already in the late 1940s. Research work [10] in WSD was started during the late 1940s. At that time, researchers had already in mind essential ingredients of WSD, such as the context in which a target word occurs, statistical information about words and senses, knowledge resources, etc. Very soon it became clear that WSD was a very difficult problem, also given the limited means available for computation. Indeed, its acknowledged hardness [14] was one of the main obstacles to the development of MT in the 1960s. In the 1970s, WSD was a subtask of semantic interpretation systems developed within the field of artificial intelligence, but since WSD systems were largely rule-based and hand-coded they were prone to a knowledge acquisition bottleneck. By the 1980s large-scale lexical resources, such as the Oxford Advanced Learner's Dictionary of Current English (OALD), became available: hand-

coding was replaced with knowledge automatically extracted from these resources, but disambiguation was still knowledge-based or dictionary-based. In the 1990s, the statistical revolution swept through computational linguistics, and WSD became a paradigm problem on which to apply supervised machine learning techniques. The 2000s saw supervised techniques reach a plateau in accuracy, and so attention has shifted to coarser-grained senses, domain adaptation, semi-supervised and unsupervised corpus-based systems, combinations of different methods, and the return of knowledge-based systems via graph-based methods. Still, supervised systems continue to perform best.

As [10] indicated, in 1949 Zipf proposed his "Law of Meaning" theory. This theory states that there exists a power-law relationship between the more frequent words and the less frequent words. The more frequent words have more senses than the less frequent words. The relationship has been confirmed later for the British National Corpus. As the data sets, corpora, online Dictionaries vary language to language all over the world, there was not any bench mark of performance measurement in this domain in the early age.

2.3. Approaches of WSD

Most disambiguation approaches tend to focus on the identification of word-specific contextual indicators that can be used to distinguish between a word's senses. Efforts to acquire these clues or indicators have been characterized by their need for intensive human involvement for each word, which creates the associated problem of limited vocabulary coverage. This is termed as the knowledge acquisition bottleneck in WSD literature. WSD systems can thus be classified based on how they attempt to deal with the knowledge acquisition bottleneck, by considering how they acquire disambiguation information [36]. Different WSD approaches have been used through the evolution of WSD research. As [11] indicated currently, most current WSD approaches can be characterized as knowledge-based approach, corpus-based approach and Hybrid approach. Knowledge based uses external sources as otology information. Corpus-based is machine learning approach. Machine learning approach is farther divided into supervised, semi-supervised and unsupervised learning methods. The hybrid approach is a combination of both knowledge based and corpus-based approach. From those approaches our main focus is on developing WSD by using corpus-based approach. Therefore, our literature review focused on corpus-based approaches.

2.3.1. Corpus-based Approaches

Corpus-based methods provide an alternative strategy for overcoming the lexical acquisition bottleneck, by obtaining information necessary for WSD directly from textual data. A corpus provides a bank of samples which enable the development of numerical language models, and thus the use of corpora goes hand-in-hand with empirical methods [9]. A major limitation of knowledge-based WSD systems is their reliance on pre-coded knowledge sources, which affects their inability to handle large vocabulary in a wide variety of contexts due to the associated expense of manual acquisition of lexical and disambiguation information. In an effort to overcome this problem, fueled by the increased availability of natural language data in electronic form, WSD researchers have recently turned to corpora to help extend the coverage of existing systems as well as bootstrap or train new systems [27].

Corpus-based methods use machine-learning techniques to induce models of word usages from large collections of text examples. Corpus-based methods uses semantically annotated corpora (most commonly wordnet) to train machine learning algorithms. According to [1] Corpus-based is farther divided into supervised, semi-supervised and unsupervised learning methods.

2.3.1.1. Supervised Machine Learning Method

In the last 15 years, the NLP community has witnessed a significant shift from the use of manually crafted systems to the employment of automated classification methods. Such a dramatic increase of interest toward machine learning techniques is reflected by the number of supervised approaches applied to the problem of WSD [14]. Supervised machine learning is an approach that relies on sense tagged corpora for training the model. They yield very high accuracy in the domain of the training corpus. But this accuracy comes at the cost of sense tagged corpora which is a costly resource in terms of the time and the manual efforts involved creating such corpora for all languages in all domains will be impracticable. Hence, those approaches cannot be easily ported to different languages or domains [28]. It uses machine learning techniques for inducing a classifier from manually sense-annotated data sets. Usually the classifier (often called word expert) is concerned with a single word and performs a classification task in order to assign the appropriate sense to each instance of that word. The training set is used to learn the classifier typically contains a set of examples in which a given target word is manually tagged with a sense from the sense

inventory of a reference dictionary. Generally supervised approaches have obtained better results than unsupervised approaches [29].

Algorithms that can learn to predict discrete valued labels are called classification algorithms or classifiers, whereas the algorithms that can learn to predict continuous valued labels are called regression algorithms. As the task of WSD only involves discrete valued labels for word senses, we use only classification algorithms [22]. To achieve good classification, supervised approaches have their own limitations; because supervised approaches require large sense-tagged training set. Despite the availability of large corpora in some language, manually sense-tagging of a corpus is very difficult; because manually sense-tagging of a corpus requires external teachers which is laborious (expensive), subjective and time taking [1].

In supervised approaches, a sense disambiguation system is learned from a representative set of labeled instances drawn from sense annotated corpus. Input instances to these approaches are features encoded along with their appropriate labels. The output of the system is a classifier system capable of assigning labels to new feature encoded inputs. The following are the most commonly used algorithms in supervised machine learning:

Decision trees: Decision trees are trees that classify instances by sorting them based on feature values. Each node in a decision tree represents a feature in an instance to be classified, and each branch represents a value that the node can assume. Instances are classified starting at the root node and sorted based on their feature values. The feature that best divides the training data would be the root node of the tree [30].

Let us take the ambiguous word “kefele” in Tigrigna language to make it clear about decision tree algorithm. This Tigrigna word has two different meanings which are “divide” and “pay”. Context words that make the ambiguous words to have the meaning of “pay “are denoted by negative (+) and negative (-) for “divide” contexts. Context words like “meswaIti” and “genezeb” expresses about “pay” sense whereas context words like “maOre” and “OCa” expresses about “divide” sense. The idea is demonstrated in Figure 2.1:

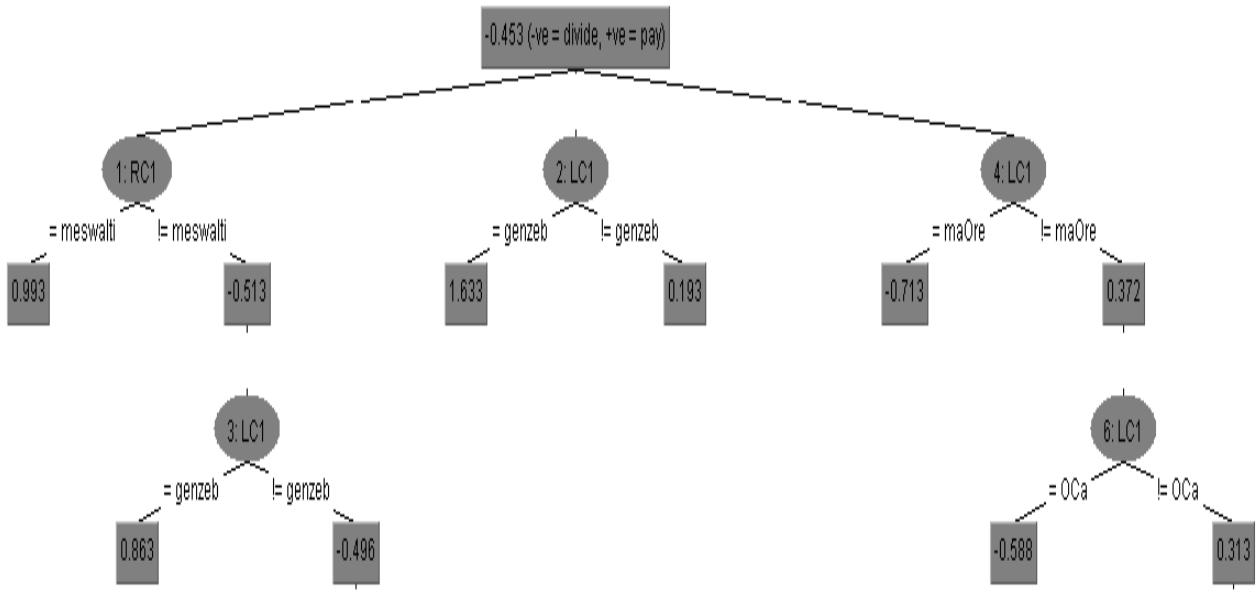


Figure 2.1 : An example of Decision Tree structure

The decision tree algorithm usually finds entropy of attributes and converted into information gain as heuristic for selecting the best attribute that will classify the training data into their corresponding classes. The attribute with the highest information gain will help to start training data classification. The information gain is the difference between the entropy of the label before and after the split [31].

Even though decision tree has the advantages such as reasonable training time, fast application, easy to interpret in that it can be expressed as if- else statement, easy to implement, can handle large number of features, it cannot handle complicated relationship between features and, problems with lots of missing data and overfitting. Therefore, to solve this kind of problem we can use two methods. One is limit the number of iterations of the decision tree which leads to a tree with a bounded number of nodes and the other is pruning the tree after it is built which means removing attributes with low information gain [31].

Naïve Bayes: A **Naive Bayes classifier** is a simple probabilistic classifier based on the application of Bayes' theorem. It is assumed that the feature variables representing a problem are conditionally independent, given the classes [32]. It relies on the calculation of the conditional probability of each sense S_i of a word w given the features f_j in the context. The sense $\hat{S}$ which maximizes the following formula is chosen as the most appropriate sense in context:

$$\hat{S} = \underset{S_i \in Sense_D(w)}{\operatorname{argmax}} P(S_i \mid f_1, \ldots, f_m) = \underset{S_i \in Senses_D(w)}{\operatorname{argmax}} \frac{P(f_1, \ldots, f_m | S_i) P(S_i)}{P(f_1, \ldots, f_m)}$$

$$= \underset{S_i \in Senses_D(w)}{argmax} P(S_i) \prod_{j=1}^{m} P(f_j | S_i) \qquad (2.1)$$

Here, the numbers of features are represented by m. The probability P (Si) is calculated from the Co-occurrence frequency in training set of sense and P (f_j | Si) is calculated from the feature in the presence of the sense [10].

Bayesian classifiers have also exhibited high accuracy and speed when applied to large databases. Naive Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is known as class conditional independence. However, The Naive Bayes assumption has two consequences. The first is that all the structure and linear ordering of words within the context is ignored. This is often referred to as a bug of words model. The other is that the presence of one word in the bag is independent of another [14].

Support Vector Machine: Support Vector Machine based algorithms use the theory of Structural Risk Minimization. The goal of SVM is to separate positive examples from negative examples with maximum margin. Margin is the distance of hyperplane to the nearest of the positive and negative examples. The positive and negative examples which are closest to the hyperplane are called support vector. The SVM based algorithms are used to classify few examples into two distinct classes. This algorithm finds a hyperplane in between these two classes, so that, the separation margin between these two classes becomes maximum [10]. A test example is classified depending on the side of the hyper plane it relies in. Input features can be mapped into high dimensional space before performing the optimization and classification. A kernel function can be used to reduce the computational cost of training and testing in high dimensional space. If the training examples are non-separable, a regularization parameter C (=1 by default) can be used to control the trade-off between achieving a large margin and a low training error [33].

One such algorithm that effectively works around the time-consuming step of numerical quadratic programming is the Sequential Minimal Optimization (SMO) algorithm. SMO is one of the SVM

fast iterative algorithms which is also easy to implement and has attractive scaling property when it is compared with other support vector machines [34].

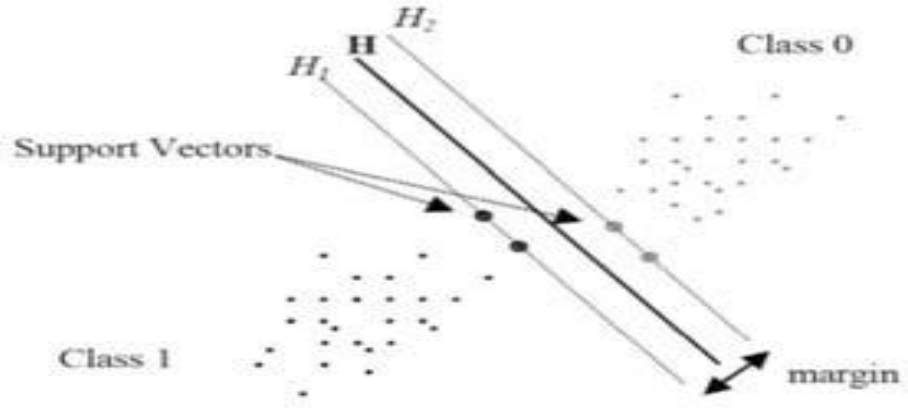


Figure 2.2: Support Vector Machine

Ensemble Methods

Sometimes different classifiers are available which we want to combine for improving the overall disambiguation accuracy. Combination strategies called ensemble methods [21] typically put together learning algorithms of different nature, that is, with significantly different characteristics. In other words, features should be chosen so as to yield significantly different, possibly independent, views of the training data (e.g., lexical, grammatical, semantic features, etc.). Ensemble methods are becoming more and more popular as they allow one to overcome the weaknesses of single supervised approaches. Several systems participating in recent evaluation campaigns employed these methods [14].

The classifiers can be combined by using different strategies [10]. Some of the strategies that are used to combine different classifiers are: -

Majority voting: - In this strategy, one vote is given to a particular sense of the word. Given a target word *w*, each ensemble component can give one vote for a sense of *w*. The sense *S* which has the majority of votes is selected:

$$\hat{S} = \text{argmax}_{S_i \in Senses_D(w)} \mid \{j : vote(C_j) = S_i\} \mid, \quad (2.3)$$

Where, *vote* is a function that, given a classifier, outputs the sense chosen by that classifier. In case of tie, a random choice can be made among the senses with a majority vote. Alternatively, the ensemble does not output any sense.

Probability Mixture: - In this strategy, first the confidence score for the meaning of a target word is evaluated by the first order classifiers and then normalization is applied. As a result, the probability distribution on the senses of the word is obtained. Next, these probabilities are added, and the sense for which the score is highest, considered as the desired sense.

Rank-Based Combination: - First order classifier gives the rank to the senses for a given input target word and this method selects the sense "s" of word by finding the maximum value among the summations of its ranks in the classifiers $C_1, \ldots, C_m$. Equation is given as below

$$\hat{S} = argmax_{S_i \in Senses_D(w)} \sum_{j=1}^{m} -Rank_{C_j}(S_i) \quad (2.4)$$

Where, C_j represents a classifier and the rank of the sense S_i is represented by Rank $_{Cj}(S_i)$.

2.3.1.2. Unsupervised Machine learning

The approach which doesn't use any sense tagged corpus is unsupervised. These are mainly clustering approaches. The neighboring words are clustered to one or more groups depending on some similarity strategy and each cluster corresponds to a sense. This approach requires no supervision. As a child is never taught to smile or laugh, they learn automatically is a perfect example of an unsupervised approach. Unsupervised methods have the potential to overcome the knowledge acquisition bottleneck, that is, the lack of large-scale resources manually annotated with word senses. They are able to induce word senses from input text by clustering word occurrences, and then classifying new occurrences into the induced clusters [6]. They do not rely on labeled training text and, in their purest version, do not make use of any machine–readable resources like dictionaries, thesauri, ontologies, etc. However, the main disadvantage of fully unsupervised systems is that, as they do not exploit any dictionary, they cannot rely on a shared reference inventory of senses [14]. It is not suitable for a larger scale situation, incorrect assignment of instances in training data, formation of heterogeneous clusters and the difference between the number of clusters formed and the number of senses of the target word [32].

The quality of a clustering result depends on the similarity measure used by the method. All measures refer to the feature values in some way, but they consider different properties of the feature vector. If the features are represented by a distributional feature vector; a range of measures calculate either the distance or the similarity between two objects. The smaller the distance between objects, the more similar they are to each other [1]. A range of measures can be used for calculating the similarity of distributional objects. One is Minkowski Metric or L_q norm which calculates the distance *d* between the two objects *x* and *y* by comparing the values of their *n* features as shown equation (2.5). The Minkowski metric can be applied to frequency, probability and binary values.

$$d(X,Y) = \sqrt[q]{\sum_{i=1}^{n} (|x_i - y_i|)^q} \tag{2.5}$$

where X = (x_1, x_2, …, x_n) and Y = (y_1, y_2, …, y_n) are two n-dimensional data objects; n is size of vector attributes of the data object; q= 1,2,3, …Two important special cases of the Minkowski metric are q=1 and q=2.

If q = 1, d is Manhattan distance or City block distance or L_1 norm as shown in the following equation (2.6).

$$d(X,Y) = \sum_{i=1}^{n} (|x_i - y_i|) \tag{2.6}$$

If q=2, the distance is Euclidean distance or L_2 norm as shown in the following equation (2.7).

$$d(X,Y) = \sqrt{\sum_{i=1}^{n} (|x_i - y_i|)^2} \tag{2.7}$$

If X and Y are two vector attributes of data objects, then cosine similarity measure can be used as shown æ the following equation (2.8).

$$d(X,Y) = \frac{|x_i \bullet y_i|}{\|x_i\| * |y_i\|} \tag{2.8}$$

2.3.1.2.1. Learning Algorithms under Unsupervised WSD

Clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to those in other clusters. Clustering is a main task of explorative data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics [35]. Mainly, clustering algorithms are generally categorized as hierarchical, partitional and Density Based algorithms [36].

Hierarchical clustering algorithms: A hierarchical clustering method works by grouping data objects into a tree of clusters unlike partitioning algorithm. Depending on whether the clustering is performed top-down, i.e. from a single cluster to the maximum number of clusters, or bottom-up, i.e. from the maximum number of clusters to a single cluster, we distinguish divisive and agglomerative clustering [37]. The result of a hierarchical clustering algorithm can be graphically displayed as tree, called a dendrogram. This tree graphically displays the merging process and the intermediate clusters. This graphical structure shows how points can be merged into a single cluster. Hierarchical methods suffer from the fact that once we have performed either merge or split step, it can never be undone. This inflexibility is useful in that it leads to smaller computation costs by not having to worry about a combinatorial number of different choices [38].

Agglomerative Hierarchical clustering: This bottom-up strategy starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters, until all of the objects are in a single cluster or until certain termination conditions are satisfied. Most hierarchical clustering methods belong to this category and they differ only in their definition of inter-cluster similarity. The agglomerative one has also three categories. These are Single link, complete link, and average link [38].

Single Link clustering algorithm: In single-link clustering the similarity between two clusters is the similarity between their most similar members that are from the two different clusters and select the pair with the greatest similarity. We search over all pairs of objects that are from the two different clusters and select the pair with shortest distance between them [38]. Single-link clustering has clusters with the good local coherence since similarity function is locally defined. However, single-link is not practical because it suffers from the chaining effect. Single-link

clustering is closely related to the minimum spanning tree of a set of points. A single-link clustering can be constructed top-down from a minimum spanning tree by removing the longest edge in the minimum spanning tree so that two unconnected components are created, corresponding to two sub clusters. The same operation is then recursively applied to these two sub clusters [39]. The following Figure 2.5 illustrates the way cluster similarity can be computed using single-link algorithm.

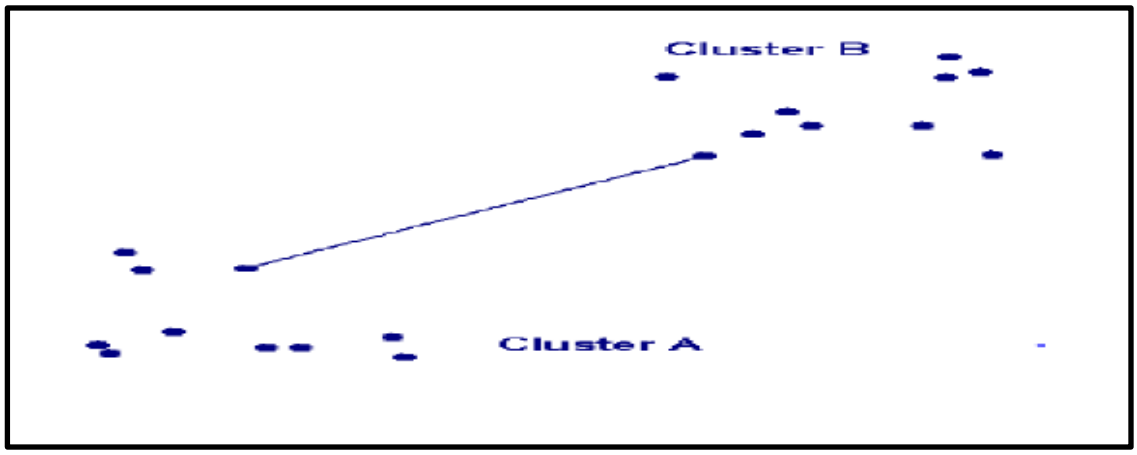


Figure 2.3: Single link clustering

Complete-link clustering algorithm: In complete-link clustering, the similarity between two clusters is the similarity between their least similar members (e.g. using the Euclidean distance) [40]. It has a similarity function that focuses on global cluster quality as opposed to locally coherent clusters as in that if single-link clustering [39]. In other words, the distance between two clusters is given by the value of the longest link between the clusters. The similarity of two clusters is the similarity of their two most dissimilar members [41]. Complete-link clustering avoids elongated clusters which means it does not suffer from the chaining effect. Rather than producing straggly elongated clusters like single-link, complete-link generates compact clusters [40]. The following Figure 2.6 illustrates the way cluster similarity can be computed using complete-link algorithm.

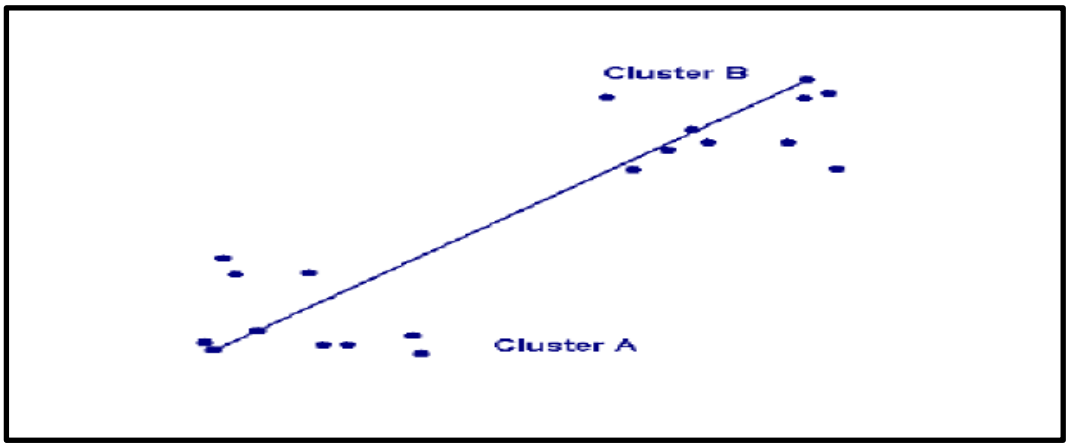


Figure 2.4: Complete-link clustering

Average-link clustering algorithm: Average-link clustering produces similar clusters to complete-link clustering except that it is less susceptible to outliers. It computes the similarity between two clusters as the average similarity between all pairs of elements across clusters. The distance between two clusters is defined as the average of distances between all pairs of objects, where each pair is made up of one object from each group [41]. Average-linkage is sensitive to the shape and size of clusters. Thus, it can easily fail when clusters have complicated forms departing from the hyper spherical shape [38].

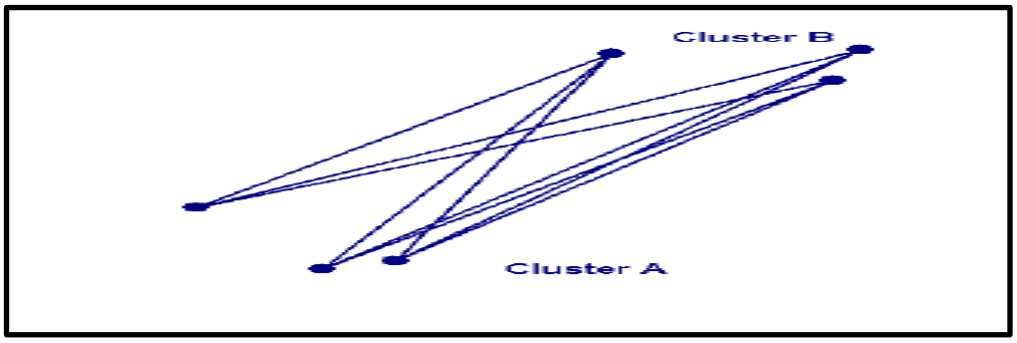
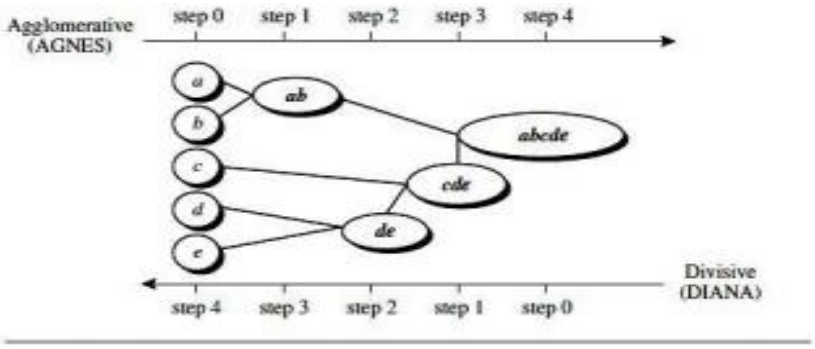


Figure 2.5: Average-link clustering

Divisive Hierarchical clustering: This is top-down strategy and does the reverse of agglomerative hierarchical clustering by starting with all objects in one cluster. It subdivides the cluster into smaller and smaller pieces, until each object forms a cluster on its own or until it satisfies certain termination conditions, such as a desired number of clusters is obtained or the diameter of each cluster is within a certain threshold [36]. Figure 2.8 makes clear the difference between two of the hierarchical clustering algorithms which means Agglomerative and Divisive Hierarchical clustering's.



Agglomerative and Divisive Hierarchical clustering's on data objects {a, b, c, d, e}.

Figure 2.6: Example of a dendrogram for hierarchical clustering (adopted from *[36]*)

Partitional Algorithms: Partitional algorithms do not produce a nested series of partitions. Instead, they generate a single partitioning, often of predefined size *K*, by optimizing some criterion. A combinatorial search of all possible clustering's to find the optimal solution is clearly intractable. The algorithms are then typically run multiple times with different starting points. Partitional algorithms are not as versatile as hierarchical algorithms but they often offer more efficient running time. The partitioning algorithms can include: K-means, Expectation maximization, Farthest First algorithms.

K-means clustering algorithm: K-means is a partitioning clustering algorithm which is used to classify the given data objects into k different clusters through the iterative method, which tends to converge to a local minimum. So, the outcomes of generated clusters are dense and independent of each other [42]. It is widely used due to easy implementation and efficient execution. It is a simpler clustering algorithm as it divides the dataset into fixed no. of clusters [36]. The algorithm consists of two separate phases. In the first phase user selects k centers randomly, where the value k is fixed æ advance. To take each data object to the nearest center. Several distance functions are considered to determine the distance between each data object and the cluster centers. When all the data objects are included in some clusters, the first step is completed and an early grouping is done. Then the second phase is to recalculate the average of the early formed clusters. This iterative process continues repeatedly until the criterion function becomes the minimum [43].

Expectation Maximization Clustering Algorithm: Expectation Maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates

of parameters in statistical models, where the model depends on unobserved latent variables [37]. The EM iteration alternates between performing an expectation (E) step, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a maximization (M) step, which computes parameters maximizing the expected log-likelihood found on the *E* step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step. EM assigns a probability distribution to each instance which indicates the probability of it belonging to each of the clusters [44].

Farthest First Clustering Algorithm: Farthest first is a modified of K-Means that places each cluster center in turn at the point further most from the existing cluster center. This point must lie within the data area. This greatly increases the clustering speed in most of the cases since less reassignment and modification is needed. Farthest-point heuristic-based method is fast and suitable for large-scale data mining applications [35].

2.3.1.3. Semi-supervised machine learning

Supervised machine learning approach of WSD performs better by human intervention, but it has limitations of knowledge-acquisition bottleneck i.e. it requires manually labeled sense examples which takes much time, very laborious and therefore very expensive to create when the corpus size increases. Whereas Unsupervised machine learning method deals with grouping of contexts for the given word that express the same meaning without proving explicit sense labels for each group. Even though it avoids human intervention, is inexpensive and extremely scalable, it has worse performance than that of supervised one because it relies on less knowledge of sense annotation. Since the supervised and unsupervised machine leaning methods have their own drawbacks, the semi-supervised method narrows the gap of those methods by making use of labeled and unlabeled training data. In other tokens, it minimizes knowledge-acquisition bottleneck of supervised learning, and it improves poor performance of unsupervised learning. It is an interesting method that considers how to avoid as many limitations as possible without losing their major advantages. One possible advantage is to achieve better classification result with less effort [1].

Considering both the availability of a large amount of unlabeled data and direct use of word; semi-supervised learning has recently become an active research area. It requires only a small amount of labeled training data and is sometimes able to improve performance using unlabeled data. Semi-

supervised learning is a class of machine learning techniques that use both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. The name "semi-supervised learning" comes from the fact that the data used is between supervised learning (with completely labeled training data) and unsupervised learning (without any labeled training data). Many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent and the process is expensive and time consuming whereas acquisition of unlabeled data is relatively inexpensive and less time consuming [45].

The basic idea of semi-supervised learning is to automatically label the unlabeled examples using a small number of human labeled examples as seeds. By doing this, semi-supervised learning yields a large labeled dataset that can be used as training data for a normal supervised learning algorithm. While the labeling of unlabeled data is indeed another classification problem, this classification can exploit the fact that all examples needed to be classified (the unlabeled data) are available at the time of training [7].

Semi-supervised learning [45] attempts to make use of this combined information to surpass the classification performance that could be obtained either by discarding the unlabeled data and doing supervised learning or by discarding the labels and doing unsupervised learning. Semi-supervised learning promises higher accuracies with less annotating effort. It is therefore of great theoretic and practical interest. In this method, given a set of m independent and distributed examples (x_1, x_2, x_3……. x_m) ∈ X with corresponding labels (y_1, y_2, y_3,……y_m) ∈ Y. In addition to this we also having n unlabeled examples (x_{m+1}, x_{m+2} ,….. x_{m+n})∈X. Semi-supervised learning gives higher accuracies with less effort on annotating data.

Generally, semi-supervised learning is used to automatically label the unlabeled examples using a small number of manually labeled examples as seeds. In this paper, we incorporate unlabeled data by combining some seeds instances directly into the data representation (features), so that unsupervised algorithms directly applied for clustering based on instances similarity; and Supervised algorithms applied for classification after the unlabeled data are given their labels using the cluster labels found during clustering. Sometimes unsupervised learning may suggest change

of seed instances labeled automatically during clustering. However, manually annotated labels should not be altered. This is because once they were annotated as seed examples intentionally.
In using semi-supervised machine learning algorithms, unlabeled data can be used in different ways depending on the assumptions employed in WSD model. Assumptions that can be employed in WSD model are clustering and manifold assumptions [1].

Semi-supervised classification (Manifold assumption): - it involves both labeled and unlabeled datasets with in the training datasets. During this assumption, training datasets are building by exploiting unlabeled datasets by using the already labeled seed examples. In other words, this assumption includes supervised learning and additional unlabeled datasets [46]. The goal of this assumption is classifying or labeling the unlabeled datasets only by pooling more unlabeled datasets during training. According to this assumption, data with similar inputs are supposed to have similar categories/class [47]. It supposes that samples are distributed on a low dimensional manifold in the data space [48]. Graph based algorithms which are grouped under this assumption are label propagation, LapSVM and transductive learning [1].

Semi-supervised clustering (clustering assumption): -clustering assumption includes unsupervised learning and additional labeled datasets [1]. It assumes that the dataset contains clusters that have homogenous labels and the unlabeled datasets are used to identify those clusters [49]. In this assumption, the training datasets contains some class labeled datasets along with large number of unlabeled datasets to get better clustering performance. the clustered result serves for classification task [1]. It also assumes that similar instances should share the same class label in semi-supervised learning methods. It can also be interpreted as the requirement that the decision boundary has to lie in low density regions [1] [49]. This interpretation has been widely used in learning since it can be used in the design of standard algorithms such as Boosting and SVM [49]. Clustering assumption is also named as clustering with side information or constrained clustering. Side-information in the form of pair-wise constraints, which means making a pair of objects belong to the same cluster or different clusters. For clustering purpose with some labeled data, there are must-link and cannot-link constraints. The clustering algorithm participating in must-link constraint must try to assign the same label to the pair of points talking similarly and the clustering algorithm participating in a cannot-link constraint assigns different class labels to a pair of points

that are talking different. Inferring the Pair wise constraints automatically from the structure of the data can be also possible without a user modification on them [46].

Co-training, self-training (ADtree AdaboostM1, bagging, semi-boost) and Semi-supervised SVM such as SMO algorithms are incorporated in the cluster assumption [1]. Implementing those two assumptions have common interesting thing which is the unlabeled data provide some helpful information about data distribution in the training data. From two of those assumptions we used clustering assumption for this study. Therefore, algorithms that are included in this assumption and used in this study are discussed below except SVM algorithm because we have already discussed in the above when we discuss about supervised learning methods.

2.3.1.3.1. Bootstrapping

Semi-supervised methods for WSD are characterized in terms of exploiting unlabeled data in learning procedure with the requirement of predefined sense inventory for target words. There are two main approaches of semi-supervised learning methods in WSD: co-training and self-training [50]. Bootstrapping algorithms are commonly used semi-supervised learning methods for WSD. Bootstrapping sense tagged seed examples are used to overcome the bottleneck of acquisition of large sense-tagged data. It works by iteratively classifying unlabeled examples and adding confidently classified examples into labeled dataset using a model learned from augmented labeled dataset in previous iteration. It can be found that the affinity information among unlabeled examples is not fully explored in this bootstrapping process. Bootstrapping is based on a local consistency assumption: examples close to labeled examples within same class will have same labels, which is also the assumption underlying many supervised learning algorithms, such as kNN [45].

The aim of bootstrapping is to build a sense classifier with a little training data, and thus overcome the main problems of supervision: the lack of annotated data and the data sparsity problem. Bootstrapping usually starts from few annotated data *A*, a large corpus of un-annotated data *U*, and a set of one or more basic classifiers. As a result of iterative applications of a bootstrapping algorithm, the annotated corpus *A* grows increasingly and the untagged data set *U* shrinks until some threshold is reached for the remaining examples in *U* [14].

Self-training is a commonly used method for semi-supervised learning. In this method a classifier is first trained with the sample of labeled data. Then the classifier is used to classify the unlabeled datasets. Normally the most assured unlabeled data, along with their predicted labels, are appended to the training set. The classifier is re-trained with the new data and the process is repeated. This process of re-training the classifier by itself is called self-teaching or bootstrapping. The self-training mechanism used is a five-step process. (1) The detector is trained by using a limited set of completely labeled positive samples and a complete set of negative samples. the newly labeled data is selected using the selection metric. (5) The above steps are repeated until all the data to be trained are added. The method is implemented as a wrapper method over the training process of an existing object detector and the experimental results are provided [50]. Bootstrapping is about training a classifier on dataset by picking a word that co-occurs with the target word in particular sense. The following algorithm shown in Algorithm 1 shows how bootstrapping algorithm works.

Bootstrap Algorithm

Input: $Z = \{z1, z2, \ldots, zN\}$, with $zi = (xi, yi)$ as training set.

B, number of sampled versions of the training set.

Output: S(Z), statistical estimate and its accuracy.

for n=1 to B

a) Draw, with replacement, $L \leq N$ samples from the training set Z, obtaining the nth sample Z^{*n}.

b) For each sample Z^{*n}, estimate a statistic $S(Z^{*n})$.

2. Produce the bootstrap estimate S(Z), using $S(Z^{*n})$ with $n = \{1, \ldots, B\}$.

3. Compute the accuracy of the estimate, using the variance or some other criterion

Algorithm 1: Bootstrapping algorithm (adopted from [1])

To depict the bootstrap algorithm with graphical idea the algorithm starts with the training set Z, obtaining several bootstrap samples (Z*n). For each bootstrap samples, statistical measure can be computed S (Z*n). The following Figure 2.9 shows A graphical idea of bootstrapping approach.

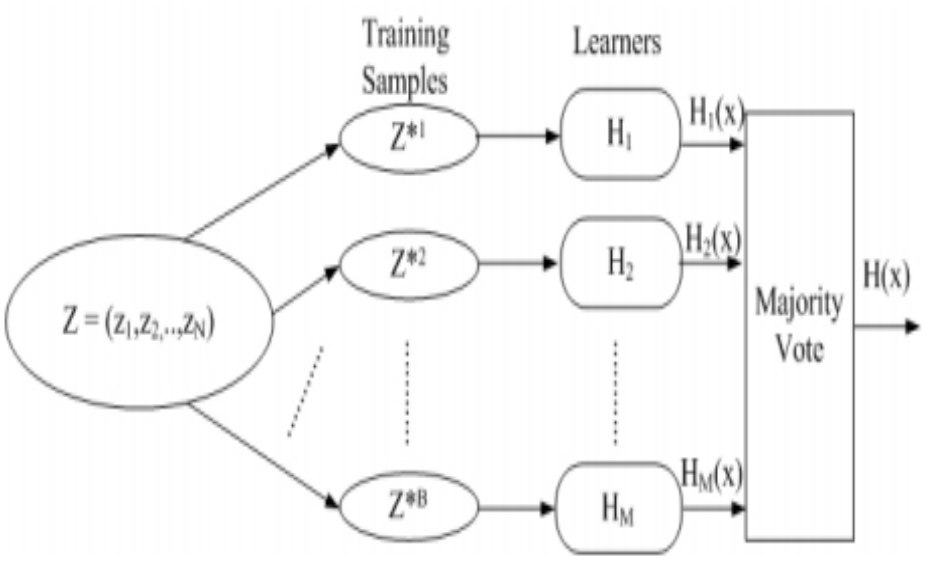


Figure 2.7: Bootstrapping procedure for classification (adopted from *[51]*)

Some of the bootstrapping algorithms are discussed as follows: -

Bagging: it stands for bootstrap aggregation. Bagging is a technique which uses bootstrap sampling to reduce the variance and improve the accuracy of some predictor. It may be used in regression and classification. Consider a size of N Dataset $Z = \{z_1, z_2, ..., z_N\}$, where now $z_i = (x_i, y_i)$, where y_i is a class label, in classification problems, or a real number, in regression problems. The rationale of bagging is to learn a set of B predictors (each from a bootstrap sample $Z^{*}_{b} \subseteq Z$, for b = 1,...,B) and then produce a final predictor by combining this set of predictors. This algorithm works by majority voting [51]. The combination of multiple predictors decreases the expected error because it reduces the variance component of the bias-variance decomposition. The reduction on this variance component is proportional to the number of classifiers applied in the ensemble. The use of the bagging technique improves the classification results whenever the base classifiers are unstable, this being the main reasons why the bagging approach works well for classification. The final classification decision is produced by a majority vote on the weak learners output [52].

AdaBoost: it stands for Adaptive Boosting. AdaBoost is a boosting algorithm which is mostly applied when boosting the accuracy of a learning method is considered as important. Boosting is a general method for improving the accuracy of any given learning algorithm. The main idea of boosting algorithms is to combine many simple and moderately accurate hypotheses (called weak

classifiers) into a single, highly accurate classifier for the task at hand. The weak classifiers are trained sequentially and, conceptually, each of them is trained on the examples which were most difficult to classify by the preceding weak classifiers [53]. The AdaBoost algorithm is now a well-known and deeply studied boosting method to build ensembles of classifiers with very good performance. The key idea behind AdaBoost algorithm is to use weighted versions of the same training data instead of choosing randomly. The same training set is repeatedly used and, for this reason, it does need to be very large training set [54]. The algorithm learns a set of classifiers, using a weak learner, in order to produce the final classifier of the form. The weak classifiers are obtained sequentially, using re-weighted versions of the training data, with the weights depending on the accuracy of the previous classifiers. The training set is always the same at each iteration, with each training instance weighted according to its miss-classification by the previous classifiers. This allows the weak learner at each iteration to focus on patterns that were not well classified by the previous weak classifiers [55]. It is important to choose weak learners to obtain the base classifiers, allowing them to learn without decreasing significantly the weight of the previously correctly classified instances. If the base learner is too strong, it may achieve high accuracy, leaving only outliers and noisy instances with significant weight to be learned in the following rounds [51].

ADtree: Since ADtree has a reduced training cost and a very much reduced computation cost with respect to Adaboost, it is comparably applied for real-time applications as Adaboost. ADtree becomes more similar with Adaboost when the input data has little noise i.e. it is strong to avoid the data overfitting that is why it is applied directly in real-time applications [1].

2.3.2. Hybrid Approaches

Hybrid approaches obtain disambiguation information from both corpora and explicit knowledge-bases. Hybrid systems aim to use the strengths of both approaches to overcome the specific limitations associated with a particular approach and improve WSD accuracy [24]. These approaches can neither be properly classified as knowledge or corpus-based, since they obtain disambiguation information from both corpora and explicit knowledge-bases [27].

2.4. Applications of WSD

Word Sense Disambiguation (WSD) is vital in many Natural Language Processing (NLP) applications [56]. NLP applications can be categorized into two parts as the end tasks and the intermediate ones considering their functionalities. Machine translation (MT), information retrieval/ extraction (IR/IE), search engines, etc. are the typical end tasks in which complete solutions have been provided. On the other hand, morphological analysis, parsing and word sense disambiguation (WSD), etc. are intermediate tasks that contribute to the end tasks rather than providing overall solutions. Word sense disambiguation (WSD) is an important intermediate step in many natural language processing applications [57].

There are various applications of word sense disambiguation some of which are as follows: -

Machine Translation: It is used to remove ambiguity between different senses of a word in a given domain to convey information to a machine correctly and provide further conversion using intermediate code [32]. Word Sense Disambiguation is required in machine translation for words that have different translations for different senses. If we take literal meaning of the word from one language then it translates it to another language then sometimes the translated sentence does not give the same meaning as of original language [58].

Information Retrieval: Information Retrieval is another natural language processing application that benefits from WSD. Most of the words used to execute queries in IR systems have more than one meaning and therefore when performing a query, the system may retrieve documents which are not relevant to the search [56]. Because most of the search engines do not use explicit semantics to prune out documents which are not related to user query [58]. An accurate disambiguation of the document base, together with a possible disambiguation of the query words, would allow it to eliminate documents containing the same words used with different meanings (thus increasing precision) and to retrieve documents expressing the same meaning with different wordings (thus increasing recall) [5].

Information Extraction / Text Mining: it is required for the accurate analysis of text in many applications [58]. It focuses on extracting certain information from a single document. For example, when searching for an article on the internet, there may be results that don't match the user requirements or there may be too many similar articles in the search results. In this case, WSD

is used to remove confusion and find out the correct sense of the word [32]. WSD plays an important role for information extraction in different research works as Bioinformatics research, Named Entity recognition system, co-reference resolution etc. [10].

Speech Recognition Systems: In speech recognition systems, a WSD module can be used to increase systems´ performance by distinguishing senses for homophone and homograph words according to their context [56].

Question–answering: Question Answering is the oldest natural language processing application. In this system, specific questions are asked for example "when the computers are invented?" and it receives the concise answer rather than a set of relevant documents. To retrieve concise pages, we need to have correct sense of a particular word so word sense disambiguation plays a very important role [58].

2.5. Related Works

This gives brief description of the various solutions that are done before for word sense disambiguation. Research work on Word Sense Disambiguation started in the 1940's and a wide range of Word Sense Disambiguation algorithms has been proposed over the years. Some of them proposed a solution by using supervised approach that uses labeled data for training, some of them follows unsupervised approach that disambiguates a word without using labeled data i.e. by using unlabeled corpora, some of them proposed a solution by using knowledge-based approach to disambiguate the ambiguous words, and some of them also proposed a solution by using hybrid approaches. The task of disambiguation system is to resolve the lexical ambiguity of a word in a given context. To put it more precisely, the term "lexical ambiguity" refers to two different concepts "homonymy" and "polysemy". The distinction between bank ("river edge") and bank ("financial institution") has been used as an example of homonym, and rust (verb) and rust (noun) for polysemy. Reflecting the rapid growth in utilization of machine-readable texts, word sense disambiguation techniques have been explored variously in the context of "Corpus-Based and Knowledge-based approaches". This Chapter surveys past research associated with Corpus-Based, Knowledge-based, and hybrid approaches of word sense disambiguation for local languages only which are Tigrigna, Amharic, and Afaan Oromo.

2.5.1. WSD for Tigrigna Language

Meresa M. [21] applied corpus-based approach to develop WSD prototype model for Tigrigna language. To do this, the researcher used unsupervised machine learning method which requires information that can be automatically extracted from untagged texts. He reported his experiment on four ambiguous words of tigrigna language. Those word are: ሓለፈ (Halefe), መደብ (medeb), ሃደመ (hademe), ክበረ (kebere). The first two words have four and three senses respectively, but the rest three words have two senses each. The researcher used a total of 631 sense examples for four of the ambiguous words of the language. Those datasets were collected from different online tigrigna websites, newspapers, and Tigirgna bible. To disambiguate the selected ambiguous words of the language; the researcher used five clustering algorithms which are available in Weka 3.8.1 package. Clustering algorithms selected by the researcher for this study were Simple K-means, Single Link, Average Link, Complete Link, and Expectation Maximization. Before loading those datasets to the the selected machine learning tool which is Weka 3.8.1 package, the researcher applied preprocessing tasks like tokenization, stop word removal, stemming, normalization, and transliteration using Java NetBeans 8.0.1.

Meresa M. [21] conducted two experiments. The first experiment was showing the effect of stemming on WSD of Tigrigna language. To do this, they tried to show the results obtained before stemming and after stemming. Then they compared those results, and they found that the results achieved after stemming became higher compared to results achieved before stemming. Lastly, they concluded that stemming can enhance the performance of Tigrigna WSD. The second experiment was deteriming the standard window size for identifying the meaning of ambiguous words of Tigrigna language using clustering algorithms. Then they found that window size two-two became the optimal window sizes for Tigrigna WSD using clustering algorithms. To evaluate the system, the researcher used manually tagged texts and selected “use training set” evaluation mode which is available in Weka package. Averagically, they achieved results within the range of 52% to 77.5% for simple k-means, 67% to 83.3% for EM, 45.6% to 74.1%, for single link, 65 to 73.3% for both Average link and complete link algorithms. The researcher conluded that the best accuracy was achieved within the range of 67% to 83.3% using EM algorithm. As Roberto N. [14] indicated, the current state of the art accuracy for unsupervised learning yields within 60 to 70% range. By considering Roberto Ns’. [14] idea’, the results achieved by Meresa M. [21] became

encouraging. Even though the achieved results were encouraging, the datasets they used were less in number.

2.5.2. WSD for Amharic Language

Previously different researchers have conducted a research on WSD for Amharic language by using different approaches. Some of the WSD researches that are developed for Amharic language are: Word Sense Disambiguation for Amharic text retrieval: a case study for legal documents that was developed by Teshome Kassie [25]. Here, he has done on WSD for Amharic language. Mainly, the researcher focused on how linguistic disambiguation can improve the effectiveness of Amharic document query retrieval algorithm. To prove that, the author developed Amharic disambiguation algorithm based on the principles of semantic vectors analysis. As the researcher indicated, the disambiguation algorithm was used to improve the precision and recall measurements of information retrieval for Amharic legal texts. And the improvement of those measurements was used to develop a document search engine. He developed his algorithm using Java programming language.

The researcher used the Ethiopian Penal code as a corpus for his experiment. His corpus contained 865 articles. Those articles composed of 19, 493 words in size without applying any preprocessing tasks. 10 queries which contain Amharic keywords with ambiguous meanings were selected by the researcher in his experiment. He performed preprocessing tasks for his corpus before using on his experiment such as stemming, normalization, tokenization, and stop words removal. For enhancing the results of his information retrieval system query that he developed, he used word sense discrimination method.

Teshome [25] used a word space model which makes him to use distributional statistics that generates high dimensional vector spaces. He developed an algorithm based on distributional hypothesis stating that words with similar meanings tend to occur in similar contexts. The algorithm that developed by the researcher was semantic vector query algorithm. For disambiguation of a given word, the author computed the context vector of each occurrence of the words. The context vector was derived from the sum of the thesaurus vectors of the context words. The author constructed the thesaurus by associating each word with its nearest neighbors. Similarity between thesaurus vectors for each was calculated by cosine distance function. The

researcher also used random indexing algorithm for dimension reduction in the frequency matrix. To index the corpus, he used an information retrieval library called Lucene.

For evaluating the WSD, the author used some artificial words called pseudo words rather than real sense tagged words because it was so laborious and costly to prepare sense annotated data. However, for evaluating his information retrieval systems that he developed, he used precision and recall measurements. Those measurements were taken by formulating a set of queries for retrieval purposes. The researcher compared the results of the algorithm that he developed (semantic vector query algorithm) with Lucene algorithm and he reported that semantic vector query algorithm is superior over Lucene algorithm. By using Lucene algorithm, the author achieved 32% and 49% of average precision and recall respectively; whereas by using semantic vector query algorithm, he achieved 58% and 82% of average precision and recall respectively.

Solomon Mekonnen [23] developed word sense disambiguation for Amharic texts using a machine learning approach. The main focus of the researcher was to address the problem of automatically deciding the correct sense of an ambiguous word based on its surrounding context for Amharic texts. To decide the correct sense of an ambiguous word, he used corpus-based approach with machine learning techniques. The classification algorithm that the researcher applied in his study was Naive-Bayes classifier. To use this classification algorithm, the researcher used a machine tool called Weka 3.6.2 package because Naïve-Bayes classifier is already implemented in this machine learning tool. Before he applied the classification algorithm (or Naive Bayes) to build a classifier, he has done sentences preprocessing tasks to make them suitable for further processing in the selected machine learning tool.

The researcher used five ambiguous words for his experiment. Here, sense annotated datasets were used to disambiguate words that have different meanings depending on the contexts because he conducted his research by using supervised machine learning approach. As he described in his research, getting manually sense annotated data for Amharic was difficult. Therefore, to acquire those manually annotated data; the researcher used monolingual corpora of second language and translate to the original language. By using this method, the researcher collected the corpus from British National Corpus (BNC) and then he translated in to their Amharic meanings using dictionary. The author collected a total of 1045 sense example sentences for the five ambiguous words for the selected Amharic words. He also tried to use balanced distribution of senses for the

ambiguous words to maximize performance when enough sense examples are available. From the corpus that he collected, 90% of the datasets were applied for training and the remaining 10% of the datasets were applied for testing purposes.

The author conducted a total of five experiments by using Naïve Bayes classifier. Those experiments were: - the effect of stemming on the accuracy of the classifiers, determining the appropriate learning and testing split options by comparing 10-fold cross validation and 66% split options which are defaults in Weka 3.6.2 package, comparing 10-fold cross validation with different percentile split options, determining the effect of different context sizes on disambiguation accuracy, and investigating the effect of sense distribution on the performance of the classifiers.

Lastly, the researcher concluded: stemming could enhance the accuracy of classifiers significantly, 10-fold cross validation test split option performed better when he compared to other percentile test split options, window size of 3-3 became the most favorable window size for determining the meanings of Amharic texts, balanced sense distribution could give better result for the accuracy of classifiers compared to unbalanced sense distribution. The final accuracies of the classifier that the researcher achieved by using three-three window size ranges from 70% to 83%.

Solomon Assemu [22] also conducted a research on WSD for Amharic by using corpus-based approach. From the corpus-based approaches, He applied unsupervised approach to automatically deciding the correct sense of the ambiguous words. He investigated his experiment on five selected Amharic ambiguous words which were also used in [23]. The corpora that he used in his experiment were taken from Solomon [23] that means 1045 sense examples of the ambiguous words. To maximize the performance of his study, the researcher tried to use balanced distribution of senses of the ambiguous words when enough sense examples are available. Before he used those sense examples in his experiment, he has done preprocessing tasks to make them comfortable for training and evaluating using the selected algorithms.

Since the researcher used unsupervised approach for his study, splitting the corpus to training sets and testing sets was not required for the evaluation purpose. Here, annotated datasets were not used for clustering the sense of the ambiguous word but they were used for the evaluation of the clustering assignments.

To evaluate his experiment the researcher used sources of sense annotated corpus. He selected class to cluster evaluation mode to evaluate the discovered sense groups. The researcher selected five clustering algorithms for experimentation of his study such as K-means algorithms, Agglomerative single algorithms, Average and complete link clustering algorithms, and Expectation Maximization algorithms. He used Weka 3.6.4 package as a tool for the implementation of his experiment. The researcher used the same experimental settings and data sets with experiments presented in [23], except the approaches that he used. The main reason that he tried to use the same experimental settings and data sets was to compare the results that were prepared by using supervised approaches and unsupervised approaches. In his study he conducted four experiments by using the selected five clustering algorithms. Those experiments were: - the effect of stemming on the accuracy of the result, determining optimal context window, the effect of distribution of training data on the accuracy of the result, comparison of results found by using different clustering algorithms that were used in his study with results found by using supervised algorithm i.e. Naïve Bayes classifier.

After he conducted those experiments in his study, he concluded that stemming improves the accuracy of the results in almost all of each clustering algorithms, Window size of 3-3 considered to be effective for Simple K means and EM clustering algorithms, whereas Window size of two-two became effective for agglomerative SL and CL clustering algorithms but he couldn't determine the optimal context window of average link clustering algorithm in his experiment because the maximum accuracy was dispersed over one-one to five-five word window for each ambiguous word, the accuracy of unbalanced sense distribution resulted less accuracy for all five ambiguous words in simple k means, three of the ambiguous word (except, *eTen* and mesal) in EM and four of the ambiguous word (except, *qereSe*) in agglomerative Complete Link, and in determining optimal context window size and the effect of stemming on the accuracy of the result experiments, Expectation maximization and simple k-means clustering algorithms achieved almost the same result. In the experiments of determining the effect of sense distribution on the accuracy of the result, simple k-means algorithm performs a better result of all clustering algorithms, next to that EM performed a good result. The worst result was recorded in Average Link algorithms next to Single Link. But, the results of Agglomerative Complete Link clustering algorithm were interesting when compared to Single and Average Link clustering algorithms. In addition to comparing the results found by using different clustering algorithms, he also tried to

compare the results achieved in his study with the result that was achieved by [23] because his study tested on the same data sets and the same feature sets. Here, Solomon [23] reported that the accuracy achieved by using supervised algorithm is 70.1% to 83.2%. Whereas Solomon Assemu [22] achieved accuracy within the range of 65.1% to 79.4 % for simple k means, 67.9% to 76.9% for EM and 54.4% to 71.1% for complete link clustering algorithms for five of the selected Amharic ambiguous words.

The strength of this research is that the context of an ambiguous word is cluster in to a number of groups and discriminate these groups without actually labeling them. However, the limitation of this research is that training data is required for each word that need to be disambiguated and unsupervised method cannot rely on a shared reference inventory of sense. The approach used by the researcher is that the same sense of a particular word will have alike neighbor words and clustering word occurrences and classifying new occurrences into induced clusters.

Segid Hassen [12] is also another researcher who has done Amharic word sense disambiguation using WordNet. He used one of knowledge-based WSD approaches which is WordNet. The researcher developed Amharic WordNets contained 10,000 synsets and 2000 words. He used SQL server, java and python programming language as developmental tools in his study. Here SQL server was used for the development of Amharic WordNet, java and python programming language were used to develop the prototype of the study. Python programming language was also used to write the extraction of the root word. To evaluate the performance of the proposed system, he conducted two experiments which were determining the optimal context window size and to analyze the effect of morphological analyzer. In his experiment, he took 200 random sentences as test sentences contained the ambiguous words from the knowledge base. Those random sentences were taken from linguistic experts and newspapers. In the first experiment, the researcher conducted Amharic WordNet with morphological analyzer and without morphological analyzer for comparison purpose. At that time, he obtained better result in Amharic WordNet with morphological analyzer than without morphological analyzer. The author achieved an accuracy of 80% in Amharic WordNet with morphological analyzer and 57.5% without morphological analyzer. The second experiment was carried out to test 1-1 window up to 5-5 window on both side of the target word for some ambiguous words. After conducting the second experiment, the researcher compared the results for each window size; and he achieved higher accuracy at two-

two window size i.e. 86.5%. Therefore, depending on his study two-two window size became the most favorable window size for Amharic language.

Getahun Wassie [1] became another researcher who has developed WSD model for Amharic language. To develop the model, he used semi-supervised machine learning approach. He used semi-supervised machine learning approach to balance the drawbacks of both supervised machine learning and unsupervised learning approach because semi-supervised approach requires small seed examples to train the WSD model of the language. The researcher has done his work by incorporating unlabeled data with some seed instances directly into the data representation, after that they applied unsupervised algorithms for clustering based on instance similarity. Next to that they applied supervised algorithms for classification. By taking those large amounts of unlabeled data with some labeled examples; he built better classifiers with less human efforts.

He used a semi-supervised method called bootstrapping to build WSD model. He prepared a corpus which contain 5 ambiguous words with a total of 1031 sense example sentences of Amharic language. Those 5 ambiguous words were selected from Amharic dictionary. Getahun W. [1] collected those sentences manually. The researcher conducted four experiments in his study. The first experiment was Analysis of the effect of training seeds options and the second was comparison of the three machine learning methods. The third and fourth experiments were investigating the best performing classification algorithm and finding the optimal window size for the selected datasets of the language. Those experiments were conducted by using Bootstrapping algorithms (ADTree, AdaBoostM1, and Bagging), SVM algorithm (SMO), and Bayes algorithm (Naïve Bayes) for classification purpose. He also used two clustering algorithms (K-means, EM) for clustering purpose. For classification purpose of his datasets, he used the same classification algorithms for both supervised and semi-supervised machine learning methods. Weka package was used as a tool for clustering and classification tasks because both clustering algorithms and classification algorithms are built in Weka package.

After conducting his experiment, he concluded that semi-supervised machine learning method performs better compared to the other machine learning methods and ADtree algorithm results the highest accuracy using semi-supervised machine learning methods compared to the other selected classification algorithms. To find the optimal context window size for WSD model that he developed, he tried one–one to ten-ten window sizes. After that he recommended that window

size of 3-3 became the optimal window size for Amharic datasets using semi-supervised machine learning. The experimental result shows performance score of 88.45%,87.89%, 84.90%, 81.25%, and 67.01% using ADTree, SMO, AdaboostM1, Bagging, and Naïve Bayes algorithms respectively.

Biruk Reta [59] also another researcher who conducted a research on application of part-of-speech tagged corpus to improve the performance of WSD in the case of Amharic language. The researchers selected the ambiguous words which were used by Getahun W. [1]. They also used the same clustering algorithms, classification algorithms, and number of datasets with [1]. The only thing that Biruk Rs'. [59] study differs from Getahun Ws'. [1] study was applying part-of-speech tagging for the datasets that were used by [1]. They used all the methods and techniques that [1] was used in his study. They tried to investigate the effectiveness of part-of-speech tagged corpus for WSD of Amharic language. They used CRF POS tagger that developed using CRF $^{++}$ tool in order to attach POS tag to all of the words that they used in their study. Since all the classification and clustering algorthms that they used in their study are available in Weka 3.6.11; they used Weka 3.6.11 for developing the WSD prototype model of Amahric. 10-fold cross validation and classes to clusters evaluation modes were used for measuring performance of classification algorithms and clustering algorithms respectively.

They also tried to conduct experiments on the three machine learning methods by applying POS tagger for their corpus in each machine learning methods; and they obtained better result using semi-supervised machine learning rather than the other two machine learning methods which are unsupervised and supervised machine learning methods. Using Semi-supervised machine learning methods; they achieved a score of 92.66% using ADtree, 92.33% using AdaboostM1, 89.92% using SMO, 80.98% using Bagging and 60.62% using Naïve Bayes algorithm. Therefore, the researcher obtained an attractive result when we compared with results obtained by [1]. They also considered that window size of 6-6 became enough for deteriming the sense of the ambiguous words when they apply part-of-speech tagged corpus for the ambiguous Amharic words. Lastly, we recommend other researcher to conduct researches for Tigrigna language by applying POS tagger for the corpus that we used in this study.

2.5.3. WSD for Afaan Oromo Language

Afaan Oromo is one of the major languages that are widely spoken in Ethiopia. Different natural language tasks are developed for Afaan Oromo language. WSD is one of the natural language tasks that was developed for this language. Different researchers have developed WSD for Afaan Oromo language by using different approaches. Some of them are: -

Tesfa K. [11] has done WSD for Afaan Oromo language by using a corpus-based approach. From the corpus-based approaches, he applied supervised machine learning approaches to automatically disambiguate the ambiguous words of the language. He also applied Naïve Bayes' theorem to find the prior probability and likelihood ratio of the sense in the given context. For his study, the researcher collected a total of 1240 Afaan Oromo sense examples for five selected ambiguous words namely Sanyii, Karaa, horii, Sirna, and qoqhii. Those sense examples were annotated manually by language experts. He collected his corpus from different magazines, bulletins, and Afaan Oromo newspapers. The researcher used sense inventory to select word senses of the ambiguous word, a training corpus as a knowledge source, and preprocessing of the input text in order to represent the context. He also tried to use balanced distribution of senses for each ambiguous word to maximize the performance of his study.

By considering the nature of Afaan Oromo language, the architecture of his study used two phases' i.e. Preprocessing phase and Disambiguation phase. Preprocessing phase was required to prepare the data for further processing. The researcher adopted stop word remover algorithm and stemmer algorithm from another researcher who has been developed a stemmer for Afaan Oromo language. The researcher has applied the second phase which was disambiguation phase by incorporating AWI, sense identifier, context extractor, sense counter, context counter, and disambiguation modules. He implemented his own algorithm for each of the module that he incorporated in his study. To compute the score of each sense of the ambiguous words, Tesfa K. [11] has used a learning algorithm called Naïve Bayesian classification techniques. To develop the prototype of his work, the author used a Java programming language.

Tesfa K. [11] has conducted learning curve analysis that best suits to detect goodness of the size of the dataset to evaluate his work. Based on the learning curve analysis the researcher concluded that datasets were not sufficiently large. Therefore, he used a statistical technique called k-fold

cross-validation in order to evaluate the performance of his research. He also used the most commonly used evaluation metrics to measure the rate of disambiguation of his system i.e. precision P, recall R, F1-measure, and Accuracy Acc. Two experiments were conducted in his study. The first experiment was conducted to evaluate the performance of the algorithm that he was applied in his study and the second experiment was conducted to investigate the optimal context window size for his study. In the first experiment, the author was achieved 76% precision, 88% recall, and 81% F1-measure, and 79% accuracy. Depending on this experiment, the researcher concluded that the achieved result was satisfactory in terms of accuracy and it could also improve the accuracy of other Afaan Oromo NLP applications. In the second experiment of his study, he tried window sizes starting from one-one to ten-ten to find the optimal window size for WSD of Afaan Oromo language for all of the five ambiguous words that he selected. Then by calculating the average accuracy of each window size for all ambiguous word, four-four window size achieved greater results. Therefore, four-four window size was effective for his work. Supervised machine learning approach of WSD performs better by human intervention; however, this research has limitations of knowledge-acquisition bottleneck, i.e., it requires manually labeled sense examples which takes much time, very laborious and very expensive to create when the corpus size increases and training data is required.

Yehuwalashet B. [60] is also another researcher who has done WSD for Afaan Oromo language by using Hybrid approach. The main objective of his work was to design and test a hybrid system which finds the meaning of words based on the surrounding contexts combining unsupervised approach with rule-based approach. It was presented by combining unsupervised approach that exploits sense in a corpus and the manually crafted rule using hybrid approach. The researcher has done his research by using quantitative experiments. He used hybrid approach in his study because it depends on the available and more reliable linguistic knowledge besides the semantic techniques and training methods which consider the role of the words in its context. He collected his corpus from different sources like newspapers, Bible, news (Oromia Radio and Television Organization, magazines, historical documents and bulletins.

He trained his system by using unannotated corpus, which contains ambiguous words, and tested by using 20 ambiguous words with a total of 62986 lines of sentences. Java Net Beans 8.0.2 and Weka 3.7.9 package were selected for implementing WSD model of his work. Weka 3.7.9 package

was used to cluster by using hierarchical clustering, EM and K-means clustering algorithms built in this package. To measure the distance between clusters, the researcher used a default Euclidian distance. Java Net Beans 8.0.2 programming language was used in this research for the purpose of preprocessing activities i.e. tokenization, stop word removal, and normalization.

He developed two kinds of approaches towards the word sense disambiguation. The first one was completely machine learning approach in its nature. Here machine learning was used to extract two important features automatically without applying linguistic rule i.e. all possible contexts of the ambiguous words and their clustering. The second approach was combining machine learning approaches with rule-based approach (called hybrid approach). The researcher built this approach by combining the rule learned from linguistic feature of Afaan Oromo with the semantic feature learned from the corpus using machine learning approach i.e. unsupervised machine learning approach. The researcher also used the manually crafted linguistic rule to extract the various contexts assumed by the ambiguous word. After that he relied on machine learning algorithm to cluster those contexts. In order to extract the two important features, the following procedures were applied in his study: - extracting context words by using distributional hypothesis, clustering the extracting context words using hierarchical clustering, K-means and EM algorithms by constructing vector space matrix. Experiment on machine learning approach using different context window sizes and clustering algorithms were conducted. This experiment was done by using five clustering algorithms such as, K-means algorithm, Expectation Maximization (EM), Single link algorithm, complete link algorithm, and Average link algorithm. In this experiment, window sizes of up to ±10 words on both sides of the ambiguous words in the above clustering algorithms were tried. By applying this experiment, the researcher concluded that smaller window sizes have yield significantly higher accuracy than other window sizes. The best accuracy that achieved by this experiment was 80.6% with EM clustering algorithm in the window size of one-one.

Experiment on the hybrid approach was also another experiment that conducted in his study. The main objective of conducting this experiment in his study was to answer “does the addition of linguistic knowledge to unsupervised machine learning approach improves a WSD performance of Afaan Oromo ambiguous words?”. By using this experiment, the researcher obtained better results when compared to the first experiment that he was conducted i.e. Unsupervised machine

learning approach for each clustering algorithms and window sizes. According to the researcher, the EM and K-Means clusters which yield significantly higher accuracy than other clusters when compared with machine learning result. The researcher achieved 90.35% and 86.28% by using EM algorithm in one-one and two-two window sizes respectively; 86.6% and 83.55% by using K-means algorithm in one-one and two-two respectively. According to results he has been achieved, the accuracy obtained by using both EM and K-means algorithm in hybrid approach is better than results obtained in machine learning approach. Based on the result he has obtained, he selected one-one and two-two window sizes were the optimal window size for Afaan Oromo WSD.

The evaluation of his system was undertaken on the basis of precision and recall. In evaluation of his system, he evaluated the system that was modeled by using machine learning approach independently and by using hybrid approach in order to compare whether the addition of linguistic knowledge to unsupervised machine learning approach improves the accuracy of the WSD or not. By using unsupervised machine learning approach, he achieved 76.05% and 65% precision and recall respectively. Whereas in hybrid approach, he achieved 89.47% and 85% precision and recall respectively. At the last he concluded, the hybrid approach shows promising result as compared to the performance of independent machine learning approaches to WSD task.

2.6. Summary

Researches on WSD have a history as long as natural language processing started and faced problems in 1950s [9]. Obviously, the problem in word sense disambiguation can be solved through different approaches and techniques. Different researchers have been conducted a research for different languages using different approaches mainly using knowledge-based approach and corpus-based approach. Knowledge based approach uses external knowledge sources to disambiguate words and corpus-based approach on the other hand employees different machine learning techniques to automatically assign meaning to words. We reviewed WSD researches conducted for local languages like Amharic and Afaan Oromo that have been done by using corpus-based approach and knowledge-based approach. From our review, the best result was obtained with the works done by using corpus-based approach. We also reviewed WSD research conducted for Tigirgna language using unsupervised machine learning and they achieved encouraging results. But, they have limited datasets. Therefore, we conducted our study by adding datasets and by changing a machine learning method to semi-supervised inorder to compare the results achieved by using both machine learninhg methods which means unsupervised and semi-supervised.

Chapter Three

3. Tigrigna Language

3.1. Overview of Tigrigna Language

Ethiopia is a linguistically diverse country where more than 80 languages are used in day-to-day communication. Tigrigna is one of these languages which is used in day-to-day activities especially in northern part of Ethiopia. Tigrigna is a Semitic language of the Afro-asiatic family originated from the ancient Geez language. Tigrigna is the designated official language of Tigray region in Ethiopia and is the language most widely spoken in this region. It is also one of the national languages of Eritrea. It is closely related to Amharic and Tigre. It is estimated to be spoken by over six million people worldwide [61]. It is also a medium of instruction in primary schools in the aforementioned areas and other Tigrigna speaking people. It has become the primary language of oral and written communication in Eritrea and Tigray. In some institutions of Tigray like Tigray Teachers training institution, all the subjects are taught in Tigrigna for elementary teachers. It is also thought as a subject in the junior and senior secondary schools of Eritrea and Tigray [62].

3.2. Tigrigna Writing System

The Ethiopic writing system is used to represent Semitic languages. some of the semitic language that are represented using Ethiopic writing system are: - Ge'ez, Amharic, Gurage, and Tigrigna. These languages are confined to Ethiopia and Eritrea. Ge'ez is a dead language. It is no more mother tongue of any person, but it still has a very significant role in the traditional language of literature and religion. Amharic and Tigrigna are closely related to each other. Ethiopic writing system is written from left to right. It does not make any distinction between upper and lower-case letters and has no conventional cursive form. There are no systematic variations in the form of the symbol according to its position in the word [15].

The writing system is syllabary. In syllabary type of writing system, every script/symbol represents a combination of consonant and vowel. Ethiopic is a name given to writing system of Eritrea and Ethiopia. The name Ethiopic is used by linguists mostly from foreign origins. Locally, in Eritrea

and Ethiopia, the writing system is known affectionately as “Geez”, “Fidel”, and “Fidelat”. Therefore, Tigrigna language is written in Ethiopic scripts that means it is written from left to right. It has its own writing system and its own characters (alphabets), punctuation marks, and numbers systems [18]. Having its own alphabets, punctuation marks, and number systems, users of the language are used for writing different documents of language. It has thirty-five base symbols with seven orders which represent seven vowels for each base symbol. The writing system of the language is borrowed from Geez which is another Ethiopian Semitic language [63].

Each Tigrigna language scripts occur in one basic form and six other forms that may be describes as orders. The normal syllable in Tigrigna is considered to be a consonant followed by a vowel. If a consonant ends a syllable, the sixth, neutral vowel is used with it. Most consonants are written in seven slightly different forms corresponding to the traditional seven vowels. These seven orders (the first basic order and the other six orders) represent the different sounds of a consonant-vowel combination (a characterization known as syllabic) [64].

The Tigrigna writing system can be translated in Latin representation by finding a Latin letter with similar sound Tigrigna letter. For example, the Tigrigna letter ‘ማ ‘has a similar sound with Latin letter ‘m’. As a result, the seven orders of the letter ‘ማ ‘can be represented by combining the Latin letter ‘m’ with vowels as shown below in table 3.1.

Table 3.1: Sample of Tigrigna letter and their corresponding Latin letter

Order	1st	2nd	3rd	4th	5th	6th	7th
Tigrigna letter	ማ	ሙ	ሚ	ማ	ሜ	ም	ሞ
Equivalent Latin letter	Me	mu	mi	ma	mie	m	mo

As shown in table 2.1 the Tigrigna letter and its seven orders can be translated in to similar sound Latin letter. The seven orders of the Tigrigna letter can be translated by combining the similar sound Latin letter and vowels. One can understand from the above table that the first order in Tigrigna represents vowel /e/, the second vowel /u/, the third vowel /i/, the fourth vowel /a/, the fifth vowel /ie/, and the seventh vowel/o/. The sixth order represents the consonant with no vowel. Unlike English, Tigrigna does not have upper and lower-case letter representations.

As Leslau [63] noted, Tigrigna writing system have also its own writing system problems like redundancy of some characters, spelling variation of the same word, abbreviation, compound words. When we say that redundancy of some characters, in Tigrigna language there are some alphabets that have the same pronunciation but different symbols. For instance, letters like "ሀ" and "ኀ"; "ጸ" and "ፀ"; "ሰ" and "ሠ" have similar sounds in this language. Here only one them can be used interchangeably without changing their meaning. Therefore, the use of various forms of characters for the same sound has a problem in the process of feature preparation for the classifier learning. Hence, understanding Fidel usage in words is the expected task to deal with words. If a word is written in a consistent Fidel, it provides efficient and accurate natural language processing systems. Such types of word ambiguities can be handled through character normalization.

3.3. Tigrigna Punctuation Marks

Identifying punctuation marks is very important to know word demarcation for natural language processing. The Tigrigna writing system uses some indigenous and foreign (or English) punctuation marks in addition to the Tigrigna characters. Some of the most commonly used Tigrigna punctuation marks are: -**The word separator mark** (:) is used in the old literature to separate one word from other words. In the current literature, it is rarely used. As, a result a single space is used to separate words instead of this punctuation marks. **The end of sentence mark** (: :) is used to shows when an idea is finished. **The sentence connector mark** (፤) is used to connect two sentences in to one sentence. **The list separator mark** (፥) is used to list things, separate parts of a sentence, and indicate a pause in a sentence or question. Like the other punctuation marks, **the beginning of the list mark** (፡-) is used at the beginning of the lists. In addition to these punctuation marks, the Tigrigna language is also borrowed some punctuation marks from English language such as ?,!, and ". Those borrowed punctuation marks are used in Tigrigna language. The removal of punctuation marks increases the effectiveness and efficiency of natural language processing systems as morphological analyzer and stop word removal does [65].

3.4. Tigrigna Morphology

Morphology (ስነ ምእላድ) is the branch of linguistics that deals with the internal structure of words and word formation, including affixation behavior, roots, and pattern properties. Morphology is the main source of variation in natural language text, with suffixing and prefixing being the most

common ways of creating a word variant. Morpheme (ምእላድ) is the minimal units of morphology. Morphemes in Tigrigna can be categorized as free (ነፃ) and bound (ፅግዕተኛ) morphemes. Free morphemes are morphemes that can stand on their own to give meaning; whereas, bound morphemes are morphemes that cannot stand on their own to give as a word. For example, the Tigrigna word "ፍፃመታት" contains two morphemes i.e. "ፍፃመ" and "ታት". From this the first word "ፍፃመ" is free morpheme and the second word "ታት" is bound morpheme [66]. Morphology can be classified as either inflectional or derivational. Inflection is variation or change of form that words undergo to mark distinctions of case, gender, number, tense, person, mood, voice, comparison. Inflectional morphology is applied to a given stem with predictable formation. It does not affect the word's grammatical category, such as noun, verb, etc. Case, gender, number, tense, person, mood, and voice are some examples of characteristics that might be affected by inflection. Derivational morphology, on the other hand, concatenates to a given word a set of morphemes that may affect the grammatical and syntactic category of the word [20].

Tigrigna Like other Semitic languages such as Arabic and Amharic exhibits a root pattern morphological phenomenon. Each word group of the language generates an increased verb forms and noun forms by the addition of derivational and inflectional affixes. Words in Tigrigna are built from the roots by means of a variety of morphological operations such as compounding, affixation, and reduplication. These words may be new in meaning and structure from their respective roots [13].

Tigrigna affix is a morpheme that can be added at the end, beginning or middle (inside) of the root to create the inflectional or derivational morphology of Tigrigna words. Tigrigna affixes have the feature of concatenating with each other by governing morphological and syntactic rules of the language. This feature increases the overall number of affixes. Tigrigna is highly productive both derivationally and inflectionally. Definite articles, conjunctions, particles and other prefixes can attach to the beginning of a word, and large numbers of suffixes can attach to the end. A given head word can be found in huge number of different forms. Tigrigna affixes can be classified in to four categories as prefixes that come at the beginning of the root, such as ን፣ዝ፣እንተ፣ም፣ብም፣...፣suffixes that come at the end of the root, such as ና፣ታት፣ት፣ነት፣ን፣... ፣infixes that come inside the root, such as ባ in ሰባበረ፣ላ in በላልዐ፣ታ in ሰታተየ፣... and circumfixes that are attached before and after the base form at the same time such as ኣይበለዐን. While circumfixes formally are

combination of allowed prefixes and suffixes, they have to be treated as discontinuous units for semantic and grammatical reasons. Tigrigna non-concatenative morphology refers to reduplicated morpheme forms. Reduplicated words based on morpheme regularity are grouped into full reduplication (e.g., the word ስትይስትይ is derived from the stem ስትይ) and partial reduplication of different kinds. The latter includes reduplicated stems with affixes (e.g. word ሰባበረ is derived from stem "ሰበረ" sebere, "ተረጋገመ" 'teregagemu' is derived from stem "ረገመ" 'regeme', the word "ገልጠምጠም" 'gelTemTem' is derived from the stem "ገልጠም" 'gelTem') and there also various irregular reduplications [67].

Morphological variations of words in morphologically rich languages may have some consequences in some natural language applications or tasks. In such language, ambiguous words might occur in several morphological forms. Hence, without morphological analysis it would be impossible, even to identify these forms as ambiguous word forms, for assigning the correct sense. Therefore, morphological-analyzer plays an important role in WSD by reducing the different forms of an ambiguous word into their root forms [11]. As we explained in the above, Tigrigna language is one of the morphologically rich languages. Therefore, morphological analyzer might very important for this language.

3.5. Ambiguities in Tigrigna Language

Ambiguity is an attribute of any concept, idea, and statement or claim whose meaning, intention or interpretation cannot be definitively resolved according to a rule or process consisting of a finite number of steps. In ambiguity, specific and distinct interpretations are permitted (although some may not be immediately apparent), whereas with information that is vague, it is difficult to form any interpretation at the desired level of specificity. The ambiguity words in a language raised by speakers or writers can be disambiguated for understandability by listeners or readers.

As stated in [1] there are six types of ambiguity in Amharic and those ambiguities are also present in Tigrigna similar. Those are: - Phonological, Lexical, Structural, Referential, Semantic and Orthographic ambiguities. We now summarize each type of ambiguity and some of the examples are adopted from [1] by translating to Tigrigna.

3.5.1. Orthographic Ambiguities

Orthographic Ambiguity is resulted from geminate and non-geminate sounds. The ambiguity can be resolved using context [1]. For example,

አገልግል ኢላትኒ።

Agelgil elatny.

The word "Agelgil "is the cause of ambiguity. The sentence is ambiguous between the following meanings. When we geminate the word, its meaning is "**She ordered me to give service**" and when we use non-geminate sounds, its meaning is "**She asked me to bring a dish**".

3.5.2. Phonological ambiguity

It is a result due to the sound used for the word from the placement of pause with in a structure which occurs in speech. It can be illustrated through the following example:

[እቲ ቆልዓ ተመን + እዩ] ። (Ati qolOa temenyu) or (Ati qolOa temen Ayu)

He is misled person.

In the above sentence '+' sign shows where the pause is. When the sentence is pronounced with pause it, its meaning is "**He is misled or deceive person**" but the meaning differs if it is pronounced without pause. It will mean "**He is boring or tedious person**".

3.5.3. Lexical Ambiguity

Word meanings in more than one sense can lead to different interpretations by different individuals. The scope of lexical ambiguity is on individual words or word-level understanding. Lexical ambiguity refers to a case in which either a lexical unit belongs to different part-of-speech categories with different senses, or to a lexical unit for which there is more than one sense, while these different senses fall into the same part-of-speech category [68]. There are three factors that can cause lexical ambiguity which are: categorical ambiguity, Homonymy and Homophonous Affixes and synonymy [1].

Categorical ambiguity: it refers an ambiguity which results from lexical elements which have the same phonological and homographic form but belongs to different word class [69]. For example, let's look the following sentence:

ሕዳር ሂባትኒ። [Hdar hibatni]

In this example, the word “Hdar” is an ambiguous because it has both nominal and verbal meaning.

i. She gave me on a month of “November”. [with nominal meaning]

ii. She gave me unplowed farmland. [with verbal meaning]

Homonymy: Homonymy is the state of a given word’s having the same phonological form (and possibly with same spelling) however with different meanings which will cause ambiguity [1]. For instance, the Tigrigna word “ክበፅሕ” or ”kbeSH” have two different meanings which are “arrival” and “show sadness”. The following sentence shows the two different meanings.

ፅባሕ ክበፅሕ ኣለኒ። [SbaH kbeSH aleni]

i. Tomorrow I have to show sadness because somebody was died.

ii. Tomorrow I have to arrive somewhere.

Homophonous Affixes: This ambiguity is resulted when affixes serve as different word classes. The word can be morphologically analyzed into three separate morphemes: the prefix, the root and suffix. Since the meaning of each morpheme remains the same across words, it creates similar words that are difficult to understand [4]. The following example show how homophonous affixes cause ambiguity.

ሓሰሩ ነዲዱ። [Haseru nedidu]

The above sentence is ambiguous because the suffix /-u/ serves as a definite article or as a third person masculine marker. It has two different meanings:

i. The straw is burned-out.

ii. His straw is burned

Synonymy: words that are very close in meaning. For example, the Tigrigna words like “ደስታ [Desta]” and “ሓጎስ [Hagos]” have the same meaning i.e. means “happiness”. Those words may be written in different areas to refer the English word “happiness”.

3.5.4. Semantic Ambiguity

Semantic ambiguity determines the possible meanings of a sentence by focusing on the interactions among word-level meanings in the sentence. Polysemic, idiomatic and metaphorical constituents are some of the possible causes of semantic ambiguity. Brief examples are described below:

Polysemy: Many ambiguity words can have different meaning by stressing some characters in the word while reading because most words are polysemy having a range of meanings. Since polysemy word is a word with different meanings and, therefore results in the rise of ambiguity problems, that again becomes the first issue whenever these words are used in natural language possessing (WSD) systems. Polysemous words depend on the linguistic context to determine their meaning. This makes polysemous words more difficult to disambiguate than other words. They can cause ambiguity not only for semantic ambiguity but also for others too [1]. Example of semantic ambiguity due to polysemic constituent:

ደስታ የለን። [Desta yelen]

The above sentence has two different meanings. Those are:

i. There is no happiness.

ii. Desta is not there. [here Desta is the name of the person]

Idioms: refer to an expression that means something other than the literal meanings of its individual words [1]. Example of semantic ambiguity due to Idiom:

ብዕራይ ወለደ። [bOray welede]

The literal meaning of the above sentence is "**An ox gave birth to a calf**" but the idiomatic expression refers to "**impossible to happen**" or "**something unheard of**".

Metaphors: have literal or non-literal (metaphoric) senses.

Examples of sematic ambiguity due to metaphors:

a. ሓራስ ኣድጊ። [Haras adgi]

It has two different senses. The literal meaning of the above sentence is "**Donkey with new-born cubs"** and the metaphoric sense is "**Generous act, kind**".

b. ሓራስ ነብሪ። [Haras nebri]

This sentence has two different meanings due to metaphor. The literal meaning of the above sentence is "**leopard with new-born cubs**" and the metaphoric sense is "**Irascible, hot tempered**".

3.5.5. Structural Ambiguity

Structural ambiguity results when word order becomes in unorganized manner and holds more than one possible position in the grammatical structure of the sentence. Syntactic disambiguation

of words that can function as multiple parts-of-speech is accomplished by reorganizing the order at the syntactic level [12].

Example: ናይ ዓረብ ታሪክ መምህር። [Nay Oareb tariK memhr]

The above sentence has two different meanings:

i. A person who teaches arab history.

ii. An arab who teaches history. [a person from arab family who teaches history]

3.5.6. Referential Ambiguity

Referential ambiguity is a type of ambiguity which arises when a pronoun has more than one possible antecedent. Let's look the following example:

ተኽላይ ፈተና ስለዝሓለፈ ተሓጒሱ። [Teklay fetena slezHalefe teHaGUisu]

The above sentence has two different meanings:

i. Teklay was happy because he passed the exam.

ii. Somebody was happy because Teklay passed the exam.

Chapter Four

4. Word Sense Disambiguation System Design and Architecture

This Chapter describes the design of WSD system for Tigrigna language. It mainly focuses on how the corpus is prepared, design requirements, architecture of Tigrigna WSD prototype model, document preprocessing techniques, preparing machine readable datasets, and evaluation techniques of the model.

4.1. Data Collection

In this experimental research, the researcher has used corpus-based approach. It is challenging to acquire sense annotated corpus for WSD studies due to lack of standard sense annotated corpus or context-based repository (Wordnets) for Tigrigna language. Due to this reason, data collection becomes first rather than corpus preparation. By considering this, the researcher collected data from different sources like Meklah Tigray, Dimtsi Weyane Tigray, Aiga AHferom, Tigray Radio and Television organization (TRTO), Demhit, Tigray youth site, VOA news, bible and newspapers like Weyin, Mekalh, Wirayina, and constitution. The researcher also collected data from Eritrea sites like Hadas_Eriterea, Eritrea Radio and television organization, etc. Social media was also one of the sources of data collection for this study. Here the researcher first collected a huge data that contains around 90,000 sentences. Those huge data were collected online. To retrieve the sentences that contain the selected ambiguous words; we develop a simple algorithm. This simple algorithm accepts a word (OR an ambiguous word) from the user and then displays the sentences that contain this ambiguous word. The algorithm is developed using python programming language.

Collecting data from different sources can avoid bias between senses of ambiguous words. As [70] explained in his research, selecting sentences of ambiguous words from a variety of domains is very important to build efficient and reliable WSD prototype since similar domains usually restrict words to one sense. Therefore, in order to build efficient and reliable WSD prototype for Tigrigna language, the researcher collected data from different domain areas. The following Figure 4.1. shows how the corpus is prepared from the collected data:

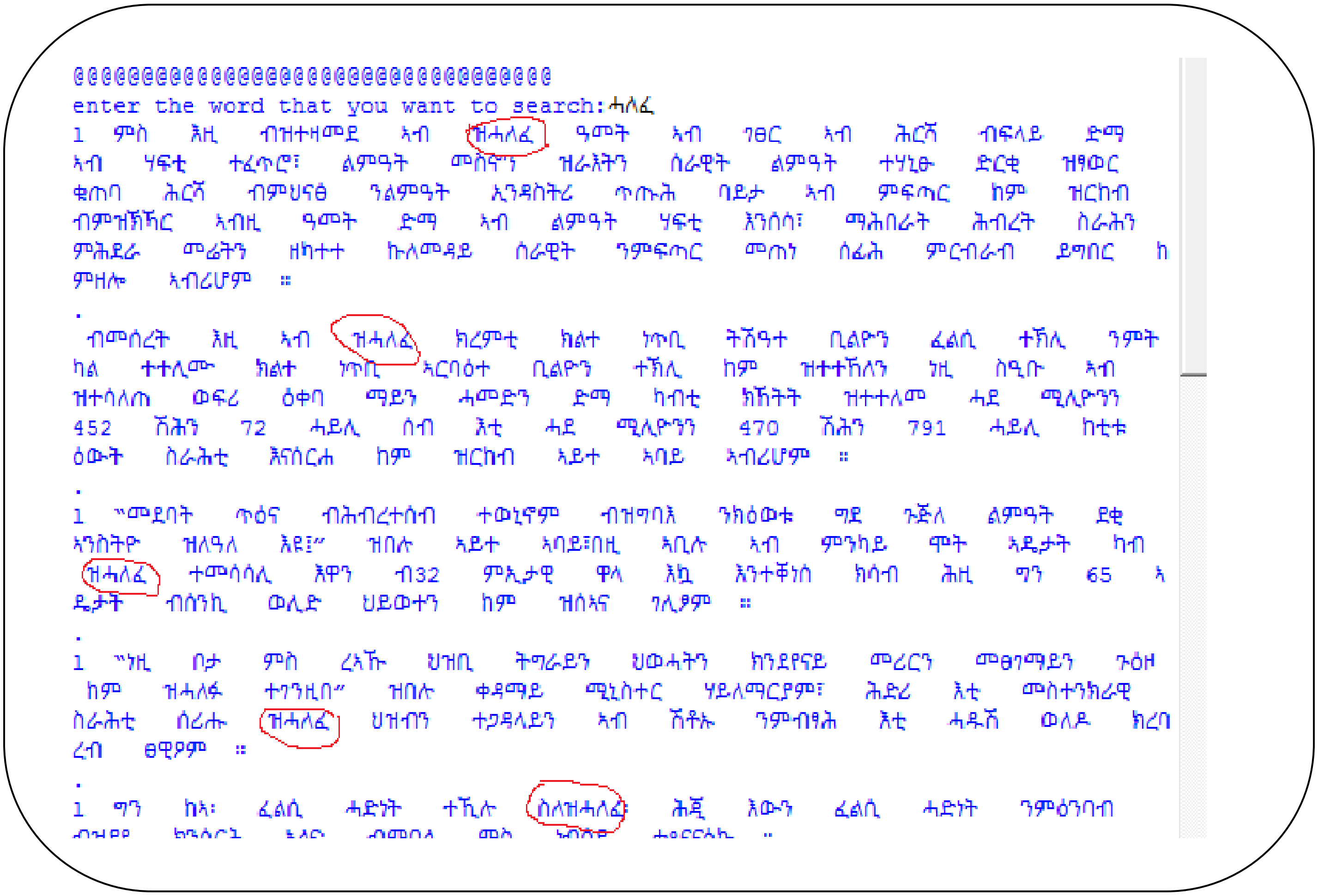


Figure 4.1: Searching result

As it shown in Figure 4.1, the algorithm asks us to enter the word that we want to search from the whole sentences that we have collected. For example, the word "ሓለፈ" is one of the ambiguous word of the language that we select in our study which means we need to have sentences which contain this ambiguous word. To do this, the algorithm that we develop asks us to enter the word to be searched. Since, Tigrigna language is morphologically rich; words of the language can be written in different formats. Therefore, we identified the formats of the selcetd ambiguous words manually. For instance, the word "ሓለፈ" can be written as "ሓሊፎም", "ሓሊፈን", "ሓሊፉ", "ሓለፉ", etc. Knowing diiferent formats of the word help us to identify sentences which contain that ambiguous word because we could search by using those different formats of the word. This kind of searching method also used for all of the ambiguous words that we used in our study.

4.2. Corpus Preparation

A corpus (plural corpora) is a collection of texts used for linguistic analyses, usually stored in an electronic database so that the data can be accessed easily by means of a computer. WSD systems need well organized datasets for training to make their accuracy attractive. But in Tigrigna language there is absence of standard sense annotated corpus and wordnets. Therefore, corpus preparation becomes must after data collection; And then the researcher prepared a corpus from the documents that we collected from different sources. Because of the absence of standard documents for Tigrigna language, the researchers' corpus was limited to a total of 1250 Tigrigna sentences which contain 5 ambiguous words of Tigrigna language to develop the proposed prototype model. Those ambiguous words are selected from Tigrigna dictionary [71]. Selected ambiguous words with their corresponding number of sentences are shown below in table 4.1.

Table 4.1: selected ambiguous words with their sense examples

Ambiguous word	senses	Sense examples	total
ሰዓረ	winning	103	204
	Not Fasting	101	
ከፈለ	Pay	102	204
	Divide	102	
ዓረቐ	Negotiate	82	164
	Poverty	82	
መደበ	Plan	102	274
	Grouping or Assigning	102	
	Traditional Bed	70	
ሓለፈ	Pass	101	404
	Pass Away (Die)	101	
	Promote	101	
	Lead	101	
Total			1250

Accuracy of machine learning algorithms degrade significantly when the training and testing samples have different distributions for the senses. In this study, we tried to use a balanced distribution of senses for the ambiguous words to maximize performance when enough sense examples are available. Balanced sentences were acquired for each sense of ambiguous words with the exception of one senses (ጦይሰ) on which enough example senses were not acquired from the collected data.

The sense of the ambiguous word is determined by the words which are surrounding it i.e. means by using context words which are surrounding the target word. The set of surrounding words to the left and right of the target word is called window, which is used for disambiguation. The target word is identified and written at the middle of the sentences. The maximum window size that we prepared for this research is 10 to the left -10 to the right of the target word. As Eker [70] explained the importance of window experiment is to determine which window size is optimal for the developed WSD of languages.

4.3. Proposed System Architecture

The flow of activities that are used to develop the proposed WSD model has been presenting using the proposed architecture. The architecture of the proposed WSD model is presented below in Figure 4.2, 4.3, and 4.4. The proposed system architecture contains different steps. The first step is accepting sentences that contain ambiguous words. The next step is applying preprocessing activities like tokenization, stop word removal, stemming, transliteration and normalization in all of the three machine learning methods which are unsupervised, supervised, and semi-supervised. Figure 4.2 shows that 1250 unlabeld datasets are given to the selected clustering algorithms inorder to have fully labeled datasets. Whereas, figure 4.3 shows 1250 labeled datasets are given to the selected classification algorithms to build WSD prototype model of Tigrigna language. The third diagram which is figure 4.4 shows that few number of labeled seed examples together with large number of unlabeled datatasets are given to the selected clustering algorithms inorder to have fully labeled datasets based on labeled seed examples. Those fully labeled datasets are given to the selected classification algorithms to build WSD prototype model of the language. Detailed description of preprocessing techniques is presented in the next subsection.

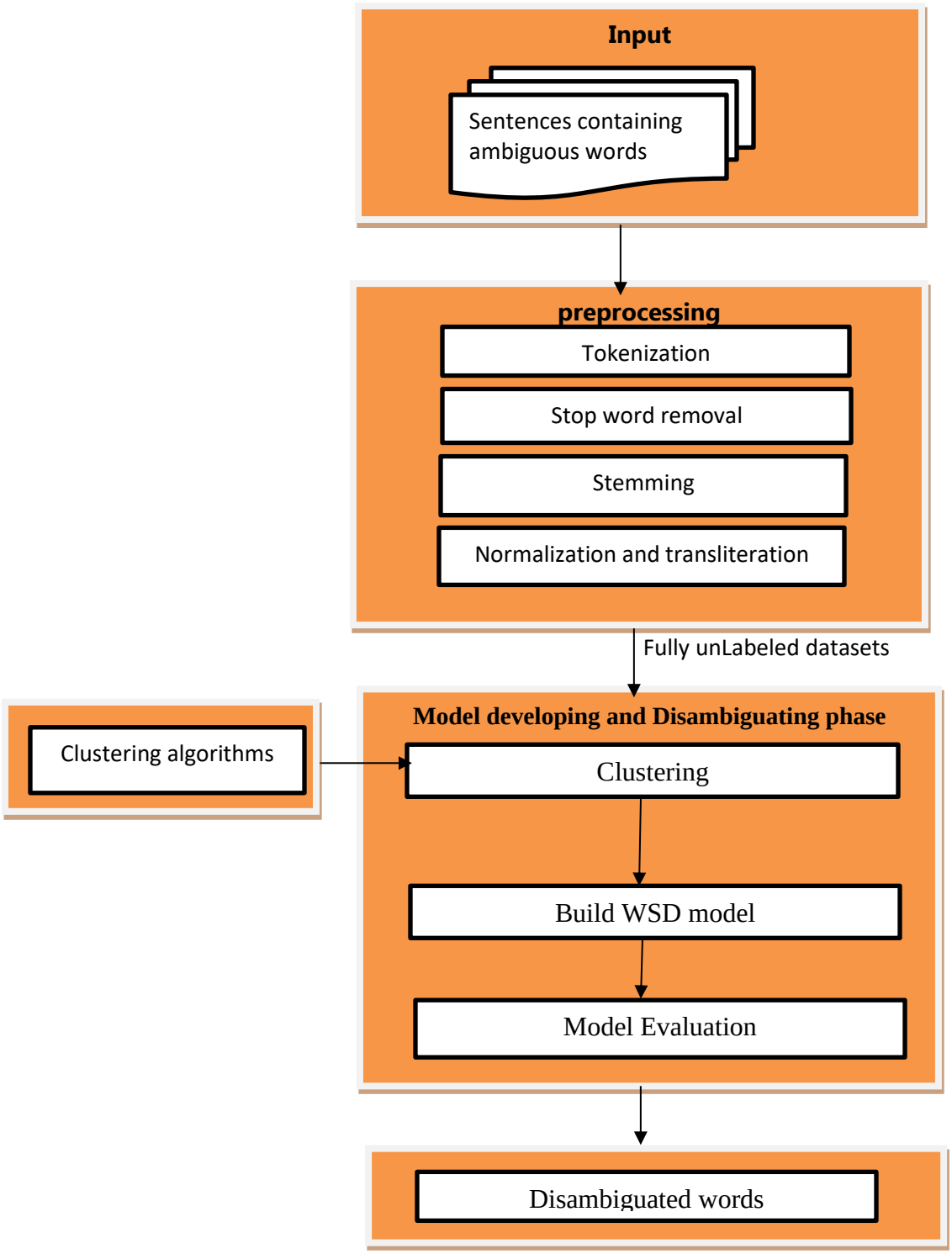


Figure 4.2: Proposed architecture of WSD for Tigrigna Language using unsupervised approach

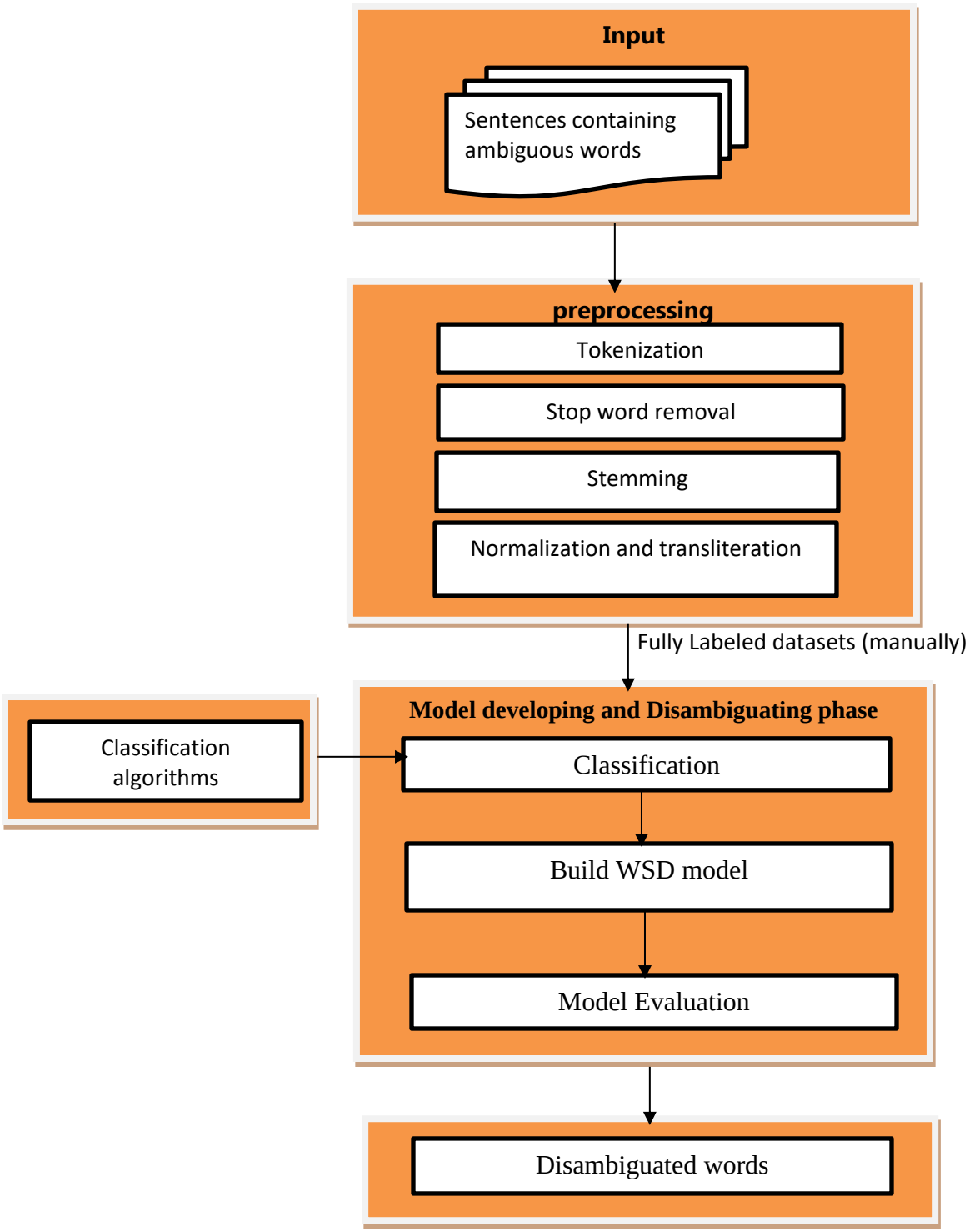


Figure 4.3: Proposed architecture of WSD for Tigrigna Language using supervised approach

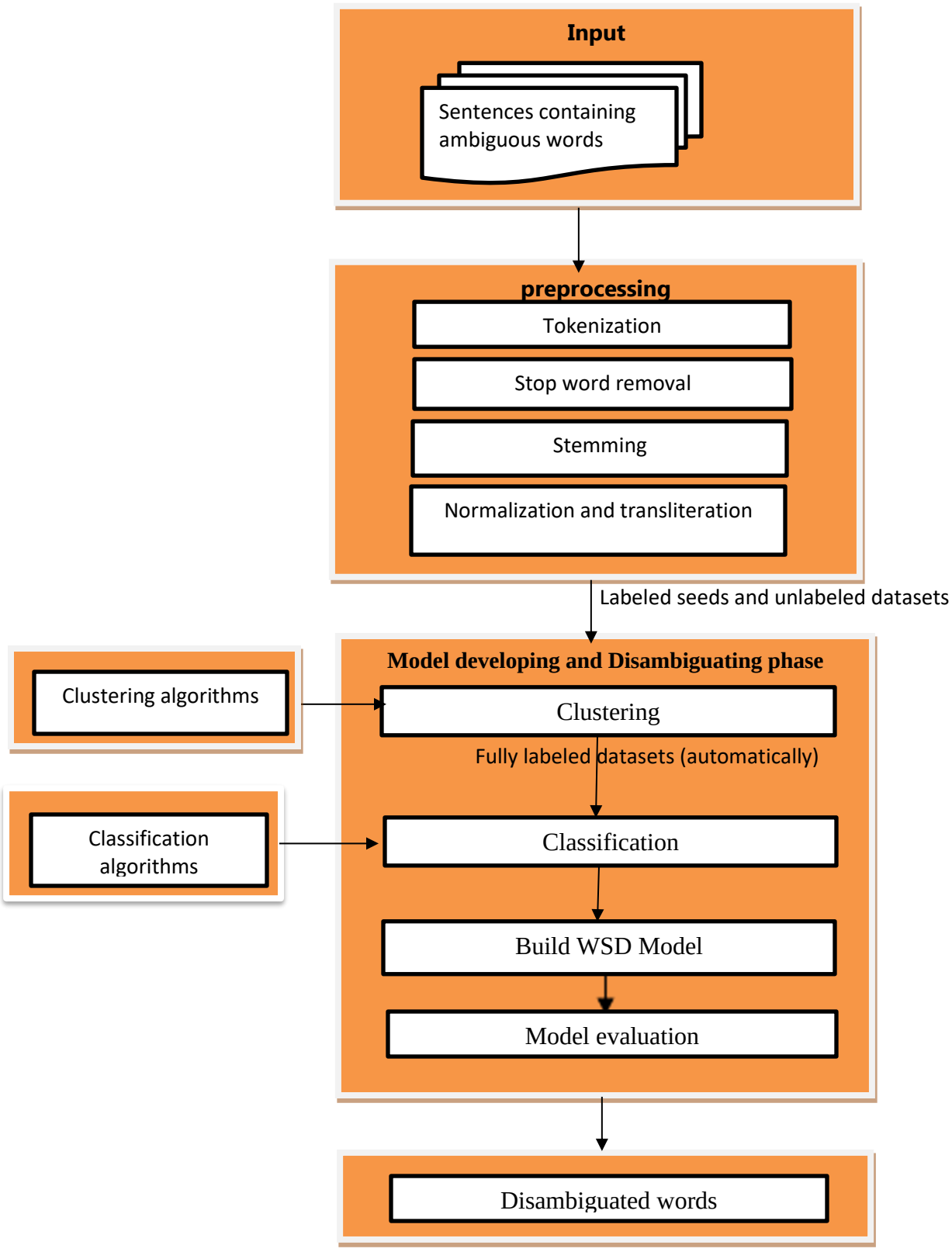


Figure 4.4: Proposed architecture of WSD for Tigrigna Language using semi-supervised approach

Document Preprocessing: Document preprocessing describes any type of processing performed on raw data to prepare it for another processing procedure. Hence, preprocessing is the preliminary step which transforms the data into a format that will be more easily and effectively processed [72]. Preprocessing must ensure that the source text be presented to NLP in a form usable for it. In this study, document preprocessing is a primary step to make our data sets compatible with the machine learning tool that was used in our study called Weka. In the preprocessing stage this study addresses tokenization, stemming, stop word removal, transliteration and normalization. All the processing tasks that we have used in this study were implemented using python software.

Tokenization: Tokenization is a module which splits up the text into a set of tokens usually words, based on the boundaries of a written text. Tokenization takes the input texts that is supplied from a user and tokenizes it in to a sequence of tokens. Token is the smallest unit that will be extracted from the input sentence before performing word sense disambiguation. All punctuation marks, numbers and special characters are removed from the text before the data is processed. All punctuation marks are converted to space and space is used as a word demarcation. Hence, if a sequence of characters is followed by space, that sequence is identified as a word. A consecutive sequence of valid characters was recognized as a word in the tokenization process. Tokenization is used to get context words for disambiguation purpose. For instance, if we have a sentence like: *-ተኽላይ እቲ ኣብ ጥቓ ገዝኡም ዝነበረ ሳዕሪ ነቒልዎ።* Here if we tokenize this sentence it looks like *ተኽላይ*, *እቲ*, *ኣብ*, *ጥቓ*, *ገዝኡም*, *ዝነበረ*, *ሳዕሪ*, *ነቒልዎ*, and ።

Stop word removal: Not all terms found in the document (or corpus) are equally important for disambiguation purpose. Some terms are common in most documents. Therefore, removing terms which are not used to determine the meaning of the ambiguous word is very important. Such terms may be removed using two methods. The first method is to remove high frequent terms by counting the number of occurrences (frequency). The second method is using stop word list of the language. Therefore, in this study the second method were used to apply stop word removal. Tigrigna domain independent stop words include prepositions, conjunctions, and articles. Those words need to be removed during preprocessing phase. Words like **‘ከም’, ‘ክንዲ’, ‘ኣብ’, ‘እዩ’, ‘ዘሎ’, ‘ናብ’, ‘ናይ’, ድማ**’, etc. are commonly appeared Tigrigna stop words. For the stop word removal process, the stop words are taken from the list identified by Yonas F. [20] and Gebrehiwot A. [15] who were

conducted a research on development of stemming algorithm for Tigrigna texts and A Two Step Approach for tigrigna text categorization respectively. But there are some stop word lists that are very important for disambiguation purpose in this research. Therefore, those terms are not removed from the corpus. Stop words like **ካብ, ካብዚ, ምኳኑ, ምኳን, ካብኡ** are not removed from the corpus that contains the ambiguous word **'ሓለፈ'** because they are used as context words which means they used to differentiate the meaning or sense of the ambiguous word. The algorithm that shows the removal of stop words is shown in the following Algorithm 1.

1. Open a file that contains stop word list
2. Read a sentence from the corpus
   For each word in the sentence
      If a word is in stop word list then
         Remove a word
      End if
   End for
3. Return the remaining words
4. Close file

Algorithm 1: Stop word removal algorithm

Stemming: Stemming is the process of reducing morphological variants of words into base or root form. In morphologically rich languages like Tigrigna, a stemmer will lead to significant improvements in WSD systems. In Tigrigna, there are different terms that are generated from the same root word due to their grammatical use. To create different derivational and inflectional word forms Tigrigna language makes use of prefixes, suffixes, and infixes. Therefore, those extra words or characters that change the root word to different forms are stemmed from the corpus using the stemming algorithm developed by Yonas F. [20]. This algorithm removes both prefixes and suffixes only without normalization. Therefore, to get the common form of the ambiguous word the researcher tried to normalize it. For Example, let's take the ambiguous word 'ዓረቐ' may become like ዓሪቐ, ዓራቂ, ዓሪቓ, ዓሪቐ, etc after removing both prefixes and suffixes. To make it suitable for machine learning algorithms, the researcher normalized all those words into one word which is

‘ዓረቐ. The same thing was applied for other ambiguous words that are used in this study but not for the context words. The reason behind not applying normalization after stemming for the context words is due time taken to normalize all the context words that are used in this study. **Normalization:** normalization is the process of normalizing alphabets that have the same meaning and pronunciation, but they have different representations of alphabets. In Tigrigna language, there are words that have different orthographic representation but the same phonemes and the same usage like “ፀ”, “ጸ”, “ሰ”, “ሠ”. Therefore, normalizing similar symbols with one symbol is an important task to avoid ambiguity i.e. means converting those characters into one symbol. For example, let’s take the word “ሠዓረ” and “ሰዓረ”, they have the same meaning (or usage) and pronunciation but different representation. In our corpus, this word might be written by using those two representations; but this word is one of the ambiguous words that the researcher selected in this study. Therefore, the word must be represented whether in the form of “ሠዓረ” or “ሰዓረ” but not both in the whole document. In Tigrigna writing system, there are also many abbreviations (short) words. These short words can be single or compound short forms of Tigrigna language words abbreviated by either slash (/) or period (.). In order to make the corpus usable for stemming process, the short words should be expanding to their expanded form. Therefore, normalization is also used to expand those short word in to their expanded form. For example, ቤት ት/ቲ when we expand, it becomes ቤት ትምህርቲ. Normalization algorithm is shown in the following Algorithm 3.

1. Open file and short word list
2. Normalize the document
   For each word in the sentence
   If a word is in short word list
   Expand the word
   End if
   End for
3. Return the sentence with expanded word
4. Close file

Algorithm 2: Normalization Algorithm

Transliteration: after the above preprocessing tasks are done for Tigrigna documents that we collect; transliteration was accomplished for Tigrigna documents. Transliteration is the representation of the characters of one language by corresponding characters of another language. In this paper, the transliteration is accomplished from Tigrigna characters into Latin characters. As Getahun W. [1] noted in his study; doing transliteration makes documents compatible with the machine learning tool called Weka. Since we selected a machine learning tool (called Weka) for conducting our experiment; we applied transliteration after preprocessing tasks. The transliteration of the Tigrigna corpus was conducted by using System for Ethiopic representation in ASCII (SERA) [73]. SERA, is case sensitive, i.e., upper and lower cases of the English alphabet representing different symbols in the Tigrigna language alphabet. Therefore, we can represent simply the Tigrigna alphabets to its corresponding Latin alphabets. In this study, all Tigrigna texts were transliterated into corresponding Latin scripts using a simple python programming language.

In this study, transliteration is only used to make our corpus Weka readable. This is the reason that makes us to use transliteration after preprocessing. But, performance might be improved if researchers transliterate their corpus to latin scripts before all the preprocessing tasks. Therefore, we could recommend other researchers to use transliteration before preprocessing tasks in oreder to investigate the performance of WSD.

4.4. Preparing Machine Readable Data

After preprocessing tasks and transliteration were applied for the corpus, preparing machine readable data became important for applying clustering and classifying activities of data mining techniques. First, we prepared our corpus in the form text files. But for experimentation of this study, we used a machine learning tool which is Weka as a development tool for the proposed WSD prototype model. Therefore, preparing corpora in the form of text file is not compatible (or readable) with Weka. To make it readable using Weka, datasets must be prepared in the form of CSV or ARFF file formats. Therefore, we prepared our datasets in the form of ARFF file formats. When we prepared our datasets, the target word (or ambiguous word) must be aligned in the middle column and context words were aligned to the right and left of the target word. Data having big window sizes were prepared for each ambiguous word in this study. But the researcher used a maximum of ten window sizes in both sides for building the main datasets. A maximum of ten window sizes were prepared by removing extra columns of each ambiguous word. A maximum

of ten window sizes means ten context words to the right and ten context words to the left of the target word. When we use ten words to the left and ten words to the right, there might be missing values. Those missing values were replaced by question marks, because question mark is compatible with Weka.

4.5. Preparing Datasets

In this section, how datasets are prepared for this study will be presented. In this study, we used semi-supervised approach. Since we used semi-supervised machine learning method for this study, we prepared a combination of labeled datasets and unlabeled datasets for training. Because, semi-supervised learning uses both labeled and unlabeled datasets for training.

In semi-supervised approach, annotation is very important for some of the datasets. Annotation is the process of tagging or labeling some seed training examples with the label of its corresponding class or sense [74]. For this study, ambiguous words "**Halefe**" and "**medeb**" have four and three senses respectively, and the rest of the ambiguous words have two senses of each. Each sense of seed examples was labeled manually. As Getahun W. [1] noted in his study, more number of sentences need not be annotated manually in semi-supervised machine learning approach. As [75] explained, the representative labeled and unlabeled data size distribution for training set is typically 85-98% unlabeled datasets; and the rest are for labeled datasets. By considering this, we prepared 12% of labeled datasets and 88% of unlabeled datasets for each of the five chosen ambiguous words from the total datasets before clustering. When we labeled seed examples manually, we applied some techniques. Those techniques will be described in the next section. Unlabeled datasets will become fully labeled datasets by using automatic clustering technique. Out of the total datasets for each of the five chosen ambiguous words 10% examples was randomly selected to be the test set where as 90 % of dataset was training set. Some of the datasets that were given classes for the word 'kefele' is shown in table 4.2.

Table 4.2: Example of WSD Dataset for Semi-Supervised Learning

LC2	LC1	Target word	RC1	RC2	class
zeri	haymanot	kefele	delyu	seb	divide
zeri	haymanot	kefele	halyu	Habar	divide
zeri	haymanot	kefele	adkiKa	adenquirka	divide
Oaleba	ityoPya	kefele	Habiru	?	?
wereda	Tabya	kefele	geliSo	?	?
Oda	lmdo	kefele	gebru	Habiro	?
meseret	ruba	kefele	sgaro	Kewn	?
Tyt	waga	kefele	gebr	neyru	pay
rezin	waga	kefele	sdra	bEtu	pay
kebid	waga	kefele	seb	mesel	pay
xaObiya	kefele	kefele	fenSigu	wxTi	?
Hares	kaHsa	kefele	mfSamu	Tequmu	?
gbaI	gbri	kefele	quSSr	polis	?
tgray	hzbi	kefele	meswaIti	jebha	?

We prepared our datasets with 22 attributes which includes the target word and class. The rest 20 attributes are context words that are used to determine the meaning of the ambiguous word which means 10 words to the left and 10 words to the right of the target word. Attributes that exceeds from this size was removed because the researcher was focused on 10-10 window sizes. Reducing dimensionality of datasets can improve the performance of WSD; because instances having redundancy and missing values problems will be reduced [1].

4.6. Seed Selection Techniques

In this section the techniques that were applied in this study will be presented. This research was conducted by using semi-supervised approach. Therefore, both labeled and unlabeled documents must be available in this study because conducting a research using semi-supervised approach requires both labeled and unlabeled training datasets. To do this, the researcher used important techniques. Those techniques were taken from the research that was conducted by Getahun W. [1]. Techniques that were used in this study consist of four steps. Those steps are: -

- Selecting representative seed examples for each class or sense of the ambiguous words.
- Clustering both labeled and unlabeled seed examples using “classes to clusters evaluation” mode.
- Perform feature selection on fully labeled dataset and feature extraction.

- Build the final classifier.

Step 1: - Selecting representative seed examples for each class or sense of the ambiguous words

Selecting representative seed examples for each class is effective; because this study was used semi-supervised machine learning approach. Those selected seed words are used to label unlabeled documents. As [45] described in his study, selecting seed words to select representative seed examples for semi-supervised approach is challenging task. That means if we select improper seed words, the selected seed examples would be improper. Then the result of selecting improper seed examples will be poor in performance. Improper seed examples can be selected when we tagged (labeled) our datasets randomly by humans. When humans select seed examples randomly, they may select improper seed words which means the selected seed words cannot differentiate the senses (meanings) of the ambiguous word.

To minimize manual selection of limited number of seed examples from the total datasets, we used tree algorithms because tree algorithms represent the related concept of the target word starting from the node. As Getahun W. [1] discussed, tree algorithms used information gain as lexical knowledge and information gain can minimize subjectivity problem in manual selection of seed examples. By considering this concept, we used ADTree algorithm to select seed words which are used to select seed examples from our datasets. We classified our datasets by using ADTree algorithm; then tree visualization would be taking place. After tree visualization the researcher took seed words scoring high information gain in the tree structure. Those seed words were used for discrimination purpose of each sense of the ambiguous word. After all those processes were taking place, the researcher was selected instances which contain those seed words and assigned their senses depending on the selected seed words. [76] explained, instances which contain seed words scoring high information would have high discriminating power for sense classification purpose. For example, 'kefele' is one of the ambiguous words that we used in our study; the lexical knowledge or heuristic information for this ambiguous word are 'divide' and 'pay'. Seed words scoring high information gain for expressing sense of 'pay' are: '*meswaIti', kaHsa', 'gbri', 'genzeb','waga*' and for sense of 'divide' are: - '*maOr*e', '*OCa','hafti'*. But selecting seed words by using ADTree algorithm was a challenging task for two of the ambiguous words which were used in this study (which means for '*medeb', 'Halefe'*). Because, ADTree algorithm is used for words having two senses only but those two words have three and four senses respectively.

Therefore, to select seed words for those two ambiguous words by using ADTree algorithm we used a combination method of their senses in pair. For example, the ambiguous word "**medeb**" have three senses which are 'traditional bed', 'plan', and grouping. Therefore, the researcher grouped the datasets in to three groups by using a combination method. Datasets for the senses of "traditional bed" and "plan" in one group, datasets for the senses of "traditional bed" and "grouping" in one group, and datasets for the senses of "plan" and "grouping" in one group. Then after, seed words having high information gain in all of the groups were selected. For instance, the word "medeqesi" has high information gain for the sense of "traditional bed" in two of the groups which means groups of "traditional bed" and "grouping", and groups of "traditional bed" and "plan". The word "zala" has also high information gain in two of the groups which means groups of "traditional bed" and "grouping", and groups of "plan" and "grouping" for the sense of "traditional bed". The word "afeSaSma" has also high information gain in two of the groups which means groups of "plan" and "traditional bed", and groups of "plan" and "grouping" for the sense of "plan". The same action was taking place to select words for each senses of the ambiguous word "Halefe". Therefore, the representative seed examples for each senses of the selected ambiguous words were selected by considering seed words having high information gain in both of the groups. But in order to find the accuracy of the classifier using ADTree algorithm for those two ambiguous; we took the average accuracy of all the groups. For instance, to find the accuracy of the classifier for the ambiguous word "**medeb**" using ADTree algorithm; first the researcher was finding the accuracy for each of the group in each window size which means the accuracy of "medeb" for finding of the senses "traditional bed" and "plan", "traditional bed" and "grouping" in another group, "grouping" and "plan" in each window size (which means starting from 1-1 to 10-10). Lastly, we took the average accuracy of those groups for determining the accuracy of the classifier for the ambiguous word "medeb" using ADTree algorithm. The same action was taking place to find the accuracy of the classifier for the ambiguous word "Halefe".

But, we also selected seed words manually for selecting representative seed examples of Tigrigna datasets. This kind of selection was used for comparing the results obtained using manual selection of seed words with the results obtained using seed words selected using ADTree algorithm. This kind of comparison was taking place in the second experiment of our study. In order to select representative seed examples using seed words manually, we were used two methods of seed

words selection. As [1] explained in his study, Seed words can be selected by using the following methods (strategies):

- Extract seed words from a dictionary's entry for the target sense. Words in the entry appearing in the most reliable collocation relationships with the target word are given the most weight.
- Use a single defining collocate for each class or sense: by identifying a single defining collocate for each class may yields remarkably good performance using for seeds only those contexts containing one of these words. For example, '*medeqesi'* for 'traditional bed' sense, '*zala'* for sense 'grouping' sense, and 'qesali' for sense 'plan' for the word "*medeb*".
- Label salient corpus collocates: Label salient corpus collocates are Words that appear significantly more often in the context with the target word, especially in certain collocational relationships, will tend to be reliable indicators of one of the target word's sense. For example, words like '*medeqesi', 'gdm, 'kef', 'igri', etc*were seed words used for 'traditional bed' sense of the ambiguous word '*medeb'.*

From those seed word selection strategies, we used the second and the third strategies for this study which means use a single defining collocate for each class and label salient corpus collocates.

Step 2: Clustering both labeled and unlabeled seed examples using "classes to clusters evaluation" mode:

Clustering is unsupervised learning technique that clusters similar items together. This step finds inherent clusters in the unlabeled data. For clustering purpose, we used four clustering algorithms which are available in Weka. The clustering algorithms used in our study are: Expected Maximization (EM), SimpleKMeans, Farthest First, HierarchicalClusterer. We select those clustering algorithms because they have better efficiency than the other clustering algorithms for our datasets that we prepared for this study.

Here, we could not use the resulting clusters for classification. We only use them to identify the cluster of missing instances based on Labeled seed examples. After clustering have been done using EM algorithm, we can see the effect of semi-supervised learning method in our work. The clustering result may show that the class labels of some seed examples were misclassified. However, automatic clustering suggests such label changes, the change was not required because those seeds were labeled with their sense class as the promise they are chosen by experts

intentionally. The researcher conducted Clustering algorithms on five ambiguous words of the language and the result was evaluated using "classes to clusters evaluation" mode in a machine learning tool Weka 3.8. We used "classes to clusters evaluation" mode to see the performance of clustering algorithms.

Step 3: Feature Selection and feature extraction

Feature Selection: The success of machine learning requires instances to be represented using an effective set of features that are correlated with the categories of word senses. For this study, Feature selection was performed by preparing a ten-ten window size. Instances with missing values were also removed from the feature sets. Therefore, feature selection is a data reduction mechanism.

Feature Extraction: feature vector which represents words for each instance of a target word, i.e. files of comma-separated values, a line in WEKA with an extension of. arff or .csv. These vectors should represent a text window surrounding ambiguous word of a ten-ten words in our work case.

Step 4: build the classifier

Before classifying our datasets using the selected classification algorithms, we labeled our datasets manually depending on the selected seed words. Because manually labeled seed examples being used as cluster labels during clustering on both labeled and unlabeled documents (datasets). Knowing cluster label of each instance become important for differentiating the class of missed instances by taking each cluster as a distinctive class. This helps us to label unlabeled instances with their classes.

Classification of possible senses of the datasets may vary depending on the classification algorithms. The final classifier was build using five classifying algorithms. Applying different classifying algorithms is very important to identify which classification algorithm is highly effective for Tigrigna WSD models. To classify our datasets, we were selected five classifying algorithm which are available in in a machine learning tool Weka 3.8. Namely those algorithms are: Naïve Bayes, SMO, AdaBoostM1, Bagging, and ADTree. We tried to choose these supervised algorithms representing a few different algorithmic approaches to semi-supervised learning. From the selected algorithms, ADTree, AdaBoostM1 and bagging represent bootstrapping algorithms. Naïve Bayes and SMO algorithms are representatives of probabilistic and support vector machines respectively.

4.7. Evaluation Techniques

Since we used semi-supervised approach, two evaluation measures have been applied in this study. The first task was evaluating the performance of clustering algorithms and the second task was evaluating the performance of classification algorithms. Evaluating the performance of clustering algorithms help to continues to the next classification task. After evaluating the clustering performance in terms of accuracy, then evaluation of supervised learning was followed.

To evaluate performance of clustering algorithms, four different cluster modes are available in WEKA. Those are using training sets, supplied test set, percentage split, classes to clusters evaluation mode. For this study the researcher used 'classes to clusters evaluation' mode. When we use 'classes to cluster evaluation' mode, Weka package assigns classes to the clusters based on the majority value to the class attribute within each cluster in test phase. WEKA shows the clustering result as error rate using 'classes to clusters evaluation' mode. Therefore, accuracy of clustering algorithms was obtained after subtracting the error rate from hundred. This accuracy would be used to measure how well it has been able to generalize the clustering results. When we apply the selected clustering algorithms for our datasets, we have to limit the number of clusters. Limiting the number of clusters is used to equalize with the number of senses of the ambiguous word. Doing that helps to know which instance is grouped under which cluster. This again helps to prepare completely labeled dataset for classification by using clustering assumption.

In this study, we evaluated the hypothesis of the proposed prototype model by conducting experiments using relatively small-sized labeled data with large-sized unlabeled data (now labeled after clustering), in other words, after clustering, the cluster of the unlabeled instances is known even though their label becomes "missing value". Therefore, knowing clusters of the unlabeled data becomes easy for preparation of all labeled data for the final classification task. After evaluating clustering performances using 'classes to clusters evaluation' mode, evaluation of classifying algorithms was applied. For evaluating performance of classification algorithms, we used the cross-validation test option which is available in Weka. As most experiments employ random choice strategy, 10-fold cross validation has also been used for this study in that it iteratively experiments after dividing the dataset into ten mutually exclusive folds then it averages the results obtained in each trial. The number of folds has been used as ten in that it is a default in Weka. Hence, during the experiment, the dataset is initially divided into ten parts (folds) then

averaging the results will be done after holding out each part in turn. In 10-fold cross validation, the procedure of training and testing repeated 10 times so that each partition or fold is used once for testing and nine times for training.

The performance of classification algorithms is usually measured by parameters such as accuracy, recall, precision and F-measure. These performance parameters are the functions of the numbers of correctly and incorrectly classified instances which are obtained on the confusion matrix of WEKA output. Confusion matrix analyzes how well an algorithm classifies instances in their corresponding classes. For instance, lets discuss a two Class confusion matrix with A and B classes in the following table 4.3.

Table 4.3: Confusion matrix for two classes (A and B)

Classified as	A	B
A	True positive	False negative
B	False positive	True negative

The terms listed in Table 4.3 will be discussed below:

True positive refers to actual positive instances which are correctly labeled and True negative refers to actual negative instances which are correctly labeled. False positive refers to negative instances that are incorrectly labeled and false positives refer to negative instances that are incorrectly labeled.

Precision: precision in WSD is percentage of words that are labeled correctly, out of the words addressed by the system.

$$\text{Precision} = TP / (TP + FP)$$

Recall: recall in WSD is percentage of words that are tagged correctly, out of all words in the test set.

$$\text{Recall} = TP / (TP + FN)$$

Accuracy: accuracy in WSD is percentage of correctly classified instances.

$$\text{Accuracy} = (TP + TN) / (TP + TN + FN + FP)$$

Predictive accuracy has often been used as the main and only evaluation criterion for the predictive performance of classification or data mining algorithms [77]. By considering this idea, we used Accuracy for measuring the performance of the proposed WSD prototype model of this study.

Chapter Five

5. Experimentation and Discussion

As we have discussed in the above, semi-supervised machine learning was selected for this work. Because of this reason both labeled and unlabeled datasets were used by the researcher. Here both clustering and classification tasks were applied by using clustering and classification algorithms respectively. Clustering algorithms are used for labelling both labeled and unlabeled datasets in order to know the cluster of missed datasets. In this study four experiments were conducted by using five classification algorithms and four clustering algorithms. We used Naive Bayes, SMO, AdaBoostM1, Bagging algorithms for classification purpose and EM, SimpleKMeans, Farthest First, HierarchicalClustererfor clustering purpose of our datasets. Those clustering and classification algorithms was applied for the selected five ambiguous words of Tigrigna language. Those five selected ambiguous words of Tigrigna language for this study are: ከፈለ (kefele), ዓረቐ (OareQe), ሰዓረ (seOare), ሓለፈ (Halefe), መደብ (medeb) with a total of 1250 sentences. By using those algorithms, ambiguous words and datasets; the following experiments were conducted in this study:

1. Comparison of supervised, unsupervised and semi-supervised learning methods on Tigrigna datasets.
2. Comparison of results obtained when we used, seed words selected by using ADTree algorithm and seed words selected by humans manually.
3. Investigating the best performing classification algorithms for Tigrigna datasets.
4. Investigating the effect of different context sizes on disambiguation accuracy for Tigrigna ambiguous words, and to find out the optimal window size.

5.1. Discussion of Experimental Results

Experiment I: Comparison of supervised, unsupervised and semi-supervised learning methods on Tigrigna word sets

To compare results of unsupervised, supervised, and semi-supervised machine learning approaches; we used the same datasets of the language, and the classification algorithms for both semi-supervised and supervised approaches but for unsupervised approach we used the above selected clustering algorithms. We used clustering algorithms for unsupervised machine learning approach because clustering is unsupervised technique. Therefore, comparison of those three machine learning approaches was conducted on the same datasets. But comparison of semi-supervised and supervised approaches was conducted on the same datasets by using the same classification algorithms.

Unsupervised machine learning method

Unsupervised learning method requires unlabeled datasets which means it does not need experts to labelled the datasets. Because of that it always performs low performance when we compare to other learning methods which means supervised and semi-supervised learning methods. Unsupervised machine learning method has lack of annotated datasets. Having lack of unlabeled datasets make unsupervised machine learning method to have low performance. As we have discussed in 4.7 to select the best performing clustering algorithms for Tigrigna word sense disambiguation, we conducted our clustering tasks using four clustering algorithms which are: - EM, Farthest First, HierarchicalClusterer, SimpleKMeans. Lastly, we found that EM algorithm performs better when we compared with the other selected clustering algorithms in average. Therefore, for labeling the unlabeled datasets we used EM algorithm. Even though, HierarchicalClusterer algorithm was achieved better results for some of the ambiguous words (like kefele, seOare, medeb); we used EM algorithm in order to cluster our datasets because EM algorithm performs better in average accuracy. The results are shown in the following table 5.1.

Table 5.1: Results using unsupervised machine learning

Training datasets	EM	Farthest First	HierarchicalClusterer	SimpleKMeans
kefele	57.6531%	53.5714%	61.7347%	52.0408%
OareQe	58.75%	56.25%	50%	51.25%
seOare	74.0741%	69.7531%	85.1852%	62.3457%
Halefe	44.1558%	43.3766%	31.6883%	45.1948%
medeb	43.0328%	52.459%	48.3607%	46.3115%
Average	**55.5332%**	**55.082%**	**55.3938%**	**51.4286%**

The performance achieved by using the selected four clustering algorithms was in the range of 51.4286%-55.5332% accuracy. The results shown in the above table are the optimal results for each ambiguous word using the selected clustering algorithms. The performance that was achieved by using the selected clustering algorithms may be low because occurrences of common words on the selected datasets may be high. Occurring more common words on the selected datasets may lead to misclassification of their senses. Specially, ambiguous words like 'medeb' and 'Halefe' results low performance using four of the selected clustering algorithms when we compare with the rest of the selected ambiguous words. Because, those two ambiguous words have three and four senses respectively. This means those clustering algorithms cluster the selected datasets of those two ambiguous words in to three and four clusters respectively. When the clustering algorithms cluster those two ambiguous words depending on the surrounding words to each cluster; more common words which are difficult to differentiate their cluster may occur. Therefore, this difficulty results low performance for clustering algorithms. Generally speaking, EM algorithm results better in average accuracy (i.e. **55.5332%)** for our datasets when we compared with the other three clustering algorithms. Still the average accuracy that we obtained is not attractive and really applicable for Tigrigna WSD prototype model. Therefore, to improve the performance score of clustering algorithms other experiment is important by using other machine learning methods. Then we conducted our experiment using supervised and semi-supervised machine learning methods as follows.

Supervised machine learning method

Supervised learning requires annotated training data from which a classifier is induced by the algorithm. It is relatively easier task compared to unsupervised learning due to that, it is a labeled training set composition. However, most of the time, to achieve good performance, the amount of

training data required by supervised learning is quite large. This is undesirable as hand-labeled training data is expensive and only available for a small set of words [7].The result is shown in the following table 5.2.

Table 5.2: Results using supervised machine learning

Training datasets	ADTree	AdaBoostM1	Bagging	SMO	Naive Bayes
Kefele	74.4898%	60.7143%	77.0408%	83.6735%	84.1837%
OareQe	64.375%	66.25%	66.875%	72.5%	72.5%
seOare	87.037%	78.3951%	80.2469%	87.6543%	88.2716%
Halefe	83.8984%	43.8961%	62.5974%	75.5844%	69.6721%
Medeb	79.9786%	47.9508%	67.623%	70.082%	77.9221%
Average	**77.9558%**	**59.4413%**	**70.8766%**	**77.8988%**	**78.5099%**

The results shown in the above table 5.2 are the performances achieved using supervised machine learning methods. Here, the researcher gave fully labeled datasets to the machine learning tool which is Weka for experimentation purpose. After that, the results shown in the above was achieved. We achieved the best result using Naive Bayes algorithms when we compare to the other classification algorithms which are ADTree, AdaBoostM1, bagging and SMO for our datasets. The performance achieved using Naïve Bayes algorithm was 78.5099% in average. But the performance achieved using ADTree, AdaBostM1, Bagging and SMO were 77.9558%, 59.4413%, 70.8766%, 77.8988% respectively in average. Here, both ADTree and SMO algorithms perform comparable results to Naive Bayes algorithm i.e. 77.9558% and 77.8988% respectively. But AdaBoostM1 performs the lowest result when we compared to the other selected classification algorithms. From this we can conclude that, Naive Bayes algorithm performs better on fully labeled datasets of Tigrigna ambiguous words (or using supervised learning methods).

Naïve Bayes algorithm takes each instances value under consideration whereas bootstrapping algorithms boot from seed instances without considering all dataset instances. Naïve Bayes algorithm has a probabilistic behavior rather than Seed example selection behavior; whereas, bootstrapping algorithms have seed example selection behavior rather than probabilistic behavior. Therefore, this might be the reason why Naïve Bayes algorithm performs better whereas AdaBoostM1 and bagging algorithms perform less on fully labeled datasets of Tigrigna ambiguous words. This means supervised learning method makes bootstrapping algorithms discomfort on their performances.

Semi-supervised machine learning

Semi-supervised learning method requires only a small amount of labeled training data and is sometimes able to improve the performance using unlabeled data. Word sense disambiguation is an ideal task for effective semi-supervised learning methods as unlabeled data is easily available and labeling a large enough corpus for supervised learning of all words has so far been too expensive to carry out for the natural language processing community [74]. The basic idea of semi-supervised learning is to automatically label the unlabeled examples using a small number of human labeled examples as seeds. By doing this, semi-supervised learning yields a large labeled dataset that can be used as training data for a normal supervised learning algorithm [7].The experiment of semi-supervised methods using the selected algorithms was conducted on the five Tigrigna WSD datasets following semi-supervised clustering assumption. The result is shown in the following table 5.3.

Table 5.3: Results using semi-supervised machine learning method

Training datasets	ADTree	AdaBoostM1	Bagging	SMO	Naive Bayes
Kefele	90.82%	89.80%	89.80%	89.80%	29.08%
OareQe	81.88%	81.25%	83.13%	80.63%	46.25%
seOare	89.51%	88.27%	88.27%	88.27%	48.77%
Halefe	90.92%	88.83%	88.31%	88.57%	37.92%
Medeb	94.27%	89.34%	56.15%	89.75%	25.82%
Average	**89.48%**	**87.50%**	**81.13%**	**87.40%**	**37.57%**

As it is shown in table 5.3 semi-supervised machine learning method using bootstrapping algorithms result better performance when we compared with unsupervised and supervised machine learning methods. It results really applicable and attractive results using most of the classification algorithms specially using bootstrapping algorithms. Most of the selected classification algorithm improves their accuracy when we compared to results achieved in table 5.2. For example, the average accuracy for ADTree, AdaBoostM1, Bagging, SMO, and Naive Bayes were 77.9558%, 59.4413%, 70.8766%, 77.8988%, and 78.8766% respectively using supervised learning method. But using semi-supervised learning method, the researcher was achieved89.48%, 87.50%, 81.13%, and 87.40% for ADTree, AdaBoostM1, Bagging, and SMO algorithms respectively. But the average accuracy obtained by Naïve Bayes algorithm using semi-

supervised machine learning method is much lower than the average accuracy obtained by Naive Bayes algorithm using supervised machine learning method due to the unbalance distribution of sense classes that causes for data overfitting problem. By using semi-supervised learning method, Getahun w. [1] was obtained an accuracy of 88.47% for Amharic ambiguous words. For Tigrigna ambiguous words, we obtained an average accuracy of 89.48%using ADTree algorithm. We achieved the best performance of the classifier for Tigrigna ambiguous words using ADTree algorithm because seed words which obtained high information gain using ADTree algorithm were used for the selection of representative seed examples of this study.

SMO is a type of support vector machine which results high performance using semi-supervised machine learning for Tigrigna ambiguous words. ADTree algorithm is a bootstrapping algorithm which results an average accuracy of 89.48% which is also attractive result. Bootstrapping algorithms like Bagging and AdaBoostM1 also achieved an average accuracy of 81.13% and 87.50% respectively. This is also an attractive result when we compared to results obtained using supervised machine learning method.

For comparison purpose, let's take the maximum average performance of the three machine learning methods by using the best performing algorithms. The result is shown in table 5.4

Table 5.4: Average performance of the three machine learning methods

Machine learning method	The maximum average accuracy	Best performing algorithm
Unsupervised	55.5332%	EM
Supervised	78.5099%	Naive Bayes
Semi-supervised	89.48%	ADTree

From this table we could understand that, semi-supervised machine learning improves the accuracy of WSD prototype model. By using both labeled and unlabeled datasets, the performance of WSD prototype model was improved. This is because unlabeled datasets are clustered using manually labeled datasets during clustering. From this we can conclude that semi-supervised machine learning is the most important machine learning methods for the development of Tigrigna WSD prototype model than supervised and unsupervised machine learning methods using bootstrapping and support vector machine algorithms which means using ADTree, AdaBoostM1, bagging, and SMO algorithm.

Experiment II: Comparison of results obtained when we use seed words selected by using ADTree algorithm and seed words selected by humans manually

To do this experiment we used two techniques. The first technique was selecting seed words using ADTree algorithm and the second technique was selecting seed words manually. Then depending on the selected seed words, we were selecting representative seed examples. But to select seed words using ADTree first we have to have fully labeled datasets. Those datasets were labeled manually. Then classify those datasets using ADTree algorithm which is available in a machine learning tool called Weka. After that select seed words that have high information gain. Those seed words were used for selecting representative seed examples from the whole datasets. But, if we do not have fully labeled datasets we can use the second technique which is selecting seed words manually by humans. Getting fully labeled datasets become difficult when we use large datasets. To remove this difficulty, selecting seed words manually by humans can be important. Since, our datasets were small in size we might have fully labeled datasets. Therefore, we could select seed words using ADTree algorithm. But in this study, seed words were selecting manually by humans in order to compare with the results obtained using seed words selected by using ADTree algorithm not because of the reasons that we explained in the above. For comparison purpose of results obtained using seed words selecting by humans manually and results obtained using seed words selected by using ADTree algorithm, we took the optimum accuracy using each algorithm for each ambiguous word. The results shown in the following Table 5.5 and Table 5.6 shows results obtained using seed words selected by humans manually and results obtained using seed words selected by using ADTree algorithm respectively.

Table 5.5: Results obtained using seed words selected manually

Training datasets	ADTree	AdaBoostM1	Bagging	SMO	Naïve Bayes
kefele	91.3265	90.8163	86.2245	88.2653	75
OareQe	90	90.625	90.625	90	86.875
seOare	91.358	88.8889	83.9506	91.9753	77.1605
Halefe	81.2745	88.8312	89.6104	89.3506	61.5584
medeb	92.2715	91.8033	89.7541	89.7541	56.5549
Average	**89.2461**	**90.1929**	**88.0329**	**89.8691**	**71.4298**

As shown in the above Table 5.5, the average accuracy for ADTree, AdaBoostM1, Bagging, SMO, and Naïve Bayes algorithms were 89.2461%, 90.1929%, 88.0329%, 89.8691%, and 71.4298%

respectively. From this we could understand that, AdaBoostM1 performs better when we compared to the other selected classification algorithms by achieving an average accuracy of 90.1929%. ADTree and SMO algorithms were also achieved comparable results to AdaBoostM1 algorithm by achieving an average accuracy of 89.2461% and 89.8691% respectively. The performance that was achieved using Naïve Bayes algorithm became the lowest result compared to the other selected classification algorithms. Generally speaking, results achieved using seed words selected by humans performs attractive result. The reason that makes us to achieve attractive result when we select our seed words manually might be the methods of seed words selection. Because, we used two strategies in order to select seed words manually. Those strategies were discussed in step 2 of 4.7.

Table 5.6:Results obtained using seed words selected by using ADTree algorithm

Training datasets	ADTree	AdaBoostM1	Bagging	SMO	Naïve Bayes
kefele	91.8367	92.3469	89.7959	93.8776	77.0408
OareQe	95	86.875	83.125	90.625	50
seOare	90.1235	90.1235	88.2716	91.9753	48.1481
Halefe	93.681	85.7143	88.3117	88.0519	88.3117
medeb	97.6769	90.5738	56.1475	95.082	90.9836
Average	**93.6636**	**89.1267**	**81.1303**	**91.9224**	**70.8968**

As it is shown in above Table 5.6, ADTree algorithms performs better when we compared to the other selected classification algorithms by achieving an average performance of 93.6636%. AdaBoostM1 and SMO algorithms were also performed an attractive result. Naïve Bayes algorithms was also performed the lowest result compared to the other selected classification algorithms by achieving an average accuracy of 70.8968%. The detailed description about Table 5.6 were discussed in the following experiment IV for investigating the best performing window size which to find the optimum performance for the selected Tigrigna datasets.

But, the main purpose of this experiment was to compare results obtained using seed words selected by humans manually and results obtained using seed words selected by using ADTree algorithm. Therefore, let's compare the results obtained in Table 5.5 and results obtained in table 5.6 using each algorithm. The results achieved in table 5.5 were smaller than the results achieved in table 5.6 using ADTree algorithm and SMO algorithms. whereas, the results obtained in table 5.5 were greater than the results obtained in table 5.6 using AdaBoostM1, Bagging, and Naïve

Bayes algorithms. In Table 5.5, the results achieved using ADTree and SMO algorithms were 89.2461% and 89.8691% respectively; whereas, the results achieved in Table 5.6 using ADTreee and SMO algorithms were 93.6636% and 91.9224% respectively. Therefore, 93.6636% is greater than 89.246%, and 91.9224 is also greater than 89.8691%.

Generally speaking, performances achieved using the selected classification algorithms by selecting seed words manually or using ADTree algorithm became comparable for the selected Tigrigna datasets. However, the performances achieved using the selected clustering algorithms became different which means better performances were achieved by selecting seed words using ADTree algorithm than selecting seed words manually. Therefore, we used seed words selected using ADTree algorithm to do our next experiment.

Experiment III: Investigating the best performing classification algorithm for Tigrigna datasets

For investigating the best performing classification algorithm for Tigrigna WSD prototype model, we applied five classification algorithms namely ADTree, AdaBoostM1, Bagging, SMO, Naive Bayes Algorithms for the selected five ambiguous words of Tigrigna language. AdaboostM1, ADtree and Bagging represent bootstrapping algorithms. ADTree is also a decision tree family which embeds adaboostic behaviors. Naïve Bayes algorithms is representative of probabilistic algorithms. Sequential Minimal Optimization (SMO) is representative of support vector machines family. As we have discussed in the above, those five classification algorithms were used in both supervised and semi-supervised machine learning methods. But for investigating the best performing classification algorithm, we used the result achieved by using semi-supervised machine learning methods because the performances achieved by using semi-supervised machine learning method were the most preferable when we compared with the performances achieved using supervised machine learning method. For comparison purpose of those selected classification algorithms, we took average accuracy obtained in the above using semi-supervised machine learning method. The comparison of those five classifying algorithms was based on the achieved performance and running time taken to classify ambiguous words of Tigrigna language. Those selected classification algorithms were taking place on the same Tigrigna datasets. The result is shown in the following table 5.5.

Table 5.7: Average accuracy of each classification algorithm with their time taken

Algorithm	Performance	Running time in seconds
ADTree	89.48%	0.008
AdaBoostM1	87.50%	0.01
Bagging	81.13%	0.076
SMO	87.40%	0.268
Naive Bayes	37.57%	0

From the above table 5.7, we could understand that Naïve Bayes algorithm was the lowest and fastest algorithm compared to the other selected classification algorithms. Naïve Bayes algorithm achieved an average accuracy of 37.57% and the average running time taken to classify the ambiguous word was zero seconds. From this we concluded that, Naïve Bayes algorithm was the lowest performer algorithm for WSD prototype model of Tigrigna language using semi-supervised learning method. The reason that makes Naïve Bayes algorithm performed less on our task might be the consideration of each values of seed instances rather than the selected representative seed examples. Unbalanced data distribution might be created during clustering of both labeled and unlabeled datasets of the selected ambiguous words of Tigrigna language using clustering algorithms. This kind of data distribution might also the reason why Naïve Bayes algorithm achieved less result using semi-supervised learning algorithms. Because, unequal numbers of sense class for the dataset leads to overfitting problem that degrade performance of the algorithm.

SMO provides good generalization performance on a wide variety of problems such as handwritten character recognition, face detection, pedestrian detection, and text categorization, etc. [78]. Since, it can apply for different real-world applications, we tried to use for classification purpose of our datasets in WSD prototype model of Tigrigna language. Therefore, the average accuracy achieved using SMO was 87.40% when we used semi-supervised machine learning method which was the third well performed algorithm next to ADTree and AdaBoostM1 algorithms respectively. This result may be applicable for WSD prototype model of ambiguous words of the language. Even though SMO algorithm performs well, the running time taken to classify our datasets was high when we compared to the other selected classification algorithms which means the average running time was 0.268 seconds. As Getahun W. [1] discussed in his study, sense classification was based on seed training examples from each sense of the target word instead of randomization of training examples. Therefore, this is the reason that makes SMO algorithm performs attractive result in classification of Tigrigna word sets.

We employed bagging algorithm for classification purpose of our datasets and it achieved an average accuracy of 81.13%. Its achieved performance was less when we compared to the performances achieved using the two selected bootstrapping algorithms and SMO algorithm but more attractive than the performance achieved using Naïve Bayes algorithm. The average time taken to classify our datasets using bagging algorithm was less than the two selected bootstrapping algorithms and Naïve Bayes but much faster than SMO algorithm which was 0.076 seconds.

AdaBoostM1 algorithm was also one of the selected classification algorithms in our study. Using this algorithm, we achieved an average accuracy of 87.50% which is an attractive result compared to bagging, SMO, and Naïve Bayes algorithms but less attractive than ADTree algorithm. Its efficiency was also high next to Naïve Bayes and ADTree algorithms respectively. Its average running time taken to classify our datasets was 0.01 seconds.

When we employed ADTree algorithm for our datasets, it achieved an effective result. We achieved an average accuracy of 89.48%. This algorithm became an effective for Tigrigna WSD prototype model compared to all of the selected classification algorithms. The average running time taken to classify our datasets was 0.008 seconds. Its efficiency was also better than AdaBoostM1, bagging and SMO algorithm but less than Naïve Bayes algorithm for Tigrigna WSD prototype model.

Lastly, we conclude that bootstrapping and support vector machine algorithms which means ADTree, AdaBoostM1, bagging, and SMO achieved attractive results whereas Naïve Bayes algorithm performed less result on Tigrigna WSD using semi-supervised learning method. ADTree algorithm is the best performer algorithm for Tigrigna WSD using semi-supervised learning method. AdaBoostM1 and SMO algorithms were also performed comparable result to each other for Tigrigna WSD prototype model.

Experiment III: Find out the optimal context window size

To find the optimal context window size, different researchers have conducted different researches using different WSD approaches for different languages. WSD researches were conducted for Amharic language by different researchers starting from one-one to ten-ten window sizes to find out the optimal context window size for this language using different approaches.

Solomon M. [23] was conducted a research for Amharic language using supervised machine learning method for five ambiguous words of the language (*mesasat, meTrat, qereSe, ATenaand mesal*) and he was advised that window size 3-3 is an effective by using Naïve Bayes algorithm.

Solomon A. [22] was also conducted a research for Amharic language using unsupervised machine learning approach and he was advised an optimal context window size of 3-3 or 2-2 is enough for disambiguation purpose depending on the algorithms he was used. As he discussed in his study, window size of 3-3 became optimal window size for Simple K means and EM clustering algorithms, and 2-2 window size became optimal window size for agglomerative SL and CL clustering algorithms on the same five ambiguous words of Amharic language (*mesasat, meTrat, qereSe, ATena* and *mesal).*

Getahun W. [1] was also another researcher who have done a research on Amharic WSD using semi-supervised machine learning method and he advised that the optimal window size is 2-2 or 3-3 window size using five classification algorithms for of the selected five ambiguous words of the language (ATena, derese, tenesa, ale, bela). The optimal window size 3-3 was effective for three of the bootstrapping and SVM algorithms (ADTree, AdaBoostM1, bagging, and SMO), and 2-2 window size became effective using Naïve Bayes algorithm.

Window size of 2-2 became the optimal window size for Tigirgna language using clustering algorithms which was conducted by Meresa M. [21]. But, in this study we tried to find the optimal window size for WSD of Tigrigna language using semi-supervised machine method. To determine the optimal window size for Tigrigna WSD prototype model, the researcher conducted a research for five ambiguous words of Tigrigna language using five classification algorithms. The results and discussion are presented in the following tables. The accuracy that shows the optimal window size for each of the ambiguous word using the selected classification algorithms is represented in bold. Since we have already determined ADTree algorithm became the best performer for Tigrigna WSD in Experiment III; we could determine the optimal window size by considering the results achieved using ADTree algorithm as it is shown in Table 5.8 and 5.9 for this study. However, Table 5.10-5.17 are only used for determining the optimal window size using Bayes, Bootstrapping, and SVM algorithms as it shown in Table 5.19.

Table 5.8: Window size experimentation result using ADTree algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb
1-1	91.8367%	**95%**	90.1235%	**93.6810%**	**97.6769%**
2-2	87.2449%	95%	85.1852%	88.2965%	82.6886%
3-3	88.7755%	87.5%	88.2716%	88.6558%	90.0614%
4-4	87.7551%	88.125%	83.3333%	88.3016%	91.0832%
5-5	**93.8776%**	88.125%	88.8889%	88.4084%	91.8825%
6-6	92.3469%	86.25%	**94.4444%**	90.1281%	95.1749%
7-7	90.8163%	85%	92.5926%	88.6815%	93.9515%
8-8	93.8776%	85%	92.5926%	88.5853%	95.2023%
9-9	83.6735%	81.875%	90.7407%	89.1725%	95.2529%
10-10	90.8163%	81.875%	89.5062%	90.9228%	94.2724%

As it could be seen in the above table 5.8, window size of 1-1 became enough for ambiguous words "OareQe", "Halefe", and "medeb". Since, they achieved the accuracy of 95%, 93.6810%, and 97.6769% respectively. Window sizes of 5-5 and 6-6 became the optimal window sizes for the datasets of "kefele" and "seOare" respectively. Those window sizes became enough for those datasets, because they achieved the performance of 93.8776% and 94.4444% in 5-5 and 6-6 window sizes respectively. From this we can understand that, finding the optimal window size for all of the datasets using ADTree algorithm became difficult; because all of the datasets have different optimal window sizes. Therefore, to determine the optimal window size for all of the selected datasets using ADTree algorithm, we found the average accuracy of each window sizes. The result is shown in the following Table 5.9.

Table 5.9: Average performance of each window size using ADTree algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb	Average
1-1	91.8367%	95%	90.1235%	93.6810%	97.6769%	**93.6636%**
2-2	87.2449%	95%	85.1852%	88.2965%	82.6886%	87.683%
3-3	88.7755%	87.5%	88.2716%	88.6558%	90.0614%	88.6529%
4-4	87.7551%	88.125%	83.3333%	88.3016%	91.0832%	87.7196%
5-5	93.8776%	88.125%	88.8889%	88.4084%	91.8825%	90.2365%
6-6	92.3469%	86.25%	94.4444%	90.1281%	95.1749%	91.6689%
7-7	90.8163%	85%	92.5926%	88.6815%	93.9515%	90.2084%
8-8	93.8776%	85%	92.5926%	88.5853%	95.2023%	91.0516%
9-9	83.6735%	81.875%	90.7407%	89.1725%	95.2529%	88.1429%
10-10	90.8163%	81.875%	89.5062%	90.9228%	94.2724%	89.4785%

As it is shown in the above Table 5.9, window size of 1-1 became the optimal window size for determination of the meaning of the selected ambiguous words using ADTree algorithm by achieving an average accuracy of 93.6636% which was the highest result. Therefore, window size of 1-1 became the optimal window size for identifying the meaning of the the selected ambiguous words. However, the researcher used the following tables to find the optimal window size for the selected ambiguous words of the language using the rest of the selected classification algorithms.

Table 5.10: Window size experimentation result using AdaBoostM1 algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb
1-1	91.8367%	88.75%	90.1235%	62.5974%	**93.4426%**
2-2	85.2041%	**91.875%**	82.716%	69.0909%	62.1399%
3-3	89.2857%	88.125%	85.8025%	61.2987%	81.1475%
4-4	88.2653%	89.375%	87.037%	62.5974%	91.8033%
5-5	90.8163%	89.375%	89.5062%	63.3766%	91.8033%
6-6	88.2653%	88.125%	**93.8272%**	76.8831%	92.2131%
7-7	85.7143%	86.875%	90.7407%	84.4156%	85.2459%
8-8	**92.3469%**	86.875%	90.1235%	85.7143%	90.5738%
9-9	84.1837%	80%	89.5062%	83.8961%	90.9836%
10-10	89.7959%	81.25%	88.2716%	**88.8312%**	89.3443%

As it is shown in the above Table 5.10, the optimal window size for “kefele”, “OareQe”, “seOare”, “Halefe”, and “medeb” became 8-8, 2-2, 6-6, 10-10, 1-1 by achieving the performance of 92.3469%, 91.875%, 93.8272%, 88.8312%, and 93.4426% using AdaBoostM1 algorithm respectively. Here, all of the Tigrigna datasets have different optimal window sizes. This differentiation makes us difficult to find out the optimal window size for all of the Tigrigna datasets using AdaBoostM1 algorithm. Therefore, to find out the optimal window size for all of the datasets the researcher was used the same as he used in the above which means finding the average performance of each ambiguous words in each window size. Then the window size that the highest performance will be the optimal window size for all of the selected Tigrigna datasets. The result is shown in the following Table 5.11.

Table 5.11: Average performance of each window size using AdaBoostM1 algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb	Average
1-1	91.8367%	88.7500%	90.1235%	62.5974%	**93.4426%**	85.3500%
2-2	85.2041%	**91.8750%**	82.7160%	69.0909%	62.1399%	78.2052%
3-3	89.2857%	88.1250%	85.8025%	61.2987%	81.1475%	81.1319%
4-4	88.2653%	89.3750%	87.0370%	62.5974%	91.8033%	83.8156%
5-5	90.8163%	89.3750%	89.5062%	63.3766%	91.8033%	84.9755%
6-6	88.2653%	88.1250%	**93.8272%**	76.8831%	92.2131%	87.8627%
7-7	85.7143%	86.8750%	90.7407%	84.4156%	85.2459%	86.5983%
8-8	**92.3469%**	86.8750%	90.1235%	85.7143%	90.5738%	**89.1267%**
9-9	84.1837%	80.0000%	89.5062%	83.8961%	90.9836%	85.7139%
10-10	89.7959%	81.2500%	88.2716%	**88.8312%**	89.3443%	87.4986%

As we have seen in the above Table 5.11, window size of 8-8 became enough for sense determination of all the selected Tigrigna datasets by achieving an average accuracy of 89.1267% using AdaBoostM1 algorithm. We said that window size of 8-8 became the optimal window size in order to differentiate the meaning of the selected Tigrigna ambiguous words using AdaBoostM1 algorithm for our study. The reason that we have said window size 8-8 became the optimal window size because the highest performance was achieved on this window size.

Table 5.12: Window size experimentation using Bagging algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb
1-1	75%	75%	74.0741%	78.4416%	**93.4426%**
2-2	64.2857%	**83.125%**	64.8148%	66.7532%	51.4403%
3-3	57.1429%	73.75%	66.6667%	64.1558%	74.5902%
4-4	61.2245%	78.125%	61.1111%	53.7662%	77.8689%
5-5	82.1429%	78.125%	79.6296%	60.2597%	68.4426%
6-6	85.2041%	78.125%	86.4198%	54.5455%	69.6721%
7-7	87.7551%	76.875%	85.1852%	83.3766%	69.6721%
8-8	**91.8367%**	76.875%	85.1852%	49.3506%	74.5902%
9-9	64.2857%	78.75%	86.4198%	82.8571%	72.9508%
10-10	89.7959%	83.125%	**88.2716%**	**88.3117%**	56.1475%

As it is shown in the above Table 5.12, window sizes of 8-8, 2-2, 10-10, 10-10, and 1-1 became enough for Tigrigna datasets of “kefele”, “OareQe”, “seOare”, “Halefe”, and “medeb” by achieving an accuracy of 91.8367%, 83.125%, 88.2716%, 88.3117%, and 93.4426% respectively using bagging algorithm. From we can understand that, most of the selected datasets score the highest performance at different window size. The window size that scores the highest

performance will be the optimal window size of that dataset. Scoring highest performance at different window sizes make difficult to differentiate which window size became the optimal window size for all of the selected datasets using bagging algorithm. Therefore, to determine one optimal window size for the whole selected Tigrigna datasets calculating the average accuracy of each window size of the selected Tigrigna datasets became important. The result is shown in the following Table 5.13.

Table 5.13: Average performances of each window size using bagging algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb	Average
1-1	75.0000%	75.0000%	74.0741%	78.4416%	93.4426%	79.1917%
2-2	64.2857%	83.1250%	64.8148%	66.7532%	51.4403%	66.0838%
3-3	57.1429%	73.7500%	66.6667%	64.1558%	74.5902%	67.2611%
4-4	61.2245%	78.1250%	61.1111%	53.7662%	77.8689%	66.4191%
5-5	82.1429%	78.1250%	79.6296%	60.2597%	68.4426%	73.7200%
6-6	85.2041%	78.1250%	86.4198%	54.5455%	69.6721%	74.7933%
7-7	87.7551%	76.8750%	85.1852%	83.3766%	69.6721%	80.5728%
8-8	91.8367%	76.8750%	85.1852%	49.3506%	74.5902%	75.5675%
9-9	64.2857%	78.7500%	86.4198%	82.8571%	72.9508%	77.0527%
10-10	89.7959%	83.1250%	88.2716%	88.3117%	56.1475%	**81.1303%**

As we have seen in the above Table 5.13, window size of 10-10 became the optimal window size in order to differentiate the meaning of the selected ambiguous words of Tigrigna language by achieving an average of accuracy of 81.1303%. From this we can conclude that, window size of 10-10 became enough for determination of the senses of the selected Tigrigna datasets using bagging algorithm when we used semi-supervised learning methods.

Table 5.14: Window size experimentation result using SMO algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb
1-1	**93.8776%**	90.625%	91.9753%	88.0519%	**95.082%**
2-2	89.7959%	**95%**	91.9753%	74.8052%	86.8313%
3-3	89.7959%	88.75%	87.6543%	84.1558%	88.1148%
4-4	88.7755%	88.75%	87.6543%	83.6364%	91.3934%
5-5	91.8367%	88.125%	90.7407%	80.7792%	92.623%
6-6	88.2653%	85.625%	**97.5309%**	84.4156%	93.0328%
7-7	89.2857%	85%	89.5062%	86.4935%	88.1148%
8-8	92.8571%	83.125%	88.8889%	85.974%	90.9836%
9-9	82.1429%	83.125%	87.6543%	85.974%	90.9836%
10-10	89.7959%	80.625%	88.2716%	**88.5714%**	89.7541%

As it is shown in the above Table 5.14, window size 1-1 became enough for the datasets of "kefele" and "medeb" by achieving an accuracy of 93.8776% and 95.082% respectively. Window size of 2-2, 6-6, and 10-10 became the optimal window sizes for the datasets of "OareQe", "seOare", and "Halefe" by achieving an accuracy of 95%, 97.5309%, and 88.5714% respectively. Here, different window sizes became the optimum window size for different datasets. This was the difficulty to determine which window size became enough to differentiate the senses of the selected ambiguous words using SMO algorithm. Therefore, the researcher was tried to calculate the average accuracy of each window size for determining the optimal window size. The window size which scores the highest average accuracy became the optimal window size for differentiating the senses of the selected datasets using SMO algorithm. The result is shown in the following Table 5.15.

Table 5.15: Average performances of each window size using SMO algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb	Average
1-1	93.8776%	90.6250%	91.9753%	88.0519%	95.0820%	**91.9224%**
2-2	89.7959%	95%	91.9753%	74.8052%	86.8313%	87.6815%
3-3	89.7959%	88.75%	87.6543%	84.1558%	88.1148%	87.6942%
4-4	88.7755%	88.75%	87.6543%	83.6364%	91.3934%	88.0419%
5-5	91.8367%	88.125%	90.7407%	80.7792%	92.6230%	88.8209%
6-6	88.2653%	85.625%	97.5309%	84.4156%	93.0328%	89.7739%
7-7	89.2857%	85%	89.5062%	86.4935%	88.1148%	87.68%
8-8	92.8571%	83.125%	88.8889%	85.9740%	90.9836%	88.3657%
9-9	82.1429%	83.125%	87.6543%	85.9740%	90.9836%	85.976%
10-10	89.7959%	80.625%	88.2716%	88.5714%	89.7541%	87.4036%

As shown in the above Table 5.15, window size of 1-1 scores the highest performance when we compared to other window sizes. Window size that scores highest performance became enough to differentiate the meaning of the selected datasets. Since window size of 1-1 scores highest performance in average, the researcher concluded that window size of 1-1 became the optimal window size for the selected Tigrigna datasets using SMO algorithm by achieving an average accuracy of 91.9224%.

Table 5.16: window size experimentation using Naive Bayes algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb
1-1	**77.0408%**	50%	48.1481%	**88.3117%**	**90.9836%**
2-2	52.0408%	**87.5%**	**72.2222%**	73.5065%	55.144%
3-3	44.898%	42.5%	68.5185%	77.4026%	77.0492%
4-4	43.8776%	62.5%	54.9383%	50.3896%	62.2951%
5-5	46.4286%	56.875%	68.5185%	64.9351%	43.0328%
6-6	38.7755%	58.125%	61.7184%	36.8831%	34.8361%
7-7	33.1633%	57.5%	57.4074%	47.7922%	36.0656%
8-8	30.102%	56.875%	56.1728%	36.1039%	33.1967%
9-9	32.6531%	46.25%	50%	43.6364%	32.7869%
10-10	29.0816%	46.25%	48.7654%	37.9221%	25.8197%

As it could be seen in the above Table 5.16, window size of 1-1 became the optimal window size for the datasets of “kefele”, “Halefe”, and “medeb” by achieving an accuracy of 77.0408%, 88.31175% and 90.9836% respectively. And window size of 2-2 became the optimal window size for the datasets of “OareQe” and “seOare” by achieving an accuracy of 87.5% and 72.2222% respectively. Using Naïve Bayes algorithm, the same thing is happened as in the other algorithm was happened which is different window sizes became the optimal window size for different datasets as we have in the above. For instance, window size of 1-1 became the optimal window size for datasets of “kefele”, “Halefe”, and “medeb”; and window size of 2-2 became the optimal window size for datasets of “OareQe”, and “seOare”. Here, two window sizes became the optimal window sizes for the selected Tigrigna datasets using Naïve Bayes algorithm. But, the researcher wanted to determine a single window size that became the optimal window size for all of the selected datasets. To do this, the researcher was tried to calculate the average performances of each window size. The result is shown in the following Table 5.17.

Table 5.17: Average performances of each window size using Naïve Bayes algorithm

Window size	kefele	OareQe	seOare	Halefe	medeb	Average
1-1	77.0408%	50.0000%	48.1481%	88.3117%	90.9836%	**70.8968%**
2-2	52.0408%	87.5000%	72.2222%	73.5065%	55.1440%	68.0827%
3-3	44.8980%	42.5000%	68.5185%	77.4026%	77.0492%	62.0737%
4-4	43.8776%	62.5000%	54.9383%	50.3896%	62.2951%	54.8001%
5-5	46.4286%	56.8750%	68.5185%	64.9351%	43.0328%	55.9580%
6-6	38.7755%	58.1250%	61.7184%	36.8831%	34.8361%	46.0676%
7-7	33.1633%	57.5000%	57.4074%	47.7922%	36.0656%	46.3857%
8-8	30.1020%	56.8750%	56.1728%	36.1039%	33.1967%	42.4901%
9-9	32.6531%	46.2500%	50.0000%	43.6364%	32.7869%	41.0653%
10-10	29.0816%	46.2500%	48.7654%	37.9221%	25.8197%	37.5678%

As we have seen in the above Table 5.17, the researcher was found the maximum performance at window size of 1-1 by achieving an average accuracy of 70.8968%. Therefore, window size of 1-1 became enough to differentiate the senses of the selected Tigrigna datasets using Naïve Bayes algorithm which means window size of 1-1 became the optimal window size.

Discussion about the optimal window size for Tigrigna datasets

In the above discussion, we have discussed about the optimal window sizes of each classification algorithms which means at which window size the selected classification algorithm performs highest performance. But in this topic, we would discuss about which window size became the optimal window size for bootstrapping algorithms, SVM algorithms, and Bayes algorithm.

To find the optimal window size for bootstrapping algorithms, the researcher took the average accuracy of each window size in each bootstrapping algorithm and calculating the average of those accuracies in each window size. Then window size that scores the highest average performance will be the optimal window size for all of the selected bootstrapping algorithms. The result is shown in the following Table 5.18.

Table 5.18: Average performances of the selected bootstrapping algorithms

Average accuracy using ADTree	Average accuracy using AdaBoostM1	Average accuracy using Bagging	Average
93.6636%	85.3500%	79.1917%	**86.0685%**
87.6830%	78.2052%	66.0838%	77.3240%
88.6529%	81.1319%	67.2611%	79.0153%
87.7196%	83.8156%	66.4191%	79.3181%
90.2365%	84.9755%	73.7200%	82.9773%
91.6689%	87.8627%	74.7933%	84.7750%
90.2084%	86.5983%	80.5728%	85.7932%
91.0516%	89.1267%	75.5675%	85.2486%
88.1429%	85.7139%	77.0527%	83.6365%
89.4785%	87.4986%	81.1303%	86.0358%

As it is shown in the above Table 5.18, window size of 1-1 became the optimal window size for bootstrapping algorithms by achieving an average accuracy of 86.0685%. From this we concluded that, window size of 1-1 became enough to differentiate the meaning of the selected Tigrigna datasets using the selected bootstrapping algorithms. But to determine the optimal window size for SVM and Bayes supervised algorithms did not need to calculate the average accuracy of each window again because we used one SVM algorithm and one Bayes supervised algorithm which are SMO and Naive Bayes algorithms respectively. Therefore, we already have calculated the average accuracy in the above Table 5.15 and 5.17. Then window size that was scored the highest average accuracy became the optimal window size for the selected Tigrigna datasets. As we have already seen in Table 5.15 and 5.17, window size of 1-1 and 1-1 became the optimal window size for SMO and Naïve Bayes algorithm by achieving an average accuracy of 91.9224% and 70.8968% respectively. This means window size of 1-1 became enough for the selected datasets using Naïve Bayes and SMO algorithms.

By considering the above discussion we concluded that, window size of 1-1 became the optimal window size using bootstrapping, Naïve Bayes and SMO algorithms for differentiating the meaning of the selected ambiguous words. The result that shows the optimal window size and its accuracy is shown in the following Table 5.19.

Table 5.19: Results of optimum window sizes for the selected datasets

Ambiguous words	Optimal Window size and Algorithms				
	Bootstrapping (1-1)			SVM (1-1)	Bayes (1-1)
	ADTree	AdaBoostM1	Bagging	SMO	Naïve Bayes
kefele	91.8367%	91.8367%	75%	93.8776%	77.0408%
OareQe	95%	88.75%	75%	90.625%	50%
seOare	90.1235%	90.1235%	74.0741%	91.9753%	48.1481%
Halefe	93.6810%	62.5974%	78.4416%	88.0519%	88.3117%
medeb	97.6769%	93.4426%	93.4426%	95.082%	90.9836%
Average	**93.6636%**	**85.3500%**	**79.1917%**	**91.9224%**	**70.8968%**

From this table we concluded that, bootstrapping and SVM algorithms perform much better than Bayes algorithm which means bootstrapping algorithms (ADTree, AdaBoostM1, and Bagging), SVM algorithm (SMO), and Bayes algorithm (Naïve Bayes). ADTree, AdaBoostM1, and bagging achieving an average performance of 93.6636%, 85.35%, and 79.1917% respectively; whereas, SMO and Naïve algorithm achieving an average performance of 91.9224% and 70.8968% respectively. Therefore, ADTree algorithm became the best performer algorithm for the selected Tigrigna datasets. SMO was also the best performer algorithm next to ADTree algorithm using semi-supervised learning method. Lastly, we concluded that using ADTree algorithm and SMO algorithm is advisable for WSD prototype model of Tigrigna datasets depending on our experiments.

5.2. Summary

In general, the study was focused on developing WSD prototype model for Tigrigna language using semi-supervised machine learning approach. Even though the study focused on semi-supervised machine learning approach; we tried to compare unsupervised, semi-supervised, and supervised machine learning techniques. Therefore, the performance achieved using semi-supervised approach became much better than supervised approach and unsupervised approach. To do this comparison, the same classification algorithms and the same datasets were used for supervised and semi-supervised approach. But, the same datasets were used for three of the machine learning methods.

In addition to comparison of machine learning methods, we tried to do three experiments. One was comparison of results obtained in using seed words selected manually and selected using ADTree algorithm. Here, the results that we achieved were attractive results using both of the methods. When we used seed words selected manually, AdaBoostM1 algorithm performed better; whereas in using seed words selected using ADTree algorithm, ADTree algorithm by itself achieved better result in our datasets. But, for this study we select seed words using ADTree algorithm. The other experiment that was applied in this study was identifying the best performing algorithm. In this experiment, ADTree algorithm performed better compared to the other selected classification algorithms. The last but not the least experiment that we tried to apply in this study was identifying the optimal context window size. Therefore, in this experiment window size of 1-1 became the optimal window size for our datasets averagically.

Chapter Six

6. Conclusion and Recommendations

6.1. Conclusion

Many words have more than one meaning in natural language, and each one of them is determined by its context. The automated process of recognizing word senses in context is known as Word Sense Disambiguation (WSD). The problem of distinguishing between multiple possible senses of a word is an important subtask in many NLP applications such as machine translation (MT), semantic mapping (SM), semantic annotation (SA), and ontology learning (OL). It is also believed to be helpful in improving the performance of many applications such as information retrieval (IR), information extraction (IE), speech recognition (SR) and others.

There are three main approaches used for assigning senses to words in a context. These are knowledge-based approaches, corpus-based approaches and hybrid approach. From those approaches, we select corpus-based approaches for this study. Corpus-based approach further divided as supervised, unsupervised, and semi-supervised approach. From the corpus-based approaches, we used semi-supervised approach for this study.

This research work is the first attempt to develop a word sense disambiguation system for Tigrigna Language using semi-supervised learning method. As there is no large size corpus which is already prepared for Tigrigna language for WSD purpose, we prepared Tigrigna corpus manually. Although Tigrigna language has many ambiguous words, we selected only five ambiguous words to build Tigrigna WSD prototype model from Tigrigna dictionary. These words are kefele, OareQe, seOare, Halefe, and medeb. For those selected ambiguous words of Tigrigna language, a separate dataset has been prepared. The datasets were composed of a total of 1250 Tigrigna sentences which were collected from different domain areas. To make comfortable for the selected machine learning tool which is Weka, the ambiguous word was written at the middle of sentences with sets of surrounding contexts to the left and right for disambiguation purpose. Ten to the left and ten to the right was the maximum window sizes used for this work.

In this study, four experiments have been conducted using different classification and clustering algorithms. The first experiment was comparison of the results obtained using the three machine

learning methods which means unsupervised, semi-supervised, and supervised learning methods. In this experiment, semi-supervised approach was performed better compared to the other machine learning methods. Since semi-supervised learning method was employed in this work, we used the final fully labeled dataset which is obtained from unlabeled data sets (now labeled after clustering). Those unlabeled datasets were labeled after clustering using clustering assumption. But for clustering purpose we used EM algorithm because EM algorithm performs better results compared to the other selected clustering algorithms. The second experiment was comparing results obtained using seed words selected by humans manually and results obtained using seed words selected by using ADTree algorithm. The results obtained by using both methods were comparable to each other. ADTree and SMO algorithms performed better when we used seed words selected using ADTree algorithm. But AdaBoostM1, Bagging, and Naïve Bayes algorithms performed better when we used seed words selected by humans manually as we have seen the result in Table 5.5. The third experiment was determining the best performing algorithm for the selected Tigrigna datasets. To do this experiment, we used seed words selected using ADTree algorithm. Here representative seed examples were selected depending on the seed words that have high information gain using ADTree algorithm. Therefore, the best performing algorithm for the selected Tigrigna datasets was ADTree algorithm compared to AdaBoostM1, Bagging, SMO, and Naïve Bayes algorithms.

The last but not the least experiment that was conducted in this study was investigating the optimal window size for determining the senses of each ambiguous word. Using all of the selected algorithms window size one-one became enough for differentiating the meaning of the selected Tigrigna datasets. Therefore, we concluded that window size of 1-1 can be standard window size for Tigrigna WSD systems.

Seed word selection was also becoming one of the most important task in this study. Seed word selection requires words that are used to differentiate the senses of each ambiguous word. Those seed words are used to select representative seed examples from the selected datasets. Therefore, we select seed words heuristically to represent seed examples. This seed word selection was taking place using ADTree algorithm. However, selecting seed words using ADTree was also became the challenging task for two of the selected ambiguous words of the language which means for ambiguous words such as “medeb” and “Halefe”. Because those two words have three and four

senses respectively. But ADTree algorithm works only for words having two senses only. Therefore, we tried to solve this problem by using combination method which means using a binary combination of the sense of those ambiguous words.

In general, we conclude that Semi-supervised learning using bootstrapping algorithms and SVM is a potential learning method which performs better in our study. They are more adaptive on WSD for the Tigrigna datasets. Specifically, ADTree, AdaBoostM1 and SMO are potential algorithms to be applied for Tigrigna WSD systems using semi-supervised learning methods.

6.2. Recommendations

Word sense disambiguation researches require variety of linguistic resources like thesaurus, WordNet, Machine-Readable Dictionaries, and effective Tigrigna language stemmer in which we faced a significant challenge as Tigrigna lacks those resources. Lack of sense annotated data for the language was also another challenge that makes us to limit our study on five ambiguous words of the language. Lack of sense annotated data also makes us to limit our dataset on 1250 sentences only. Therefore, we have the following recommendations which include the development of resources and future research directions for WSD of Tigrigna texts:

- Researches in WSD for other languages used linguistic resources like Thesaurus, Lexicon, WordNet, machine readable dictionaries and effective Tigrigna language stemmer. Due to the unavailability of those resources for Tigrigna language; reseachers could not conduct a research on WSD of Tigrigna language. By considering their contribution to WSD; other researchers can conduct a research for preparing these resources with concerned institutions or individuals.
- As using larger corpus size believed more realistic for WSD model, extending this experimentation using semi-supervised learning for other ambiguous words in addition to corpora covered in this study is recommended.
- This study has only concentrated on modeling WSD to tackle lexical ambiguity. Further researches are recommended to address other type of ambiguities in Tigrigna language like semantic, phonological, referential etc.
- Developing WSD model by applying part-of-speech tagging can improve the performance WSD model as Biruk R. [59] investigated in his study by using the corpura of Getahun W.

[1]. Therefore, other researchers should investigate the effect of applying part-of-speech tagging for WSD prototype model of Tigrigna language by using the data that we have used in this study.

- Semi-supervised learning can be accomplished in two ways based on semi-supervised assumptions. These are semi-supervised classification and semi-supervised clustering assumptions. For this study, we have followed semi-supervised clustering assumption. Other researchers should test semi-supervised classification which needs label propagation.
- In addition to corpus-based approach there are also knowledge-based and hybrid approaches which were used for WSD of other languages and achieved a good result. Therefore, we recommend that these approaches need to be investigated for Tigrigna language as well.
- In this study we conducted our research by removing prefixes and suffixes only and we achieved an attractive result. However, Meresa M. [21] investigated that applying stemming for the Tigrigna texts improve the performance of WSD prototype model of the language. Because Tigrigna language is morphologically rich language. Therefore, developing an effective stemmer for Tigrigna language become important inorder to develop an effective WSD model. Then, we recommend other researchers to develop an effective stemmer for Tigrigna language by creating team with tigrigna experts.

References

[1] Getahun, Wassie; Ramesh, Babu P; Solomon, Teferra; Million, Meshesha;, "A Word Sense Disambiguation Model for Amharic Words using Semi-Supervised Learning Paradigm," *Science, Technology and Arts Research Journal,* vol. 3, pp. 147-155, July-Sep 2014.

[2] Daniel Jurafsky; Martin, James H, Speech and Language Processing: An introduction to natural language processing,, 2007.

[3] Jackson, Peter; Moulinier, sabelle;, Natural language processing for online application:Text Retrieval, Extraction and Categorization, vol. Volume 5, P. R. Mitkov, Ed., Stafford St., 1984.

[4] L. E., A book called: In Encyclopedia of Library and Information Science, 2nd Edition ed., 2011.

[5] Jisha , P.Jayan; Rajeev, R R; , Dr. S Rajendran;, "Morphological Analyser and Morphological Generator for Malayalam- Tamil Machine Translation," *International Journal of Computer Applications (0975–8887),* vol. Volume 13, pp. 1-4, January 2011.

[6] Jumi, Sarmah; Dr. Shikhar, Kumar Sarma;, "Survey On Word Sense Disambiguation: An Initiative towards an Indo-Aryan Language," *I.J. Engineering and Manufacturing,* vol. 3, pp. 37-52, 03 May 2016.

[7] Thanh , Phong Pham; Hwee , Tou Ng; Wee , Sun Lee;, *Word Sense Disambiguation with Semi-Supervised Learning,* 2005.

[8] Clara; , Allan M; , Lotofus; , Elizabeth; , F;, "A spreading-activation theory of semantic processing. Psychological Review," vol. Vol 86, p. 6, 1975.

[9] Nancy, Ide; Jean, Veronis;, "Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art," *Computational Linguistics,* 1998.

[10] Alok, Ranjan Pal; Diganta, Saha;, "WORD SENSE DISAMBIGUATION: A SURVEY," *International Journal of Control Theory and Computer Modeling (IJCTCM),* vol. Vol.5, July 2015.

[11] T. K. Hundesa, "WORD SENSE DISAMBIGUATION FOR AFAAN OROMO LANGUAGE," Addis Ababa, 2013.

[12] S. H. Yesuf, "AMHARIC WORD SENSE DISAMBIGUATION USING WORDNET," Addis Ababa, 2015.

[13] S. A, "Sawasasəw Tigrigna Besafih," Asmera, 1998.

[14] R. NAVIGLI, "Word Sense Disambiguation: A Survey," in *ACM Computing Surveys*, vol. Vol 41, 2009.

[15] A. B. GEBREHIWOT, "A TWO STEP APPROACH FOR TIGRIGNA TEXT CATEGORIZATION," Addis Ababa, 2011.

[16] H. B. Berhe, "DESIGN AND DEVELOPMENT OF TIGRIGNA SEARCH ENGINE," Addis Ababa, 2013.

[17] H. A. Miruts, "Hidden Markov Model Based Tigrigna Speech Recognition," Addis Ababa, 2009.

[18] O. T. Yohannes, "Phonetic-Based Keyboard Mapping For Tigrigna Language," Nairobi, Kenya, 2014.

[19] T. G. ABREHA, "PART OF SPEECH TAGGER FOR TIGRIGNA LANGUAGE," Addis Ababa, 2010.

[20] Y. Fisseha, "Development of Stemming Algorithm for Tigrigna Language," Addis Ababa, 2011.

[21] M. Mebrahtu, "UNSUPERVISED MACHINE LEARNING APPROACH FOR TIGRIGNA WORD SENSE DISAMBIGUATION," Gondar, March, 2017.

[22] S. Assemu, "UNSUPERVISED MACHINE LEARNING APPROACH FOR WORD SENSE DISAMBIGUATION TO AMHARIC WORDS," Addis Ababa, 2011.

[23] S. Mekonnen, "WORD SENSE DISAMBIGUATION FOR AMHARIC TEXT: A MACHINE LEARNING APPROACH," ADDIS ABABA, 2010.

[24] S, Rajasekar; P, Philominathan; V, Chinnathambi;, RESEARCH METHODOLOGY, Tamilnadu, 2013.

[25] T. Kassie, "WORD SENSE DISAMBIGUATION FOR AMHARIC TEXT RETRIEVAL: A CASE STUDY FOR LEGAL DOCUMENTS," Addis Ababa, 2009.

[26] Ping, Chen; Wei, Ding; Chris, Bowes; David, Brown;, "A Fully Unsupervised Word Sense Disambiguation Method Using Dependency Knowledge," *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL,* pp. 28-36, June 2009.

[27] W. NG'ANG'A, "Word Sense Disambiguation of Swahili:," in *Faculty of Arts of the University of Helsinki*, University of Helsinki, 2005.

[28] Gerard, Escudero; Lluis, Marquez; German, Rigau;, *A Comparison between Supervised Learning Algorithms for Word Sense Disambiguation,* 7 July 2000.

[29] A. Dey, "Machine Learning Algorithms: A Review," *International Journal of Computer Science and Information Technologies,* vol. Vol. 7 (3), pp. 1174-1179, 2016.

[30] S. B. , Kotsiantis; I. D. , Zaharakis; P. E. , Pintelas;, "Machine learning: a review of classification and combining techniques," in *Artif Intell Rev*, 10 November 2007.

[31] Shai , Shalev-Shwartz; Shai, Ben-David;, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014.

[32] Mihir, Sawant; Tanya, Sangoi; Sindhu, Nair;, "Supervised Word Sense Disambiguation," *International Journal of Science and Research (IJSR),* vol. 5, no. 10, pp. 1845-1848, October 2016.

[33] Yoong , Keok Lee; Hwee , Tou Ng;, "An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation," in *In Proceedings of the Conference on Empirical Methods in Natural Language Processing*, National University of Singapore, 2000.

[34] Mahesh, Joshi; Ted, Pedersen; Richard, Maclin;, "A Comparative Study of Support Vector Machines Applied to the Supervised Word Sense Disambiguation Problem in the Medical Domain," in *Indian International conference on Artificial Intelligence*, University of Minnesota, Duluth, MN 55812, USA, December 20-22 2005.

[35] Narendra , Sharma; Aman , Bajpai; Mr. Ratnesh , Litoriya;, "Comparison the various clustering algorithms of weka tools," *International Journal of Emerging Technology and Advanced Engineering,* vol. Volume 2, no. Issue 5, pp. 73-80, May 2012.

[36] B. Pooja , "Study of Various Clustering Algorithms Used by WEKA Tool," *International Journal of Emerging Research in Management &Technology,* Vols. Volume-4, no. Issue-8, pp. 37-40, August (2015).

[37] Garima , Sehgal; Dr. Kanwal , Garg;, "Comparison of Various Clustering Algorithms," *International Journal of Computer Science and Information Technologies (IJCSIT),* vol. Vol. 5 (3), pp. 3074-3076, 2014.

[38] Yogita , Rani[1] ; Dr. Harish , Rohil;, "A Study of Hierarchical Clustering Algorithm," *International Journal of Information and Computation Technology,* vol. Volume 3, pp. 1225-1232, Number 11 (2013).

[39] Manning , Christopher; Hinrich , Schutze;, Foundations of Statistical Natural Language Processing, 2nd ed ed., London: The MIT press, 1990.

[40] B. King, "Step-wise clustering procedures," *Journal of the American Statistical Association,* vol. Vol 69, p. 86–101, 1967.

[41] Pamulaparty , L; Rao , G;, "Novel Approach to Perform Document Clustering Using Effectiveness and Efficiency of Simhash," *International Journal of Engineering and Advanced Technology (IJEAT),* vol. Vol. 2, February 2013.

[42] Richa , Agrawal; Jitendra , Agrawal;, "Analysis of Clustering Algorithm of Weka Tool on Air Pollution Dataset," *International Journal of Computer Applications ,* vol. Volume 168, no. 13, pp. 1-5, June 2017.

[43] Richa , Loohach; Dr. Kanwal , Garg;, "Effect of Distance Functions on Simple K-means Clustering Algorithm," *International Journal of Computer Applications,* vol. Volume 49, July 2012.

[44] Sharmila ; R.C Mishra;, "Performance Evaluation of Clustering Algorithms," *International Journal of Engineering Trends and Technologies,* vol. Volume 4, no. Issue 7, July 2013.

[45] M. A. Sati, "Review: Semi-Supervised Learning Methods for Word Sense Disambiguation," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. Volume 12, no. Issue 4, Jul. - Aug. 2013.

[46] Z. Xiaojin , "Semi-Supervised Learning Literature Survey," 2008.

[47] Andres , Montoyo; Armando , Su´arez; German , Rigau; Manuel , Palomar;, "Combining Knowledge- and Corpus-based Word-Sense-Disambiguation Methods," *Journal of Artificial Intelligence Research,* 2005.

[48] Tomoya , Saka; Marthinus , Christoffel du Plessis; Gang , Niu; Masashi , Sugiyama;, "Semi-Supervised Classification Based on Classification from Positive and Unlabeled Data," in *Proceedings of the 34 th International Conference on Machine Learning*, Sydney, Australia, 2017.

[49] R. Philippe , "Generalization Error Bounds in Semi-supervised Classification Under the Cluster Assumption," *Journal of Machine Learning Research 8,* pp. 1369-1392, 2007.

[50] V. Jothi , Prakash; Dr. L.M, Nithya;, "A Survey On Semi-Supervised Learning Techniques," *International Journal of Computer Trends and Technology (IJCTT),* vol. volume 8, Feb 2014.

[51] Artur, Ferreira; Mario , Figueiredo;, "Boosting Algorithms: A Review of Methods, Theory, and Applications," Lisbon, Portugal.

[52] L. Breiman, "Bagging predictors. Machine Learning," UC Berkely,UC, 1996.

[53] Gerard , Escudero; Llu´ıs , M`arquez; German , Rigau;, "Boosting Applied to Word Sense Disambiguation," TALP Research Center, Barcelona, Catalonia, july 7,2000.

[54] Yoav , Freund; Robert , E. Schapire;, "Experiments with a new boosting algorithm," in *In Thirteenth International Conference on Machine Learning*, Bari, Italy, 1996.

[55] Yoav , Freund ; Robert , E. Schapire;, "A decision-theoretic generalization of on-line learning and appplication to boosting," in *In European Conference on Computational Learning Theory*, 1994.

[56] M. Zampieri, "A SUPERVISED MACHINE LEARNING METHOD FOR WORD SENSE DISAMBIGUATION OF PORTUGUESE NOUNS".

[57] Zeynep, Orhan; Zeynep, Altan;, "Word Sense Disambiguation for Semantic Applications," pp. 321-328.

[58] Ravi, Mante; Mahesh, Kshirsagar; Dr. Prashant, Chatur;, "A Review Of Literature On Word Sense Disambiguation," *International Journal of Computer Science and Information Technologies,* vol. 5, pp. 1475-1477, 2014.

[59] B. Retta, "Application of Part-of-Speech Tagged Corpus to improve the Performance of Word Sense Disambiguation: the case of Amharic," Addis Ababa, June, 2015.

[60] Y. B. Tesema, "Hybrid Word Sense Disambiguation Approach for Afaan Oromo Words," Addis Ababa, 2016.

[61] Omer, Osman I Ibrahim; Yoshiki , Mikami;, "Stemming Tigrinya Words for Information Retrieval," *Proceedings of COLING,* p. 345–352, December 2012.

[62] T. (. Tewolde, A modern grammar of Tigrigna, Dettivia G. Savonarola Roma., 2002.

[63] L. Wolf, "Documents Tigrigna," Libraire CKlincksieck, Paris, 1998.

[64] J. Mason, "ሰዋሰው ትግርኛ (Tigrigna Grammar)," The Red Sea Press, Inc., Lawrenceville.

[65] Atelach, Alemu Argaw; Lars, Asker;, "An Amharic Stemmer : Reducing Words to their Citation Forms," *Association for Computational Linguistics,* pp. 104-110, June 2007.

[66] D. Teklu, "ዘበናዊ ሰዋሰው ቋንቋ ትግርኛ," Mega printing enterprise, መቐለ, 2000 ዓ/ም.

[67] K. G/hiwot, "ሰዋሰው ትግርኛ," Mega printing enterprise, ኣዲስ ኣበባ, 2004.

[68] M. M. Tayebeh and D. K. Ali , "Word Sense Disambiguation Using Target Language Corpus in a Machine Translation System," *Literary and Linguistic Computing,* vol. 20, no. 2, pp. 237-249, 2005.

[69] K. Robert and C. W.Bruce, "Lexical Ambiguity and Information Retrieval," Amherst.

[70] E. Önder, *DEVELOPING METHODS FOR WORD SENSE DISAMBIGUATION,* Boğaziçi University, 2007.

[71] ተኽስተ , ተኽለ; ዳኒኤል, መሓሪ; ጸሃይነሽ, ገብረዮውሃንስ; ፀጋይ , ወልደማርያም; ታደሰ, ተኽለ; ተስፋይ, ተወልደ;, መዝገበ ቃላት ትግርኛ ብትግርኛ, ኣዲስ ኣበባ: ብናይ ኢትዮጵያ ቋንቋታት ኣካዳሚ, 1989, pp. 1-970.

[72] Y. Alginahi, "Preprocessing Techniques in Character Recognition," Taibah University, Kingdom of Saudi Arabia.

[73] Yitna , Firdyiwek ; Daniel Yaqob;, "The System for Ethiopic Representation in ASCII," 1993-1997.

[74] N. Lokesh and M. Kalyani, "Supervised, Semi-Supervised and Unsupervised WSD Approaches: An Overview," *International Journal of Science and Research (IJSR),* vol. 4, no. 2, pp. 1684-1688, February 2015.

[75] Zheng-Yu, Niu; Dong-Hong, Ji; Chew-Lim, Tan;, "Word Sense Disambiguation Using Label Propagation Based Semi-supervised Learning," *Institute for Infocomm Research.*

[76] Qiong Liu; Stephen Levinson; Ying Wu; Thomas Huang;, "Interactive and Incremental Learning via a Mixture of Supervised and Unsupervised Learning Strategies," *Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign.*

[77] Huang, J.; Lu, J.; Ling, C. X.;, "Comparing naive Bayes, decision trees, and SVM with AUC and accuracy. In Data Mining, 2003. ICDM 2003.," *Third IEEE International Conference on IEEE,* pp. 553-556, November 2003.

[78] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Microsoft Research, April 21, 1998.

[79] Gerard , Escudero; Llu´ıs , M`arquez; German , Rigau;.*Naive Bayes and Exemplar-Based approaches to Word Sense Disambiguation Revisited.*

[80] Gerard, Escudero; Lluis, Marquez; German, Rigau;, *A Comparison between Supervised Learning Algorithms for Word Sense Disambiguation.*

[81] M. Lesk, "Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Creame Cone," *Bell Communications Research Morristown, NJ0760,* pp. 24-26, 1987.

[82] G. berhe, "A Stemming algorithm development for Tigrigna language text documents," Addis Ababa, 2001.

[83] Lars , Asker; Atelach , Alemu Argaw; Björn , Gambäck; Magnus, Sahlgren;, "Applying Machine Learning to Amharic Text Classification".

[84] C. Fellbaum, "WordNet: An Electronic Lexical Database," *MIT Press,* 1998.

[85] Rahmat , Widia Sembiring; Jasni , Mohamad Zain; Abdullah , Embong;, "A Comparative Agglomerative Hierarchical Clustering Method to Cluster Implemented Course," *JOURNAL OF COMPUTING,* vol. VOLUME 2, no. ISSUE 12, pp. 1-2, DECEMBER 2010.

Appendixes

Appendix 1: Tigrigna Alphabets (‘Fidels’) with their respective Latin Scripts

Tigrigna letter	1	2	3	4	5	6	7
ሀ	he	hu	hi	ha	hE	h	ho
ለ	le	lu	li	la	lE	l	lo
ሐ	He	Hu	Hi	Ha	HE	H	Ho
መ	me	mu	mi	ma	mE	m	mo
ሠ	s2e	s2u	s2i	s2a	s2E	s2	so
ረ	re	ru	ri	ra	rE	r	ro
ሰ	se	su	si	sa	sE	s	so
ሸ	xe	xu	xi	xa	xE	x	xo
ቀ	qe	qu	qi	qa	qE	q	qo
ቈ	que	-	qui	qua	quE	qu	quo
ቐ	Qe	Qu	Qi	Qa	QE	Q	Qo
ቘ	Que	-	Qui	Qua	QuE	Qu	Qo
በ	be	bu	bi	ba	bE	b	bo
ቨ	ve	vu	vi	va	vE	v	vo
ተ	te	tu	ti	ta	tE	t	to
ቸ	ce	cu	ci	ca	cE	c	co
ኀ	h2e	h2u	h2i	h2a	h2E	h2	h2o
ነ	ne	nu	ni	na	nE	n	no
ኘ	Ne	Nu	Ni	Na	NE	N	No
አ	e	u	i	a	ie	A	o
ከ	ke	ku	ki	ka	kE	k	ko
ኰ	kue	-	kui	kua	kuE	ku	kuo
ኸ	Ke	Ku	Ki	Ka	KE	K	Ko
ኸኰ	Kue	-	Kui	Kua	KuE	Ku	Kuo
ወ	we	wu	wi	wa	wE	w	wo
ዐ	Oe	Ou	Oi	Oa	OE	O	Oo
ዘ	ze	ze	zi	za	zE	z	zo
ዠ	Ze	Zu	Zi	Za	ZE	Z	Zo
የ	ye	yu	yi	ya	yE	y	yo
ደ	de	du	di	da	dE	d	do
ጀ	je	ju	ji	ja	jE	j	jo
ገ	ge	gu	gi	ga	gE	g	go
ጐ	gue	-	gui	gua	guE	gu	guo
ጠ	Te	Tu	Ti	Ta	TE	T	To
ጨ	Ce	Cu	Ci	Ca	CE	C	Co
ጰ	Pe	Pu	Pi	Pa	PE	P	Po
ፀ	Se	Se	Si	Sa	SE	S	So
ጸ	x2e	x2u	x2i	x2a	x2E	x2	x2o
ፈ	fe	fu	fi	fa	fE	f	fo
ፐ	pe	pu	pi	pa	pE	p	po

Appendix 2: Selected ambiguous words and their Tigrigna meanings adopted from [71].

Tigrigna ambiguous word	Senses
ከፈለ	ቆረፀ፣ ገምዐ፣ መቐለ፣ መደበ
	ዋጋ ሃበ፣ ዕዳ መለሰ
ዓረቐ	ንዝተባእሱ ሰባት ኣሰማምዐ፣ ኣሰነየ፣ ይቕረ ይቕረ ኣባሃሃለ
	ክዳውንቱ ኣረገ፣ ተቐዳደደ፣ ተወደአ፣ ዝኽደኖ ሰኣነ
ሰዓረ	ንመወዳድርቱ ወይ ንፀላኢኡ ኣሸነፈ
	ፆም ግዜ ፀባ፣ ስጋ፣ ጠስሚ ዘለዎ ምግቢ በልዐ
ሓለፈ	ከደ፣ ተንቀሳቀሰ፣ ተጉዓዘ፣ ቐደመ፣ ተፈቲኑ ተዓወተ፣ ናብ ቀፃሊ ተሳገረ፣ ደንጎየ፣ ረፈደ
	ነፍሱ ወፀት፣ ዓረፈ፣ ሞተ
	ዝያዳ፣ ብበዝሒ፣ ብዝበለፀ፣ ካብ ዝድለ ንላዕሊ፣ ካብ ልክዕ ንላዕሊ፣ ካብ ስፍሪ ንላዕሊ
	ዋሕስ፣ ተወካሊ፣ ተሓላቒ፣ ተጠያቒ፣ ሓለቓ፣ በዓል ስልጣን፣ በዓል በዚ፣
መደብ	ዝተወሰነ ቦታ ኣትሓዘ፣ በብወገኑ ከምዝሕዝ ገበረ፣ ደልደለ
	ሓደ ነገር ንምፍፃም ወሰነ፣ ቀረፀ
	መደቀሲ ወይ መቐመጢ

Appendix 3: Sample lists of Tigrigna sentences used in our corpus

1. እቶም ንጉስ ምስ ጣልያን ከይበኣሱ ክብሉ ሓደ ሬሳ ክቐብር ኢሉ ዝመፀ ሻለቃ ንተዓረቅ ክሳብ ሕዚ ዝተገበረ ኩናት ይኣክል ኢሎሞ ።
2. ብጽልእን ክርክርን ይደላለዩ ዝነብሩ ብናይ ጐይታና መድኃኒና ኢየሱስ ክርስቶስ ልደት ተዓሪቆም እዮም ።
3. ኣቦና ኣዳም ትእዛዘ እግዚኣብሔር ጥሒሱ ካብታ ኣይትብላዕ ዝተባጎለ ዕጸ በለስ ምስ በልዐ ካብ ጸጋ እግዚኣብሔር ዓረቐ ።
4. መንግስቲ ሱዳን ምስ ተቓወምቲ ምብራቕ ሱዳን ዝነበሮም ጎንፂ መንግስትና ተዓዊቱ ዓሪቕዎም እዩ።
5. ንመንግስቲ ሱዳን ምስ ናይ ምብራቕ ተቓወምቶም ገይረዮ ዝብሎ ዕርቂ ምስ ሱዳን ንባዕሉ እዩ ታዓሪቒ ።
6. ዓረና ክልል እምበር ብሄር ኣይውክልን ምስ ደምበ ትምክሕቲ ተዓሪቕ ኣብ ጎቦን ሩባን ዝተመሰረተ እዩ ።
7. ብተወሳኺ ዝዓረቕ ከባቢታት ብዝግባእ ሓውዮም ሕብረተሰብ ተረባሕቲ ልምዓት ክኾኑ ከም ዝኽኣሉ እውን ኣብቲ እዋን ዘተ ተገሊፁ ኣሎ ።
8. ሓረስቶት ወረዳ እምባ ኣላጀ ኣብ ስራሕቲ ተፋሰስን ኣብ ምሕዋይ ዝዓረቀ መሬት ተዋፊሮም ልዕሊ 21 ሚልዮን ፈልሲ ከም ዝተኽሉ ተገሊፁ ።
9. ኣብ ከይዲ ምእላይ ጎንፂ ብዓረቅቲ ሽማግለ ፍታሕ ክረክብ ኣብ ምግባር እውን ኣካል እቲ ዘተ እዩ ነይሩ ።
10. ብሰብ ሰራሕን ብመርዛማት ቁምብላታትን ዓሪቐምን ባዲሞምን ዝነበሩ ንቦታት ኣኽራናት ሽንጥሮታትን ጉሕምታትን ኣብዚ ሕዚ እዋን ሓምለዋይ ለቢሶም ይርኣዩ ኣልዉ ።
11. ዝገርም እዩ ንፀሐይን ከዋክብትን ብርሃን ኣልቢሱ ዝፈጠረ ኣምላኽ ንሱ ዓረቐ ።
12. ብሕማቐ ድሌቱን ስሚዒቱን ተመሪሑ ንትእዛዝ ኣምላኽ ምስ ጠለመ መቐጻዕቲ ናይ ጥልመቱ ድማ ዓረቐ፥ ተጨነቐ፥ ተሰዲዱ ሞተ።
13. ውልቃውን ጉጅላውን ሃብቲ ምኽዕባት ደኣምበር፡ ህዝቢ ጠመየ ዓረቐ ጉዳዩ ኣይኮነን ።

14. ብሓፈሻ ነቲ ጠምየ ኣብሊዕኩምኒ፡ ጸሚኤ ኣስቲኩምኒ ፡ ጋሻ ኾይነ መጺኤ ተቐቢልኩምኒ ፡ ዓሪቐ ከዲንኩምኒ፡ ሓሚመ በጺሕኩምኒ፡ ዝብል እዩ ለበዋ ጐይታና ኢየሱስ ክርስቶስ ።
15. ኣብየት ሓሶት ፣ ኣታ ኩሉ ምስ ሓለፈ ታሪኽ ክትጠዋዊ ፣ ንብሩ ኣስታት ግርም ገንዘብ መሰብሰቢ መጽሓፍ ሓቲምካ ኣለካ ።
16. እቲ ዝገርም ካዓ ድሕሪ ክንደይ ዓመታት ምስሓለፈ እውን እንተኾነ ኣተዓባብያ ዘውረሶም ከምዚ ከማኹም ኣክረርትን ብጽልእን ሕስድናን ዝዓወሩ ምህላዎም እዩ ።
17. እዋን ኽራማት ምስ ሓለፈ ግቡእ ምዕራይ ክግበር እዩ ።
18. እቲ ገበን ምስ ተፈጸመ ብቅልጡፍ ናብ ፖሊስ ምምልካት እቲ ዝበለጸ እኳ እንተኾነ ዋላ ግዜ ምስ ሓለፈ እውን ናብ ፖሊስ ከተመልክት ትኽእል ኢኻ ።
19. ጽልዋ ስነጥበብ ኣብ ሓደ ሕብረተሰብ ፡ ኣቃሊልካ ዝረአ ከም ዘይኮነ መባኢታዊ መረዳኢታን ፍልጠትን ካብ ዝኸውን ውሑድ ግዜ ኣይሓለፈን ።
20. ኢትዮጵያ ምህርቲ በርበረ ኣብ ዕዳጋ ዓለም ንክፍለጥ ኣብ ዝሓለፈ ዓመት ኣብ ጀርመን ኣብ ዝተኻየደ ኤግዚብሽን ተሳቲፋ እያ።
21. ንሕና ህይወት ካብ ምድሓን ዝሓለፈ ካሊእ ዕላማ የብልናን ክብሉ ተማጉቶም እዮም ።
22. ካብ ስራሕ ቐልዑ ሓሊፈን መተካእታ መንግስቲ ክኾና ከምዝኽእላ ዘረድእ ስራሕ ዘየበርከታ እናሰርሓ እይኮናን።
23. ተሞክሮ መሰረት ብምግባር ክብሪ ወሲኽን ናብ ዕዳጋ ዓለም ከቕርባን ካብ ባዕለን ሓሊፈን ሃገርና ተጠቃሚት ንምግባር መሪሕ ኮይነን ክነጥፋ ከም ዘለወን ኣገንዚበን ።
24. እቲ ምስሕላብ መዕለቢ እንተዘይረኺቡ፣ ሓደጋኡ ካብቲ ዞባ ሓሊፉ ንዓለም-ለኸ ሰላምን ህድኣትን ከተዓናቅፍ ከምዝኽእል ተሓቢሩ ።
25. ተግባር ተኮር ትምህርቲ ንክበፅሕ ምክትታል ካብ ዝጀምሩ ካብ ምንባብን ምፅሓፍን ሓሊፎም ኣብ ናብርኦም ለውጢ የምፅኡ ምህላዎም ገሊፆም ።
26. ካብዚ ሓሊፉ ኣብታ ሃገር ዝኾነ ዓይነት ተቃውሞን ዕግርግርን ኣብ ልዕሊ ዝፈጥሩ ዜጋታት ግብፂ ሰራዎት ግብፂ ተሪር ስጉምቲ ከም ዝወስድ እቶም ኣዛዚ ሰራዊት ኣጠንቂቆም ።
27. ኣብ ዝተኸሰተ በረድ ዝሓወሰ ዝናብ ናይ ክልተ መዋእላ ህፃናት መምሃሪ ክፍሊ ቆርቆሮ ካብ ጥቅሚ ወፃኢ ከም ዝገበረን ካብዚ ሓሊፉ ግን ኣብ ሰብን እንስሳን ዝበፅሐ ጉድኣት የለን ።
28. ካብዚ ሓሊፉ እውን ገንዘባዊ ሓገዝ ብምግባር እተን ሃገራት ዘለወን ሕፅረት ሓይሊ ኤሌትሪክ ንምቅራፍ ቻይና ኩለ መዳያዊ ሓገዝ ከም ዝትገብር ገሊፃ ።
29. ማሕበር ልምዓት ትግራይ ፀገማት ትምህርቲ፣ ጥዕናን ዝኣመሰሉን ኣብ ምፍታሕ ትገብሮ ዘላ ፃዕሪ ካብ ሃገርና ሓሊፉ ብደረጃ ኣፍሪካ እውን ብመርኣያነቱ ከጥቀስ ዝኽእል ተግባራዊ ስራሕ እዩ ።
30. ጅግና እንተሓለፈ ታሪኹ ግን ተራፊ ኢዩ ።
31. ምንም ኣበር ዘይነበሮ እቲ ሰገን እውን ሂወቱ ሓሊፉ ።
32. ሕሉፋትስ ሓንሳብ ስለዝሓለፉ ክልተ ግዘ ዘዐዝብ ዓይኒ ኣይረኽቡን ።
33. ሓላፊ ቢሮ ሃፍቲ ማይ ክልል ትግራይ ካብ ኣባላት ቤት ምኽሪ ንዝተልዓሉ ሕቶታት ምላሽ እንትህቡ ኣብ ህንፀት ፕሮጀክታት መስተ ማይን መስኖን ዘለው ፀገማት ንምፍታሕ ፃዕሪ ይግበር ኣሎ ኢሎም ።
34. ሓላፊ ቤት ፅሕፈት ፀጥታን ምምሕዳርን ክፍለ ከተማ ሓውልቲ ከም ዝበለዎ ኣብ ዝሓለፈ ዓመት ዝነበሩ ድኹምን ጥንኩር ጎንታት ብምንፃር ኣብዚ ዓመት ዝሐሹ ስራሕቲ ፀጥታን ሰላምን ከሰርሑ እዮም ኢሎም ።

35. ምክትል ሓላፊ ቤት ፅሕፈት ማይ ማኣድንን ኢነርጅን ወረዳ ዓድዋ ዝገለፅዎ ኣብታ ወረዳ ሴሳ ኣብ ዝተብሃለ ከባቢ ኣብ ዝሓለፈ ዓመት ዝተጀመረ ግድብ መስኖ 450 ሄክታር መሬት ናይ ምልማዕ ዓቅሚ ዘለዎ እዩ ።
36. ኣብ ዝሓለፈ በጀት ዓመት ሰራሑም ብኣግባቡ ዘይተዋፅኡ 4 ኣባላት ካብኔ ከምእተሸጋሸጉን ካልኦት 3 ኣመራርሓ ድማ ብምኽንያት ሕማም ካብ ዝነበሮም ሓላፍነት ክልዓሉ ከምዝተገበረ ተገሊፁሎ ።
37. እዚ ከቢድ መስዋእቲ እዙይ ብዘይ ኤርትራ ብኩለን ኣብ 13 ክፍለ ሃገራት ዝነብሩ ህዝብታት ብዘይ ምንም ኣፈላላይ ዝተኸፈለ መስዋእቲ እዩ።
38. ምንባርን ዘይምንባርን ትርጉም ኣብ ዘይፈላለይሉ ካብ ዝነበረ ስርዓት ንምልቓቕ እቲ ውግእ ዝጠየቕ መስዋእቲ ኩሉ እንተይሰስዐ ኸፊሉ እዩ።
39. ናይ ትግራይ ህዝቢ ብዝኸፈሎ መስዋእቲ ጀብሃ ጠቕሊሉ ናብ ሱዳን ክኣቱ ተገይሩ እዩ።
40. ሓርበኛታት ኢና ዝብሉ ተሓቢኦም ናብ ምምሕዳር መግዛእቲ ምስጢራት ክንህበኩም ክንደይ ትኸፍሉና ኢሎም ብኣምሓርኛ ንዝጽሓፍዎ ናይ ጥልመት ደብዳበታት ናብ ቋንቋ ጥልያን ከም ዝተርጎምኩ ይዝክር።
41. ኣብ ሶማልያን መላእ እቲ ዞባን ሰላምን ህድኣትን መታን ክነግስ ደምና እናገበርና፣ መስዋእትነት እናኸፈልና ቅኑዕ ዘበል ክንገብር ኣለና።
42. ሙሴ ኣነ ብሓይሊ ናትካ ባሕሪ እንተኸፈልኩ፣ ጸላእቲ እንተሳዓርኩ፣ መና ካብ ሰማያት እንተ ኣውረድኩ፣ ንዓኻ ንኣምላኽይን ፈጣርየይን ከመይ ገይሮም ሙሴ ይብሉኻ፣ ናይ ሙሴ ኣምላኽ ይበሉኻ እምበር ኢሉ መስከረ።
43. ተምሃሮን መምህራንን ኣብ ተግባር ምህዞታት ስራሕቲ ፍሉይ ቆላሕታ ሂቦም ከሰርሑን ተሞክሮኦም ንኻልኦት ከካፍሉ ይግባእ ኢሎም ኣለው ።
44. ብቐፃልነት ድማ ካብ ድኽነት ንምውፃእ ኣብ ዝግበር ተሪር ጉዕዞ ተረባሒ ሕብረተሰብ ብኣተሓሳስባ ተለዓዒሉ መደባት ማረትን መንግስትን ንምፍፃም ድግድጊት ከዓጥቕ ይግባእ ንብል ።
45. ሰማያዊ እቦና ካብ ዘለዓለም ንእና ከድሕን ኢሉ ወዱ ከምዝሰዋዕ መደቡ ነይሩ እዩ ።
46. እንግሊዝ ንረብሓታታ ክትብል ፡ ንኤርትራ እንተተካኢሉ ኣብ ክልተ ወይ ከኣ ኣብ ሰለስተ ክመቓቕላ ሓደ ሰራም ዝኹነ መደብ ኣዉጺኣ ብንጥፈት ክትንቃሳቀስ ጀመረት ።
47. ብኣተሐሳስባኦም ኤርትራ እንቋዕ ድአ በቲ ሰራም ናይ እንግሊዝ መቃቃሊ መደብ ኣይተኸፋፈለት እምበር ፣ መጻኢ ዕድላስ ደሓን ብደቃ ይውሰን ዝብል ናይ ነዊሕ ብሩህ ተስፋን ዝጠመቱ ይመስሉ ።
48. ንመስኖ ዜውዕል ማይ ከይባኽን ካብ ዒላ ናብ ሕድሕድ መደብ ተኽሊ ብትሕቲ መሬት ማይ ዝወስድ ትቦ ዘርጊሑ ማይ በቐጠባ ይጥቀም ::
49. ኣብቲ ተጀሚሩ ዘሎ ካልኣይ ትልሚ ዕብየትን ስግግርን ድማ እቲ ተጀሚሩ ዘሎ ሓዱሽ ስራሓቲ መደባዊ ዛላ ኣጠናኺሮም ብምቕፃል መነባብሮም ከማሕይሹ ኣለዎም ።
50. እዚ መንእሰይ ኮነ ካልኦት ዝሳተፉ ዘለዉ ድማ ምስቶም መሓዝቱ ብምትሕብባር ነቲ ብመንግስትን ህዝብን ለሚዑ ዝተውሃቦም መደባዊ ዛላ ብዝግባእ ናብ ረብሓ ምውዓል ስለዝኸኣሉ ህይወቶም ካብ መሰረቱ ምልዋጥ ከምዝኸኣሉ እሙን እዩ ።
51. ወርሒ ንመመደብ ግዝያት ገበረ፣ ጸሓይ ንምዕራባ ትፈልጥ ።
52. ከምዚ ዚብል ቃል እግዚኣብሄር ናባይ መጸ እሞ፡ኣብ ከርሲ ኣዬኻ ኸይሀነጽኩኻ ፈለጥኩኻ፡ ካብ ማሕጸን ከይወጻእካ ኸኣ ቀደስኩኻ፡ ነህዛብ ነብዩ ኽትትከውን ድማ መደብኩኻ፡ በለ ።
53. እቲ ልዑል ንህዝብታት ምስ ኣረስተዮም፣ ንደቂ ኣዳም ምስ ፈላለዮም፣ ከም ቀኒጽሪ ደቂ እስራኤል ዶባት ህዝብታት መደበ ።
54. መደብ ልምዓት እቶት ብብርኪ ወረዳ ዝግባእ ትኹረት ስለ ዘይረኸበ ፀጋታት ግብሪ ፀንቂቕካ ዘይምጥቃም ኣሎ ።

55. ብተወሳኺ ህድሞ ዝነበረ መንበሪ ገዝአን ናብ ቆርቆሮ ዝተኸደነ ክልተ ክፍሊ ፣ መደብ ዝነበረ መደቀሲአን እውን ናብ ሓደ ዓራትን ክልተ ፍራሽን ከምዘሰገረኦ የረድኣ ።
56. እማሆይ ወዲቀን እግረን ምስተሰበራ ኣብ መደብ ንክልተ ሰሙን ደቂሰን ።።
57. እቶም ቆልዑ ተኣኪቦም መደቀሲ ዝኾኖም መደብ ሰሪሖም ።።
58. ኣብ ገጠር ከም ዓራት ዝጥቀምሉ መደቀሲ መደብ ይብሃል ።።
59. ኣብ ገዛካ ንመደቀሲ ዝከዉን ዓራት እንተዘይብልካ መደብ ሰሪሕካ ምድቃስ ይካኣል እዩ ።።
60. ናይ ገጠር ወለዲ ኣብ ዝባን እምኒ ጭቃ ብምሕዋስ መደብ ዝብሃል መደቀሲ ይሰርሓ ።።
61. ዝጠነሰት ሰበይቲ ክትሓርስ ከላ ኣብ ክንዲ ዓራት መደብ ትትቀም ።።
62. ንመደቀሲ ዝዉዕል መደብ ከከም መጠኑ ይፈላለ እዩ ።።
63. ባሻይ ዝበሃል መዓርግ ዝዋሃቦ ኣንበሳ ሃዲኑ ዝቐተለ ወይ እውን ኣብ ኩናት ንፀላኢ ዝሳዓረ እዩ።።
64. ግራኝ ኣሕመድ ምስተልዓሉ ተጋሩ ኸይዶም ኣብ ሸምብራ ቆሬ ብምስዓሮም ናይ ግራኝ ሓይሊ ንኢትዮድያ ክሕዛ ክኢሉ።
65. ነቶም ምስኡ ዝነበሩ ነገስታትን ስዒሩ ክምለስ ከሎ ፡ ንጉስ ሶዶም ክቐበሎ ናብ ለሰ ሻዌ ፡ንሱ ኸኣ ለሰ ንጉስ ፡ ወጸ ።
66. ጐይታናን ኣምላኽናን መድሓኒናን ኢየሱስ ክርስቶስ መጺኡ ዉን ነቲ ዝገዝኣና ሞት ስዒሩ፣ ሓራ ኣዉጽኣና።

Appendix 4: Sample list of stop words removed from our corpus

ኣብ	ኣይኮነን	ኣነ	ኣበይ	‘ውን	እየን	እዮም
እቲ	እታ	እዩ	’ዩ	እና	እቶም	እሞ
ኣላ	እዙይ እውን	ኣሎ	ኢና	ኣብዚ	ኣለዉ	እዚ
እያ	ኢዩ	እያተን	እያቶም	እውን	እዙይ	እምበር
እንተኮነውን	እንተኮነ 'ውን	እንተኮነ እውን	ኢዮም	ካዓ	ካብቶም	ኮይነ
ኮይኑ	እን	ተኮነግን	እንተኮነግና	እንተይኮነስ	እስኪ	እምበር
ኢሎም	እምቢ	እሺ	እንደገና	ኢና	ኢሉ	ኢልና
ኢለን	ኢልክን	ኢልኩም	ኢለ	ኢላተን	ኢላትኩም	እንተሎ
እንተኮንኩም	እንተኮይኖም	እንተኮይንካ	እንተኮይንኪ	እንተኮይ	እታ	እቲኣ

Appendix 5: Sample lists of Affixes removed from our corpus

Prefixes			Suffixes		
ብም	ነን	ንም	ታት	ውቲ	ውቲና
ንዝ	ብዝ	ዘይ	ኩም	ኹም	ን
ኣተ	ዘይተ	እንተዘይ	ናን	ክን	ኩምን
ከይ	እንተይተ	ዝተ	ነን	ውቶም	ውተን
ከምዘይተ	ምስ	ከምት	ውትኹም	ቶም	ኻ
እና	ክን	ከየ	ኺ	ም	ትኣም
በቢ	ኣይተ	ኣይ	ዎ	ኣም	ኣን
ኣይን	ኣይተ	እንተዝ	ያዊ	ያዊን	ያዊያን
ኣይት	ከይተ	ከም	ያውያን	ዊን	ናዮም
ከምዝ	ስለ	እንተዘይተ	ውን	ነት	ነታዊ
ስለዘይ	እንተ	እንድሕር	ትን	ታትን	ሎምን
እንድሕርዘይ	ከምእን	እንድሕርዘይተ			
እናተ	ዝተ	እንት			
እን	እንተ	እንተይ			
ስለዝ	ክት	ተተ			
ንተ	ከምዘይ				

Appendix 6: Python code for Tokenization

```
import codecs
import re
import string
tok=open('e:/Acadamic documents/MSc_file/All MSc materials in one folder/NLP/my proposal/corpus/pythoncode/tok.txt','r',encoding='utf-8')
tok1=open('e:/Acadamic documents/MSc_file/All MSc materials in one folder/NLP/my proposal/corpus/teklay/tokinezed.txt','w+', encoding='utf-8')
translator = str.maketrans('', '', string.punctuation)
for sen in tok:
    text=sen.split()
    print(text)
    tok1.write(sen)
```

Appendix 7: Python code for searching sentences containing ambiguous word

```
import codecs
import string
import re
Orginal_file=open('e:/Acadamic documents/MSc_file/All MSc materials in one folder/NLP/my
   proposal/corpus/teklay/areke.txt','r', encoding='utf-8')
Processed_File=open('e:/Acadamic documents/MSc_file/All MSc materials in one folder/NLP/my
   proposal/corpus/teklay/Processed_Dataareke.txt','w+', encoding='utf-8')
rf=Orginal_file.read()
sp=rf.split()
ck=re.findall(r"[\w']+|[.,!?;::።፤፡]",rf)
cl=re.split(r'(;|,|\s)\s*',rf)
p=0
dic={}
ls=[]
for i in cl:
      if (i.endswith('።')):
         c=i.rstrip("።")
         fwe=c+'  '+ "።"
         Processed_File.write(fwe+' ')
         Processed_File.write('\nl')
      elif (i.endswith('፡፡')):
         c=i.rstrip("፡፡")
         fwe=c+'  '+ "።"
         Processed_File.write(fwe+' ')
         Processed_File.write('\nl')
      elif (i.endswith('?')):
         c=i.rstrip("?")
         fwe=c+'  '+ "?"
         Processed_File.write(fwe+' ')
         Processed_File.write('\nl')
      elif (i.endswith('!')):
         c=i.rstrip("!")
         fwe=c+'  '+ "!"
         Processed_File.write(fwe+' ')
         Processed_File.write('\nl')
      else:
         Processed_File.write(i+' ')
cc=input("enter the word that you want to search:")
Orginal_file.close()
Processed_File.close()
```

```
SENTENCES = open('e:/Acadamic documents/MSc_file/All MSc materials in one folder/NLP/my
   proposal/corpus/teklay/Processed_Dataareke.txt','r', encoding='utf-8').readlines()
c=([sentence + '.' for sentence in SENTENCES if (cc in sentence) ])
if len(c)>=1:
    for i in c:
        print(i)
else:
    print('word not found in the dictionary thank you for using the software')
```

Appendix 8: python code for Stp word removal

```
import re
import string
import codecs
fh=open('F:/wes/news1.txt','r',encoding='utf-8').readlines()
Stop_Words=open('F:/wes/stop.txt','r',encoding='utf-8').read()
stop=open('F:/wes/stopr.txt','w+',encoding='utf-8')
#sp=fh.split()
for i in fh:
    if ((len(i) !=0)):
        c=str([k for k in i.split() if ((k !='') and (k.isdigit()==False) and k not in Stop_Words.split())])
        stop.write(c)
    else:
        pass
```

Appendix 8: Python code for removing prefixes

```
import codecs
cor=open('e:/Acadamic documents/MSc_file/All MSc materials in one folder/NLP/my
   proposal/corpus/teklay/final corpus/final cor/agree.txt','r',encoding='utf-8').read()
c=cor.split()
p=codecs.open('e:/Acadamic documents/MSc_file/All MSc materials in one folder/NLP/my
   proposal/corpus/teklay/final corpus/final cor/prefixes.txt','r', encoding = 'utf-8' )
pr = p.read()
y=pr.split()
print (y)
def preffix1(word):
   for j in y:
    if word.startswith(j) and len(word)>2:
      word=word.lstrip(j)
      break
   return word
def preffix2(word):
   if len(word)<3:
```

```
        return word
    elif len(word)>=3:
       while len(word)>=3:
          tempp=word[ :3]
          if y.__contains__(tempp):
                word=word[3: ]
                preffix2(word)
          else:
             tempp=word[ :2]
             if y.__contains__(tempp):
                word=word[2: ]
                preffix2(word)
             else:
                tempp=word[ :1]
                if y.__contains__(tempp):
                   word=word[1: ]
                   preffix2(word)
                else:
                   return word
```

Appendix 9: Sample output from weka 3.8

```
Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         124               77.5     %
Incorrectly Classified Instances        36               22.5     %
Kappa statistic                          0.5513
Mean absolute error                      0.2734
Root mean squared error                  0.3865
Relative absolute error                 57.8962 %
Root relative squared error             79.5537 %
Total Number of Instances              160

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.717     0.131       0.899     0.717      0.798      0.836    negotiate
                 0.869     0.283       0.654     0.869      0.746      0.836    poverty
Weighted Avg.    0.775     0.189       0.806     0.775      0.778      0.836

=== Confusion Matrix ===

  a  b   <-- classified as
 71 28 |  a = negotiate
  8 53 |  b = poverty
```