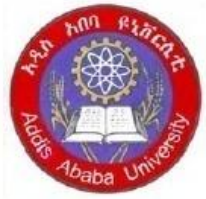


ADDIS ABABA UNIVERSITY

FACULTY OF NATURAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

# AMHARIC NAMED ENTITY RECOGNITION USING A HYBRID APPROACH

**BY: MIKIYAS TADELE BELAY**

**ADVISOR: MARTHA YIFIRU (PhD)**

A THESIS SUBMITTED TO

THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN INFORMATION SCIENCE

August, 2014

ADDIS ABABA UNIVERSITY

FACULTY OF NATURAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

# AMHARIC NAMED ENTITY RECOGNITION USING A HYBRID APPROACH

**BY: MIKIYAS TADELE BELAY**

**ADVISOR: MARTHA YIFIRU (PhD)**

APPROVED BY

EXAMINING BOARD:

1. Dr. Martha Yifiru, Advisor _____
2. _____ _____
3. _____ _____

# Declaration

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

_____

Mikiyas Tadele Belay

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

_____

Martha Yifiru (PhD)

Addis Ababa, Ethiopia
August 2014

# Acknowledgment

I would like to thank the almighty God for giving me the strength and determination to conduct my research. I would also like to thank my advisor Dr. Martha Yifiru for guiding me and also for giving me constructive criticism that helped my research go in the right direction. Special thanks to all of my family and friends for motivating and supporting me.

# Contents

# List of Tables



# List of Figures

# Abbreviations

| | |
|---|---|
| ACE | Automatic Content Extraction |
| ANER | Amharic Named Entity Recognition |
| ARFF | Attribute Relation File Format |
| CoNNL | Conference on Computational Natural Language Learning |
| CRF | Conditional Random Field |
| DARPA | Defense Advanced Research and Project Agency |
| GATE | General Architecture for Text Engineering |
| IR | Information Retrieval |
| IE | Information Extraction |
| ICT | Information Communication Technology |
| MT | Machine Translation |
| ML | Machine Learning |
| MUC | Message Understanding Conferences |
| NLP | Natural Language Processing |
| NER | Named Entity Recognition |
| NE | Named Entity |
| POS | Part of Speech |
| SVM | Support Vector Machines |
| SL | Supervised Learning |
| SSL | Semi Supervised Learning |
| WIC | Walta Information Center |
| WEKA | Waikato Environment for Knowledge Analysis |

# Abstract

Named Entity Recognition (NER) is a subcomponent of information extraction (IE) that detects and classifies named entities (NE) which, among others can be proper nouns representing person, location, and organization names and also date, time, and measurements. NER has been also found to be vital for other NLP applications, such as Information Retrieval, Question and Answering, Machine Translation, and Text summarization to mention a few.

This research reports the performance of Amharic NER (ANER) built using the hybrid approach and different feature sets to detect and classify NEs of type person, location, and organization. Two state of the art machine learning (ML) algorithms, namely decision tree and support vector machines (SVM), are used to investigate the performance of the hybrid ANER. This is the first research that has used these ML algorithms for ANER and also the first research to explore ANER using the hybrid approach.

The rule-based component of the hybrid ANER has been built using two rules that base their predictions on the presence of trigger words before and after NEs. The ML component is built using decision tree (J48) and SVM (libsvm). The hybrid ANER integrates those two components by using the NE class predicted from the rule-based component as a feature in the ML component.

We have conducted different experiments to compare the performance of the hybrid approach with that of the pure ML approach by using different feature sets. From our experiments we have obtained a high performing model for both J48 and libsvm algorithms without using the rule-based feature but using POS feature with the nominal flag feature with an F-measure of 96.1% for J48 and 85.9% for libsvm.

Based on the experimental results we have concluded that the pure ML approach with POS and nominal flag feature outperformed the hybrid approach. This is because the rule-based component used in the experiment uses only trigger words. Using rules prepared by linguists and gazetteers may improve the rule based component and consequently the hybrid ANER system.

**Keywords:** Amharic Named Entity Recognition, Information Extraction, Decision tree, Support Vector Machine, Hybrid Named Entity Recognition System.

# Chapter One

# 1. Introduction

## *1.1 Background*

Natural language processing (NLP) is a sub field of Artificial Intelligence which has the ability of computer systems to analyze and synthesize spoken and written languages as human beings. The word 'Natural' is used to distinguish those formal languages like programming languages from that of the task performed by NLP [1]. The exponential growth of computing power and also the drastic decrease to hardware cost have opened up doors to fields that required computational resources. This development has been enhanced by the Web which is a repository for massive amounts of information [2]. NLP is one of the fields that benefited from the development mentioned above. Below are some of the applications that make use of NLP to allow users interact with computer systems using natural languages:

- Information Retrieval: the ability to retrieve relevant documents from a repository of documents in response to user requests.
- Machine Translation: the ability to translate one language into another language.
- Question and Answering: the ability to give a specific answer or answers to a given question.
- Speech Recognition: recognizing spoken languages and transcribing them into written form.

Advancement in research and development of NLP applications has shown rapid progress especially for those languages which are considered to be resource rich, such as English.

Information Extraction (IE) is one application of NLP which automatically extracts structured information such as entities, relationship between entities and also attributes describing those entities from unstructured documents[13]. IE systems are effective for tackling the problem of information overload as they enable us to get the most important part of information that exists in huge documents quickly and easily.

Named Entity Recognition (NER) is the subtask of IE that detects and classifies proper names, which are called Named Entities (NEs), within unstructured texts into

predefined types such as Persons, Locations or Organizations, etc. [3]. Other than IE NLP applications such as those mentioned above make use of NER. For example, in Information retrieval, NER can be used first to recognize NE in the query and also to extract relevant documents containing this NE, in Machine translation recognizing NEs is important in order to disambiguate some words from NEs and finally in Question and answering NER plays the same role as in IR [4].

It is in 1990, at the Message Understanding Conferences (MUC) the task of NER was introduced and given attention by the community of researchers. These conferences were funded by the Defense Advance Research Project Agency (DARPA) for the purpose of developing a better information extraction systems. At the sixth MUC, three main NER subtasks were defined namely: ENAMEX (i.e. Person, Location and Organization), TIMEX (i.e. temporal expressions), and NUMEX (i.e. numerical expressions) [4].

There are three approaches for creating NER systems: rule-based, Machine learning (ML) and the hybrid approach. The rule-based approach is based on local language rules which are handcrafted whereas the ML approach uses ML algorithms to extract some features from annotated documents with NEs to build a model for predicting NEs. The hybrid approach which combines those two approaches to build a better NER system [4]. The rule-based approaches require a linguist to define all those rules and preparing the rules is time consuming. There is also lack of portability and robustness with high maintenance cost when a rule is added and outdated rule is removed. In the ML approach a large collection of annotated corpus is needed. Nevertheless it is portable, adaptable and easily maintainable [5].

NER has been a challenging task for researchers especially for languages that lack resources and also which have a more complex language structure as compared with other languages. Ambiguity is the major challenge faced by NER researchers as the same word can be used to refer different entity types. Furthermore, for morphologically rich languages like Amharic, ambiguity can be created due to the agglutinative nature of the language. Ambiguity may also result from spelling variations.

## *1.2 Statement of the problem*

Amharic is a Semitic language which has at least 21 million speakers as a mother tongue and others with different mother tongues in different parts of Ethiopia. It is also the official working language of the federal democratic republic of Ethiopia [7].

Ever since the Ethiopian government has made the development of ICT one of its strategic priorities [50] the amount of documents which are stored electronically has shown drastic progress and the use of computers to do everyday tasks. NLP applications especially those that mitigate information overload like question and answering, IR, and IE will be crucial to effectively and efficiently use information in stored documents that are found in abundance.

NER is one of the subcomponents of IE which recognizes and classifies NEs from unstructured collections of documents which plays a major role to making the IE systems that uses it more robust and reliable. Even though NER is a subcomponent of IE, other NLP applications have benefited from NER systems by integrating it with their systems. Question and answering systems uses NER for identifying NEs in questions so that they can return accurate answers [14]. IR systems also uses NER to retrieve relevant documents [40]. Machine Translation systems also benefited from a robust and reliable NER system [2]. Because of its importance in the various NLP applications, researches are conducted in the area of NER globally.

In Ethiopia, the work done by [6] in NER is the first one for the Amharic language. He used a ML approach specifically the CRF ML algorithm and the corpus he prepared himself from the Walta Information Center (WIC)[53] which is tagged with part of speech tags (POS). He used the IOB, which tags each NE type as inside, outside, and beginning ,encoding to tag tokens into their perspective NEs types and considered only generic named entity types which include Persons, Location and Organization. Considering that his work was the first attempt for the Amharic language the result he came up with is encouraging.

The other work is done by [9] who recently tried to explore the optimal feature sets that will increase the performance of the NER system using the same corpus and algorithm which have been used by [6] and achieved a maximum performance of F-

measure 80.66% which is better than that of [6] that registered a maximum performance of F-measure 74.61% percent.

The above two works done for Amharic named entity recognition(ANER) have only explored one ML algorithm namely CRF but as stated in [4] Decision Tree and Support Vector Machines (SVM) have registered higher performance in general NER systems. Moreover, both [9] and [6] have put in their feature work that NER built using the hybrid approach has the potential for having a better performing NER systems. Furthermore, as stated in [4] the hybrid approach is the optimal approach which has given higher performances for Semitic language family like Arabic. Thus this thesis aimed at exploring the hybrid approach for development of Amharic NER system. Moreover, the study will investigate the use of two ML algorithms that are not used for the development of ANER system in the previous studies.

The study will therefore answer the following research questions:

- Is it possible to improve the performance of ANER by applying a hybrid approach and by using machine learning algorithms that are not used in previous studies?
- What are the appropriate feature sets that will allow the machine learning component to perform better?

## *1.3. Objectives*

The objective of the research can be classified into two: general and specific objectives.

### 1.3.1 General Objective

The general objective of this research is to investigate the performance of the hybrid approach to ANER using SVM and Decision trees ML algorithms.

### 1.3.2 Specific Objectives

- Reviewing various literatures written in the area of NEs, especially in those languages which share common issues and features with that of Amharic.
- Study the structures and features of Amharic language related to NEs
- Building a Rule-based component for identifying NEs from a collection of Amharic documents.

- Develop feature extraction algorithm for the Machine Learning algorithm and integrate the rule-based system prediction with the set of extracted features.
- Build a hybrid NER model using the data prepared.
- Evaluate and test the NER models
- Make conclusion and give recommendations.

## *1.4. Scope and Limitation*

The task of NER as it is defined by MUC in 1990 [4] involves subtasks in which those subtasks in turn can be further broken down into other subtasks and so on. The major tasks which are defined in the MUC conference are: ENAMEX, which includes Persons, Location and organization, TIMEX, which are temporal expressions and NUMEX, which are numeric expressions.

In this study we will consider only ENAMEX, also called generic NEs, to classify a given word into its respective NE types. As stated by [6] and [9], and [4] features plays a major role in classifying words to NE types. In this research we have only used two new features the rest are features that [6] and [9] used for their research to get the maximum performance for their systems. But exploring with different feature sets has the potential to obtain a good performing ANER system.

Building a hybrid approach involves having a robust and reliable rule-based component which includes an exhaustive set of rules handcrafted by linguists. Furthermore, having large collections of lists, which includes gazetteers and trigger words is vital when building rule-based components. The rule-based component that we have built for our system can be considered as simple since it only uses a small list of trigger words for identifying NE types as building a robust rule-based component is time consuming and requires linguistic expertise.

Finally we have only tested the hybrid approach using the rule-based prediction as a feature that will be used with the machine learning algorithms. But the rules' prediction can be used after the machine learning model classified NEs to disambiguate some of the words wrongly classified by the model.

## *1.6. Application of Results*

As mentioned in previous sections the NER is the most important component for various NLP-based applications. But before other NLP systems realized the importance of incorporating NER it was used only as a subcomponent of IE. In IE extracting NEs from unstructured text is the first step as NEs gives answers to who, where, and when questions in a particular document. And those answers given by NEs are used to find out who relates to who, when and where a particular event happens and more. So we can say that having a better NER system is like having a better IE system.

But recently most NLP applications especially those related to information extraction tasks like question and answering, information retrieval, text summarization and sentiment analysis have benefited from having better performing NER systems[40][14][41]. And finally Machine translation systems have also benefited in recognizing named entities for disambiguating NEs from other words similar to NEs.

## *1.7 Thesis Organization*

The thesis is organized as follows. Chapter one presented introduction to the research area of NER with some background information. Then it talked about the research's problem statement, specific and general objectives, scope and limitation, and finally application areas.

Chapter two reviews literatures which start by discussing Information Extraction(IE) ,which Named Entity Recognition (NER) is one of its component. Then it proceeds discussion to NER including the approaches used, the architecture and evaluation techniques. And discusses Machine Learning(ML) algorithms. This chapter also covers related works by differentiating the works based on the approach they have used to solve their problems.

Chapter three illustrates the methodology that is chosen to conduct this research. The chapter discusses the method used, the way data are collected and prepared, and finally the tools used to build and evaluate this research.

Chapter four is on design and implementation of the research. This chapter is divided into two parts. The first part discusses the design of the system which includes what approach used and the architecture of the system. It then proceeds to the next part,

which is the implementation of the system, and discusses rule and feature extraction implementation.

Chapter five discusses in detail about the experiments that have been conducted. It covers the environmental set up used for conducting the experiments and also results obtained from the experiment. Finally the chapter provides discussion and analysis of the results obtained from the different experiments.

Conclusions on the basis of the experiment and results are given in chapter six. This chapter also provides recommendation for future work in the area of Amharic NER.

# Chapter Two
# 2. Literature Review

This chapter covers concepts which are vital to the understanding of Named Entity Recognition(NER). It starts with a brief introduction to Information Extraction(IE) and continues discussing about IE components among which is NER is one. Then the chapter goes on discussing about NER, which includes its architecture, the approaches used, feature sets, and evaluation metrics. Finally it discuss the two Machine Learning(ML) algorithms we have used for our experiment, namely Decision Tree and Support Vector Machines (SVMs).

## *2.1 Information Extraction*

The astronomical growth of information which are available in electronic format has been intensified by the internet, which has now become a repository for Global Knowledge. Information overload, which is a byproduct of the ability to manipulate, process, and store data electronically with ease, has been a major problem. One solution that is widely used even today to tackle the information overload problem is information retrieval which is a sub-filed of Information Science. According to [10] IR responds to user query by enlisting documents which are assumed to be relevant to the given query in a relevance order. It lets the user judge which documents returned are relevant or not. Users are further assisted by the IR system by highlighting keywords contained in the query in the retrieved documents. Even though today's IR systems provides many features that allows users to find relevant documents as [10] most IR systems are not oriented toward handling natural language texts besides determining relevance of document by calculating the relative frequency of words occurring in the corpus and query.

Another technology that has been used to tackle information overload is IE, which extracts facts that exist within a natural text and enable us to store those facts in a structured form for further manipulation and analysis[2][12] . One definition of IE that is put quite vividly and concisely along with some of the process involved is this definition by [13] " Information Extraction starts with a collection of texts, then transforms them

into information that is more readily digested and analyzed. It isolates relevant text fragments, extracts relevant information from the fragments, and then pieces together the targeted information in a coherent framework ".

Historically it was the MUC that was sponsored by DARPA in the 1980's that contributed a lot to the popularization of Information Extraction as it is the one that defined the tasks involved in IE and also the evaluation of IE systems using standard evaluation metrics. The following section provides a description of those tasks that are defined by the MUC .

### 2.1.1 Information Extraction Tasks

[11]The MUC have laid out tasks which are considered as generic tasks that allow the evaluation of different IE systems , so that those systems are evaluated based on the performance of the tasks set. The tasks also allow for consistent and reliable evaluation of IE systems. Below are description of those tasks based on [2][12]

1. **Named Entity Recognition:** The first step in most IE systems is the detection and classification of named entities , which are proper nouns, in a natural text. Depending on the domain in which the application of NER is used named entities can be referred to many things for instance in most generic news-oriented NER systems named entity types refer to places, persons, and organizations. Some applications may require the identification of other entity types , including products, proteins, genes , weapons and others. Since our research focuses on the tasks related to NER a detailed description of this task will be provided later.

2. **Relation Detection and Classification:** After detecting and classifying the named entity types the next step in IE is to again detect and classify the relations that exists between those entity types. [2] Categorizes those generic relations that exist between different entities. The first one is; Affiliation, which relates person or organizations to one another for example married to, spokesman for and alike. The second one is geospatial that relates locations to one another for example near and on the outskirts. Finally the part of relation which relates part of something like a unit of and parent of. Most relations that exists in domain-specific applications are instances of the above relation types.

**3. Temporal and Event Processing :** After detecting and classifying the different named entity types and the relations that exists between them the next step for any IE system is finding and analyzing the events in which those entities take part and how those events relate to time are important for comprehensive extraction of information from a given text.

**4. Template Filling:** The task of template filling is to find documents that invoke particular scripts, which consists of prototypical sequences of sub events, participants, roles, and properties, and followed by filling the slots in the associated templates with fillers extracted directly from the text. The fillers may consist named entities or text segments extracted from a text.

## *2.2 Named Entity Recognition*

As it was stated in section 2.1.1 Named Entity Recognition(NER) is the first task during the extraction of information. According to [14] the term for named entity as defined by most researchers and in conferences can be classified into four categories: named entity as proper names ,which is serving something or someone as a name. The second one is names used as rigid designators that refer only a given concept almost universally. The third category is named entities as a unique identifier that is the concept they refer to is unique within a well defined context. The last category of the definition of named entities is based on the purpose and domain of applications. According to[14] named entities should be defined this way as it is consistent with literatures, forums, and tools widely used. So for the our purpose we stick to the definition given by [MUC-7] " named entities are "unique identifiers " of entities (persons, locations, organizations), times (dates, times, and durations), and quantities (money, measures, percents, and cardinal numbers)".

Named Entity Recognition can be defined as the detection and classification of named entities into their proper types. The types are defined by the three leading bodies namely MUC, CoNLL, and ACE, in which they have dominated the evaluation and also have been defining the different tasks associated with NER. For the purpose of this work we have selected those named entity tasks as defined in the MUC-7 conference they are ENAMEX, TIMEX, and NUMEX. ENAMEX refers to persons, organization and location names, where as TIMEX refers to time and date and finally NUMEX refers to

monetary values and percentages. This work is limited only the detection and classification of ENAMEX.

Even though NER is one of the main component of information extraction during the extraction of different properties of a given unstructured text NER is more than Just a component of IE. The applicability of NER to other NLP application have also attracted attention of researchers who have been working on Machine Translation, Speech Recognition, Information Retrieval, Question and Answering to mention just a few. Following are a brief description of some of the application areas of NER is provided.

1) **Information Retrieval(IR) :** IR involves the retrieval of relevant documents in response to a given query.[4] Describes how NER assists IR in fulfilling its objectives in two ways; the first one is on detecting and classifying NEs within the query and the second one is on detecting and classifying NEs within the documents so that the NE identified in the query is matched with those types detected within the documents . [40] have conducted an experiment over periods of several days in 1995. Their work showed that 67.8%, 83.4%, and 38.8% of queries to Wall Street Journal, Los Angeles Times, and Washington Post, respectively, involve name searching.

2) **Machine Translation(MT):** [2] defines MT as the use of computers to automate some or all of the process of translating from one language to another. During the translation of a given word in some language into another detecting and classifying NEs into their proper types will help disambiguate some words with similar surface forms as of the NEs.

3) **Question and Answering :** Question and Answering systems are almost similar to that of IR systems but the main difference is that the answers that are given are concrete, almost exact. As stated by [14] in TREC (text retrieval conferences) it was reported that 80% of the questions asked relate to types of who, when, where, and the answers correspond to named entities of type person, time, and location.

4) **Text Summarization:** Text summarization involves reducing a single or multiple documents content into a summary which fully describes what the text is all about. In his paper[41] talked about doing text summarization by way of text extraction which extracts some useful pieces of terms using statistical or heuristic approach. By combining these

two approaches into a single coherent summary. He also proved that topic identification is the most important task in text extraction. He reported that most of the time topics are named entities and proper detection and classification of these entities plays a major role in the identification of topics. Finally he stated that NER has improved the performance of their text summarization.

### 2.2.1 NER Architectures

According to[2] most of the NER tools which are commercially available are based on a combination of lists (i.e. gazetteers),rules and supervised machine learning. The common approach that [2] mentioned is to iteratively pass over a given text sequentially which would allow the result of one iteration to influence the next step. According to[2] most practical NER architectures incorporate the following steps:

1. Using rules to tag unambiguous entity mentions.
2. Search for substring matches of previously detected names using probabilistic string matching.
3. Search those name lists which are application specific to identify names within a given domain.
4. Use machine learning algorithms that uses those tags from previous steps as an additional feature to learn and classify named entities.

The reasons for the above sequence of steps as described by [2] is twofold . First some entity mentions are more clear indicative of a given entity's class than others. Second, once those entity mentions which are considered to be unambiguous are identified it's likely that shortened versions of those entities will refer to the same entity class or type.

But a more detailed Architecture of NER is given by [8] which generically applies to whichever approaches are pursued for NER. According to[8] there are four core component for any NER tasks as depicted in Fig 1. They are:

**1) Tokenization :** that is splitting a string of words or characters comprising a document, paragraph or sentence into meaningful units.

**2) Morphological and Lexical Processing:** During this step the token identified previously is processed further like for instance for associating features like POS tag or boundary of the token whether its inside or outside of the Named entity and so on.

**3) Identification:** This step involves the detection of named entities using either the learning models or predefined rules.

**4) Classification:** After the identification of the Named entities this step will classify those detected named entities into their proper types.

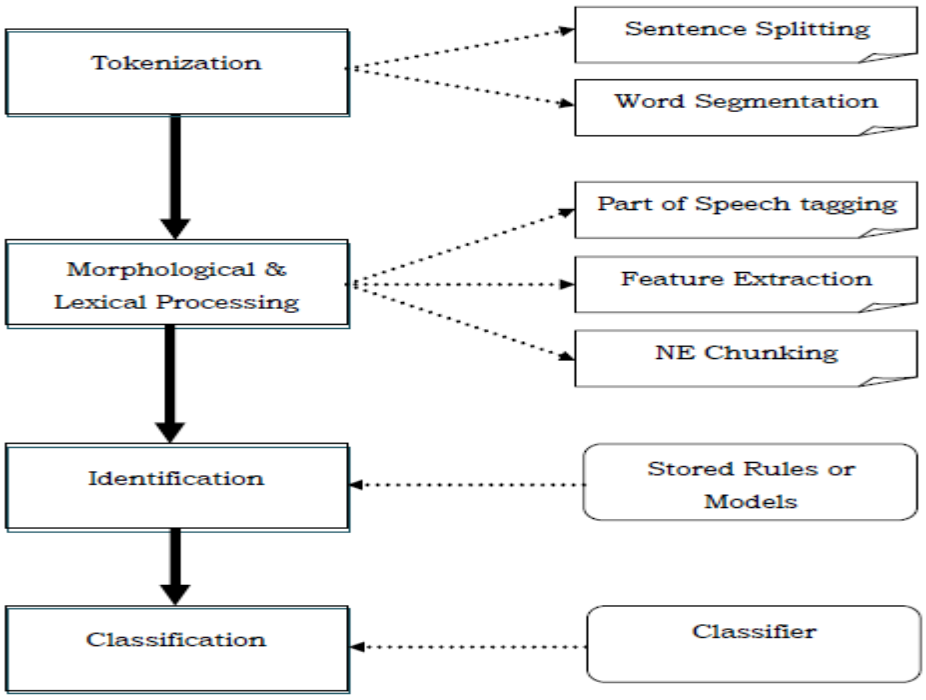


**Figure 2.1 Generic NER Architecture**

## 2.2.2 NER Approaches

As in most Natural Language Processing( NLP )systems the automatic extraction and classification of named entity systems are classified according to the approach they have used [16]. They are :Rule-based, Machine-Learning and Hybrid.

### 2.2.2.1 Rule-Based Approach

This approach uses a set of rules written and defined manually by linguists [43]. The systems consist of a set of patterns using grammatical (e.g. part of speech), syntactic (e.g. word precedence) and orthographic features (e.g. capitalization) in combination with dictionaries [16]. And [43] states that systems with rule-based approach performs better

than any other approaches for domain specific tasks. But the problem with this approach is that manual creation of rules is tiresome, expensive, and almost impossible to enlist all the rules that are required for identification and classification purposes. But once all the required rules are defined the system can efficiently detect and classify named entities. Since the approach is language and domain specific the system is not able to be ported to another language or domain.

#### 2.2.2.2 Machine- Learning Approach.

In this approach the systems look for patterns and relationships in the text to make a model using statistical models and machine learning algorithms. The systems identify and classify nouns into particular classes such as persons, locations, times, etc based on the model built, using machine learning algorithms[16] . A great advantage of ML approach is its ability to be portable and maintainable with ease as compared with the rule-based approach but still it requires a huge corpus in order to build a high performing NER systems. This approach can be classified further in to three: supervised learning, unsupervised learning and semi-supervised learning.

**a) Supervised Learning(SL):** The current dominant approach in solving the NER problem is using the supervised machine learning algorithms. They are called supervised because they learn from a well prepared labeled data which indicate them what to predict[25].This approach requires the preparation of annotated training data in order to build a robust learning model but as stated by[16] it cannot achieve high performance if the training data is small a data sparseness creates problem. Some of the popular supervised ML algorithms are; Hidden Markov, Maximum Entropy Markov Models, Support vector machines, Conditional random fields, decision tree, etc.

**b) Unsupervised Learning:** [16]In this approach the system is given data which are unlabeled and the goal of the algorithms are to find representation for the data. These representations can then be used for data compression, classification, decision making, and for other purposes. Unsupervised learning is not a very popular approach for NER and the systems that do use unsupervised learning are usually not completely unsupervised. Examples include Expected Maximization, K-means etc.

**c) Semi-supervised Learning(SSL):** This approach is a newly introduced approach for ML and in the domain of NER it has become close in terms of performance to that of SL

approach[43]. The main technique for SSL is called “bootstrapping” and involves a small degree of supervision, such as a set of seeds, for starting the learning process.

#### 2.2.2.3 Hybrid Approach

The hybrid approach to NER is simply a combination of the rule-based and machine learning based approaches. This approach makes a new method and is developed by taking the strong side from both rule-based and machine learning based approach. Our research also uses a hybrid approach with different machine learning algorithms to conduct an experiment for Amharic language.

If we take for example the works of [4][8] we can clearly see that the hybrid approach to NER can be applied in at least two ways. According to [4] the rule-based component of the NER is included as part of the feature space. This will give the ML component more contextual information about a given word during the learning process. The other method used is the one proposed by [8], where the rule-based component is used after the ML component has already classified a given test data. This means that [8] has used the rule-based component to post process and disambiguate the already classified data.

### 2.2.3 NER Feature Sets

Feature selection is a major component of many machine learning applications. Features are attributes or characteristics of a given word used for the consumption of algorithms[44]. For example one feature that is valuable in determining whether a given word is a proper noun or not is the word's orthographic feature that is whether it is capitalized or not. This orthographic feature is represented by a Boolean function that returns true or false. According to [44] a feature vector is an abstract representation of a given text in which each word is represented by different Boolean, Numeric and String values. [44] Further illustrates about this feature vector representation with a hypothetical example to represent a given word in the context of a NER. We shall repeat here his example in terms of the Amharic language. The three attributes are:

1. A Boolean attribute which holds true or false if the given word is beginning of a sentence or not.
2. A numeric attribute which holds the number of characters in a given word.

3. A Boolean attribute which holds true or false if the given word is a noun, noun phrase or not.

Based on the above feature sets the following sentence አለም አቀፍ የ ሩጫ ውድድር በአዲስ አበባ ሊካሄድ ነ ው ፤ can be represented in a feature vector for each word in a sentence as: <true,3,true>,<false,3,true>,<false,3,true>,<false,4,true>,<false,4,true>,<false,3,true>,<false,4,false>,<false,2,false>,<false,2,false>. From the above information the machine learning algorithms can classify words into predefined categories.

Features for NER can be broadly divided into two main categories. They are Language dependent and language independent features but first let's discuss categories of features most commonly used by researchers.

According to[44][45] NER features are most commonly categorized in to three feature types: word-level features, list look-up features, and document and corpus features .

1. **Word-Level Features:** These features are related to the making up of a given word and also its grammatical meaning. Examples include: first letter capitalized, numeric, punctuation, part of speech, morphology, and individual characters in a given word. Below are some of the most common features that we have taken from[44].

- Case:- whether the first letter of a word is capitalized, all letters are uppercased , or the word consists of mixed cases (not a useful feature for Amharic unless when transliterating it to English alphabet case rules for English word is applied.).
- Punctuation:- in Amharic whether word ends with two colons(::).
- Character:- whether special characters are used like Greek letters or mathematical symbols.
- Part of Speech:- whether the word is a noun, verb, adjective, and etc..
- Morphology: prefix and suffix morphemes, stem, and common ending word.

- Character length: number of characters in a word, number of characters of a prefix and suffix.

2. **List look-up features:** Lists which are synonymously called as gazetteers, dictionaries, and lexicon [ 45] are most important and expensive features. They are expensive, especially for NER as most Named Entities(NEs) are open classes that require frequent maintenance. As mentioned by [44] they represent an is-a relationship in that a given word is either a member of a given list or not. Most of the features are represented in a Boolean function that returns true or false. Based on [44] lists are categorized as:

- General List:-This list includes list of stop words, common abbreviations and capitalized nouns (e.g. Sunday, September and the like).
- List of Entities:- These are lists combined of the entities that the NER is recognizing. For example Person name lists, organization name lists and so on.
- List of entity cues: These lists can include trigger words found in most entities. For example አቶ ,ወይዘሮ, ፕሬዝዳንት.

3. **Document and Corpus Features:** These features cover a given word's feature over a given document or corpus. Some of the features are: multiple occurrences and multiple cases, entity co-reference and alias, and statistic for multiword limits[44].

Therefore all these specific categories of feature sets can be put into language dependent and independent. most word-level features are language dependent and in contrast list look-up features are almost language independent.

### 2.2.4 NER Evaluation

In order to compare one research with another research and also to assess the progress of a given research area , i.e. NER, evaluation is an important part of any research especially in NLP. According to [17] Most of the performance evaluation metric

for NER are borrowed from information retrieval and these metrics are recall, precision and F measure.

- Precision: The ratio of the number of correctly labeled responses to the total labeled[2].
- Recall: The ratio of the number of correctly labeled responses to the total that should have been labeled[2].
- F-measure: The weighted harmonic mean of precision and recall given by

$$\text{F-measure}=\frac{(\beta^2+1)PR}{\beta^2P+R}$$

where P and R refer to precision and recall respectively. The β parameter differentially weighs the importance of recall and precision depending on the needs of the application. When $\beta > 1$ it favors recall and when $\beta < 1$ it favors precision. precision and recall becomes equally balanced when $\beta = 1$ which is called F1 measure[2].

## *2.3 Machine Learning Algorithms*

The best definition that we have for Machine Learning(ML) is the one defined by[45] which says that "Machine Learning is programming computers to optimize a performance criterion using example data or past data". He further elaborates his definition stating that a model is defined up to some parameters and learning is the execution of computer programs to optimize the parameters of the model using training data or past experience. The model can be a predictive model or a descriptive model. The former is used to predict something new in the future while the latter is used to gain insight of the results obtained by the model. For NER the models we have built are the predictive ones.

There are many machine learning algorithm that are applicable in a wide area of applications which is difficult to list them all. But to mention some as categorized by[48] are: Decision tree learning algorithms, which includes C4.5, Classification and Regression Tree (CART), Iterative Dichotomiser 3 (ID3) and etc. The other category is Kernel Methods, which are Support Vector Machines (SVMs), Radial Basis Function (RBF) and Linear Discriminate Analysis (LDA). The last category are Bayesian Naive

Bayes, Averaged One-Dependence Estimators (AODE) and Bayesian Belief Network (BBN).

Next we will discuss briefly the ML algorithms that we have used in our experiment which are SVM and Decision tree.

### 2.3.1 Decision Tree

According to[19] A decision tree is a classifier expressed as a recursive partition of the instance space. The decision tree consists of nodes that form a rooted tree, meaning it is a directed tree with a node called "root" that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an internal or test node. All other nodes are called leaves (also known as terminal or decision nodes). In a decision tree, each internal node splits the instance space into two or more sub-spaces according to a certain discrete function of the input attributes values. Most frequently only each test considers one attribute at a time which are part of the feature space of the training set. The leafs of the tree are the classes in which a given instance of a training set is assigned. Fig 2.2 shows a tree that determines whether or not a given customer responds to direct mailing, where the leaf nodes are represented by a triangle as yes or no and the internal nodes which are attributes in a training set at a given instance represented by a solid circle.

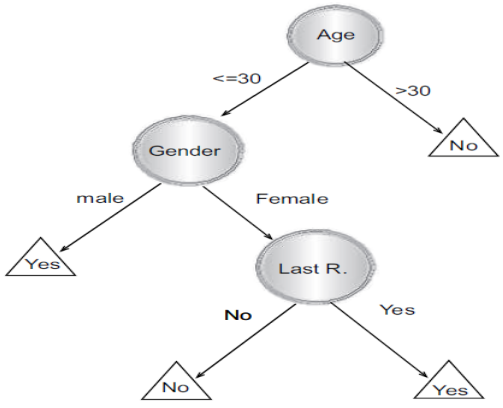


**Figure 2.2 Decision tree representing response to direct mailing**

Generally as stated by [17] decision trees can be considered as composed of three elements:

1. Future: The possible outputs of the decision tree model.

2. History: The information available to the model.

3. Questions: This is what a decision tree is all about. Finding the best sequence of questions to ask about the history to determine the future. In determining this sequence of questions the choice of the $m^{th}$ question to ask is determined by the answers to the previous m-1 questions.

[20] among the popular induction learning algorithms such as SVM, which we are going to see next ,the most attractive thing about decision tree is that its interpretability, which allows humans to validate the model and also generates of human readable explanations of results. The other important attractive side of decision trees is that when classification cost is important(because they ask only for the values of the features along a single path from the root to a leaf). In terms of accuracy, decision trees have been shown to be competitive with other classifiers for several learning tasks including NER[4].

The most widely used implementation of the decision tree algorithms are C4.5 and IDE3. [21] Those two algorithms employ a top-down, greedy search through the space of the decision tree.[22] In case of IDE3 in order to determine which attribute to select for classifying a given set , a statistical property called information gain is used to determine the worth of the attribute. Before using information gain to split attributes we need to determine the degree of impurity of a given set using entropy by using the following formula:

$$\text{Entropy} = \sum -P_j \log_2 P_j$$

Entropy is 0 if we have only a single class and entropy reaches maximum value when all classes in a given set have equal probabilities. So to determine information gain of a given attribute A relative to a given training set S, Gain(S,A), the attribute with the highest Gain(S,A) will be chosen using this formula:

$$- \sum_{V \in \text{Values}(A)} \frac{|S_V|}{|S|} \text{Entropy}(S_V)$$

But C4.5 applies a kind of normalization to information gain using a split information value and uses gain ratio for selecting attributes to split the data sets by using the formulas:

$$\text{Split Information}(S, A) = -\sum_{i=1}^{n} \frac{|S_t|}{|S|} \log_2 \frac{|S_t|}{|S|}$$

and

$$\text{Gain Ratio}(S, A) = \frac{\textbf{Gain Ratio(S,A)}}{\textbf{Split Information(S,A)}}$$

In conclusion[49] C4.5 is faster, can handle both categorical and continuous data and perform pre-pruning as compared to IDE3.

### 2.3.2 Support Vector Machines

The other machine algorithm that we have used in our experiment is the Support Vector Machine(SVM). SVM were first introduced by Vladimir Vapkin and his colleagues[23]. According to [23] SVM are well-known for their good generalization performance and have been applied to many recognition problems, including NER, chunking, parsing and text categorization to mention some. [23] Also stated that there are theoretical and empirical results that indicate the good performance of SVMs' ability to generalize in a high dimensional feature space without over-fitting the training data. Below is a brief description of SVM for a two class classification problem as described by [24][25].

A Support Vector Machine, or SVM for short, is a system which is trained to classify input data into one of two categories. The SVM is trained on a large training corpus containing marked examples of the two categories. Each training example is represented by a point plotted into a hyperspace. The SVM then attempts to draw a hyperplane between the two categories, splitting the points as evenly as possible. The simplest case to consider is a two dimensional SVM which is depicted in fig 3.

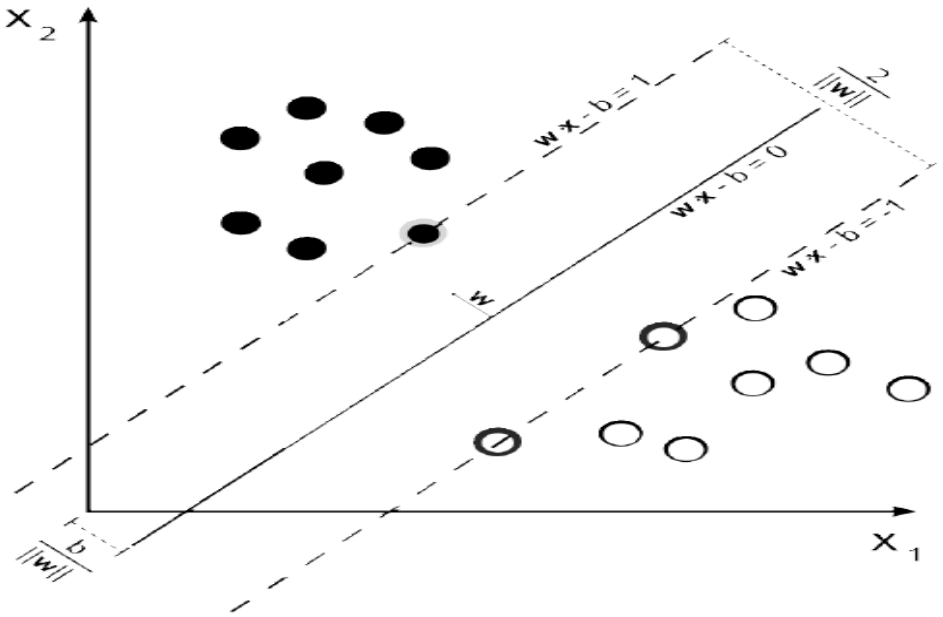


**Figure 2.3 two dimensional SVM**

Suppose that there is training data for the SVM in the form of n k-dimensional real vectors $x_i$ and integers $y_i$, where $y_i$ is either 1 or -1. Whether $y_i$ is positive or negative indicates the category for the vector i. The aim of the training phase of the SVM is to plot the vectors in a k-dimensional hyperspace and draw a hyperplane as evenly as possible to separate points from the two categories. Suppose that this hyperplane has normal vector w. Then the hyperplane can be written as the points x satisfying

$$\mathbf{w} \cdot \mathbf{x} - b = 0$$

where $b/||w||$is the offset of the hyperplane from the origin along w. This hyperplane is chosen so as to maximize the margin between the points representing the two categories. Imagine two hyperplanes lying at the 'border' of two regions in each of which there are only points of either category. These two hyperplanes are perpendicular to w and cut through the outermost training data points in their respective regions. Two such planes can be seen illustrated as dashed lines in fig 3. Wanting to maximize the margin between the points representing the two categories is the same thing as wanting to keep these two hyperplanes as far apart as possible The training data points which end up on the dashed lines in Fig. 1.2 are called support vectors, hence the name Support Vector Machine. The hyperplanes can be described by the equations.

$$\mathbf{w} \cdot \mathbf{x} - b = 1$$ and

$$\mathbf{w} \cdot \mathbf{x} - b = -1$$

The distance between the two is 2/||w||. Since the SVM wants to maximize the margin, we need to minimize ||w||. It also does not want to extend the margin indefinitely, since it does not want training data points within the margin. Thus, the following constraints are added to the problem:

$$\mathbf{w} \cdot \mathbf{x_i} - b \geq 1$$

for $x_i$ in the first category, and

$$\mathbf{w} \cdot \mathbf{x_i} - b \leq -1$$

for $x_i$ in the second category. This can be rewritten as the optimization problem of minimizing ||w|| subject to

$$(\mathbf{w} \cdot \mathbf{x_i} - b)y_i \geq 1, (1 \leq i \leq n)$$

If one replaces ||w|| with ||w||/2 , one can use Lagrange Multipliers to rewrite this optimization problem into the following quadratic optimization problem:

$$\min_{\mathbf{w},b} \max_{\alpha \geq 0} \left\{ \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^{n} \alpha_i((\mathbf{w} \cdot \mathbf{x_i} - b)y_i - 1) \right\}, \alpha_i \geq 0$$

where the $\alpha_i$ are Lagrange multipliers. Data sets which are possible to divide in two are called linearly separable. Depending on how the data is arranged, this may not be possible. It is, however, possible to use an alternative model involving a soft margin. The soft margin model allows for a minimal number of mislabeled examples. This is done by introducing slack variables i for each training data vector xi. The function to be minimized, $\|w\|^2/2$ is modified by adding a term representing the slack variables. This can be done in several ways, but a common way is to introduce a linear function, so that the problem is to minimize:

$$\frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \xi_i$$

for some constant C. To this minimization, the following modified constraint is added:

$$(\mathbf{w} \cdot \mathbf{x_i} - b)y_i \geq 1 - \xi_i, (1 \leq i \leq n)$$

By using Lagrange Multipliers as before, the problem can be rewritten as:

$$\min_{\mathbf{w},\xi,b} \max_{\alpha,\beta} \left\{ \frac{\|w\|^2}{2} + C \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} (\alpha_i (y_i (\mathbf{w} \cdot \mathbf{x_i} - b) - 1 + \xi_i) - \beta_i \xi_i) \right\}$$

for a, b >=0. To get rid of the slack variables, one can also rewrite this problem into its dual form:

$$\max_{\alpha} \left\{ \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \right\}$$

subject to constraint

$$0 \leq \alpha_i \leq C, (1 \leq i \leq n)$$

and

$$\sum_{i=1}^{n} \alpha_i y_i = 0, (1 \leq i \leq n)$$

The above formal description on SVM applies only to those problems which are linearly separable. For problems of non-linear separable In a non-linear classifier, the input vectors $x_i$ are transformed as to lie in an infinitely dimensional Hilbert Space where it is always possible to linearly separate the two data categories.

[23] noted the advantage of SVM over conventional statistical learning algorithms, such as Decision Tree, Hidden Markov Models, Maximum Entropy Models, from the following two aspects:

1) SVMs have high generalization performance independent of dimension of feature vectors. Conventional algorithms require careful feature selection, which is usually optimized heuristically to avoid over fitting which is suitable for morphologically complex languages.

2) SVMs can carry out their learning with all combinations of given features without increasing computational complexity by introducing the Kernel function, used for mapping non-linear problems into linear problems. Conventional algorithms cannot handle these combinations efficiently. Thus, we usually select important combinations heuristically while taking the trade-off between accuracy and computational complexity into consideration.

## *2.6 The Amharic Language*

In this section we are going to see some of the Amharic language grammars and structures pertaining to this thesis which is the identification and classification of NEs. First we will see some backgrounds of the Amharic language and following that we will discuss nouns and their categories. Finally we will explain some of the challenges that Amharic brings to the problem of NER.

According to[26] Amharic is a Semitic language which belongs to the family of languages like Hebrew, Syrians, and Arabic. Amharic is the second most spoken Semitic language after Arabic with an estimated 27 million speakers around the world and also the second most spoken language In Ethiopia and one of the five largest languages on the Africa continent.

Amharic is the official working language of the government of the federal republic of Ethiopia. After 1993 the year the Ethiopian constitution was drafted Ethiopia is divided into nine independent regions with their own languages. But Amharic is still used for country wide communication[27] and as a result most of the region's documents are produced in Amharic. In recent times there has been an increase in the amount of electronic Amharic documents available.

According to[27] Amharic writing uses a script which has originated from the Ge'ze language, which is now mostly used in the Ethiopian orthodox church. In the modern Amharic script each syllable patterns comes in seven different forms, reflecting the seven orders. There are 33 basic forms, giving 7*33 syllable patterns or most commonly called fidels. Two of the base forms represent vowels in isolation, but the rest are for consonants (or semi-vowels classed as consonants) and thus correspond to CV pairs, with the first order being the base symbol with no explicit vowel indicator.

The writing system also includes four (incomplete, five character) orders of labialized velars and 24 additional labialized consonants. In total, there are 275 fidels, but not all the letters of the Amharic script are strictly necessary for the pronunciation patterns of the spoken language; some were simply inherited from Ge’ez without having any semantic or phonetic distinction in modern Amharic. Most of the labialized consonants which are simply inherited from Ge'ze are redundant and according to the researches only 39 of the labialized constants are unique and that out 275 only 233

remains if those redundant ones removed. The language also has its own unique sets of punctuation marks and unlike other Semitic languages its written from left to right.

Words in Amharic are composed of one or more syllables and according to[28] words also can be classified into eight classes, also called part of speeches, which is similar to the one classified by [29]. The classes are nouns, pronouns, prepositions, adjectives, adverbs, conjunctions, interjections and exclamation. [29] gives criteria for classifying Amharic words in to these classes but stated that Most of the justifications given for classifying words into classes, which are meaning of the words and taking those part of speeches that classify English words, are invalid. But he produced a criteria in which why words belong in a given class. These criteria are shape of the word, position in which a particular word belongs to. Based on these criteria [30] produced five part of speeches namely nouns, verbs, adjectives, adverbs and interjections and give justification that the other three are left out because pronouns and interjections can be included in nouns and pronouns respectively whereas since exclamation shows emotion so that it should not be included in part of speech. Since our work involves in identification and classification of proper nouns we will discuss the noun word class.

### 2.6.1 Amharic Nouns

Nouns according to the Oxford dictionary are words used to identify any of a class of people, places or things or to name a particular one of these. For example ቤት፤ ላም፤ በግ etc are nouns. According to [18] in order for a given word or morpheme to be considered in the noun class first it is better to look those bound morphemes that exists within the words. Any word that includes the ኡ bound morpheme can be considered as nouns for example the words በግ፤ ቤት፤ ልጅ can be followed by the bound morpheme ኡ like በጉ፤ ቤቱ፤ ልጁ . Second a given word is considered as a noun if the word can be followed by the bound morphemes ኦች or ዎች. For example these words በጎች፤ ቤቶች፤ ዶሮዎች. The morpheme ኤ most of the time is added to a noun to indicate possessiveness like the words በጌ ፤ ቤቴ፤ ልጄ.

Amharic nouns can be classified into the following classes which are similar to that of the English language[28] namely:

- Tangible/Countable Nouns: Nouns which can be seen and touched by the naked eye and hand respectively. examples of those nouns are: ሰ ው፤ ካ ሜራ፤ መኪና ፤ ጠረ ፔዛ .
- Intangible/non-countable nouns: These are the nouns that cannot be seen by the naked eye and touched or felt by hand. examples are: ጨለ ማ፤ ሰ ውነ ት፤ ወፍረ ት፤ ልጅነ ት .
- Collective nouns: nouns which are used to denote a class of things or concepts instead of a particular things or concepts and these words most of the time are not followed by the አች bound morpheme to indicate plural nouns. examples are ወፍ፤ አ ሳ ፤ መን ጋ፤ አ ራዊ ት፤ ጫክ etc.
- Common nouns: are nouns which denotes a particular group of things or concepts in the real world and these nouns differ from that of the previous class of nouns in that these words can be followed by the አች bound morpheme. Examples are: መኪና ፤ አ ህ ያ ፤ በ ግ ፤ ፍ የ ል etc.
- Proper nouns: these types of nouns which are important to our works are nouns that specifically identify a specific person, organization, location names and others.

  examples of 1. person names: ሚኪ ያ ስ ፤ ሚሮ ን ፤ አ በ ራ፤ ከ በ ደ etc.

  2. Location names: አ ዲስ አ በ ባ ፤ ጎ ጃ ም፤ ድሬ ደ ዋ ፤ ጎ ን ደ ር etc

  3. Organization Names: ሚድሮ ክ ኮ ር ፕ ሬ ሽ ን ፤ ኢ ን ፎ ር ሜ ሽ ን መ ረ ብ ደ ን ነ ት ኤ ጀ ን ሲ etc. One distinguishing characteristic of the proper nouns is that they are not followed by the bound morpheme አች. In the implementation section of our thesis we are going see some of the other clues that helps identify these types of nouns from an open text.

## *2.7 Challenges of Amharic NER*

One obvious challenge encountered in trying to solve the problem of NER is that proper nouns specially person and organization names belong to the open class category, which means new names of these kind are added every now and then to those list. The other one is ambiguity and according to[2] there are two types of ambiguity. The first ambiguity occurs when the same name refers to different entities of the same type as in the name JFK can refer to the president or his son. This type of ambiguity occurs in those countries where their names are called using their last name. The other ambiguity occurs when the same entity mentions refer to different entities as is the name JFK can refer to the airport or a person's name. These challenges are challenges encountered by both resourcefully poor and rich languages. Below are some of the challenges encountered by Amharic and also by other morphologically rich but resourcefully poor languages.

**1. Lack of Capitalization:** In Amharic there is no capitalization when writing proper nouns. In most European and English languages proper nouns are written by making the initial letter of the word capital which makes it easy for the identification and classification of NEs.

**2. The Agglutinative Nature:** Amharic language like Arabic as put by [4] has a high agglutinative nature in which a word may consist of prefixes, lemma and suffixes in different combination which results in a very complicated morphology. For instance a given person name like መሉ አ ለ ም can be takes as a location name when the morpheme ከ put in front of it.

**3. Spelling Variation**: As put by [4] Amharic words may be spelled differently but the words still refer to the same name with similar meanings. For example አ መት and ዓ መት and also መስ ቀ ል and መሥ ቀ ል have the same meaning but spelled differently.

## *2.8 Related Works*

In this chapter we will cover selected research works in the field of NER and the approach followed is categorizing the researches done according to the NER approach discussed in section 2.2.2. The reason for the approach followed is to show what makes the hybrid approach different from the Rule-based and Machine Learning (ML)

approaches. We start the chapter by reviewing researches conducted using the rule-based approach followed by the machine learning and hybrid approach. We conclude our review by summarizing and comparing some of the results of those research works.

### 2.8.1 Rule-based Approach

The rule-based approach to NER [4] depends on handcrafted linguistic rules to recognize and classify NEs within an open text using linguistic, contextual clues and indicators. According to[4] the system exploits gazetteers or dictionaries that aid the identification task. The rules most of the time are implemented in the form of regular expressions or finite state transducers. As we have already stated in sections 2.2.1 maintenance of rule-based systems are tiresome and time consuming as it involves a linguist to formulate new rules and modify existing rules. In this section we are going to review three articles two of which are works based on Arabic language and the third one is on Urdu language. The reason we have chosen these works is that they share some linguistic characteristics with Amharic.

The first work that we are going to review is the work of[ 31] which uses the rule-based approach to solve the NER problem. The resources that they have created for their work includes; a White list representing a dictionary of names, and a grammar, in the form of regular expressions, which are responsible for recognizing the named entities. A filtration mechanism is used that serves two different purposes: (a) revision of the results from a named entity extractor by using metadata, in terms of a Blacklist or rejecter, about ill-formed named entities and (b) disambiguation of identical or overlapping textual matches returned by different named entity extractors to get the correct choice. They have chosen to identify 10 different NE types namely; Person names, locations, companies, dates, time, prices, measurements, phone numbers, ISBNs, and file names. In order to build their system they have used Fast ESP which is an integrated software environment for development and deployment of searching and filtering services. They have prepared their own corpora for the different categories of NEs. They have used the precision, recall, and f-measure to measure the performance of their system by individually testing the accuracy of each of the Named entity(NE) types. Based on that on average for the 10 NE types the system scores an f-measure of 92.21%

which the papers concludes it was a satisfactory result considering the size of the corpus in which they have conducted their work.

The other work which is recent compared to the previous work is the work of [32] which concentrated on only recognizing one NE type namely person names. Unlike some of the previous work this work tries to include as claimed by the researchers corpus from other domains. According to them most previous works use corpus which concentrate on political domains but this works concentrates on other domains such as sport and economics domains. As they have stated some of the triggers words used for those domains is different from that of those found in the political domain. They have identified four rules for identifying person names depending on their location relative to the trigger words. The first rule identifies the person names that occur after the Introductory Words Person List(IWPL), which is defined as the words that come before the names as definite word and they use POS tagger for identifying words that come after IWPL. The second rule identifies person names that occur before IWPL. The fourth rule identifies person names that occur before Introductory Verbs Person List (IVPL),which consisted of the verbs which can be positioned before or after the person names. And finally the forth rule is used to recognize person names that occur before the introductory person verb lists, which includes verb particle constructions. After training and testing their system with 256KB size corpus and evaluating it from the perspective of different domains they have found an f-measure of 91.71% for the overall corpus.

Finally the rule-based work that we are going to discuss is the work of[42]. This work is on the Urdu language, which is the official language of Pakistan and most spoken language in India. One unique characteristic of the Urdu language is that it uses an Arabic script unlike other south east Asian languages. Like Amharic and Arabic the Urdu language faces the same challenges faced by the other two languages. The paper also tries to justify that Hindi and Urdu are two different languages especially when it comes to recognizing names. They disproved the assertion that any computational model or algorithm that works for Hindi should work for Urdu also. The paper proposes a rule-based approach to NER as compared to the ML approach. They have enlisted the following reasons why they used the rule-based approach. To name a few: lack of large corpus, lack of gazetteers, and preparing annotated large corpus is expensive for them.

They have used two untagged corpuses prepared for NER in Urdu. They have constructed rules to identify these entities: person name, location, organization, date, and number. Their rules forms a finite state automata(FSA) based on lexical cues. These rules are corpus-based, heuristic-based, and grammar-based. The rules are implicitly weighted in the order they are applied. The rule sets are created from 200 documents corpus and the experiment run on another set of 2,262 documents corpus they have used. The experiments showed an F-measurescore of 91.1% a score higher than any of the other scores obtained so far using ML approach.

### 2.8.2 Machine Learning Approach

In this category of researches as we have done previously in the rule-based approach we will try to cover NER researches for those languages which are considered to be complex as compared to those languages which have many clues for easy recognition of Named entities. Those languages have scares resources for the applications of NLP. The reason we have done so is because Amharic belongs to these categories of languages as we have seen on previous chapter the challenges related to Amharic languages.

The first research that we are going to see is the work of [34] for two Indian languages Bengali and Hindi. Bengali is the seven most spoken language in the world, second in India and it is the national language of Bangladesh. Hindi is the third most spoken language in the world and national language of India. The paper start by pointing out the challenges faced when trying to build a NER for these two languages, which show similar characteristics to the Amharic language in terms of the challenges faced. And also claims that SVM is the state of the art in NER tasks and they have chosen this particular algorithm because it is more efficient than HMM and other statistical approach for dealing with the non-independent, diverse and overlapping features of the highly inflected Indian languages. They have used the 12 NE tag sets defined by the IJCNLP-08 NER Shared Task for SSEAL which are: person names, location names, organization names , and miscellaneous, which includes time, date, percentages, numbers, monetary expressions and measurement expressions. A total of 122,467 tokens for Bengali and 502,974 tokens for Hindi are annotated using the tag sets defined by IJCNLP-08 NER Shared Task for SSEAL. The features they have used only consist of those which are considered to be language independent among them are context word features, word

suffix, word prefix, previous NE, POS information and other. They conducted their experiment on the test data of 35k and 60k for Bengali and Hindi respectively using a 10-fold cross validation testing and found that precision, recall and f-score values of 88.61%, 80.12%, and 84.15% for Bengali and80.23%, 74.34%, and 77.17% for Hindi. Results show the improvement in the f-score by 5.13% with the use of context patterns which they have used as a feature in the SVM to post-process the system.

The other work previously done by the same authors[35] is for Bengali language . This was the first research for Bengali language using the SVM. In their work they have used both language dependent and independent features for learning. A total of 150k tokens which are manually annotated have been used as a training set.10-fold cross validation testing is used to obtain with an overall average Recall, Precision and F-Score of 94.3%, 89.4% and 91.8%, respectively. The authors claim that Their work outperforms other NER systems for Bengali language.

Another work that is based on SVM is the work of [35] for Arabic. The work investigate the impact of language dependent and independent features for NE classifications, which includes lexical, contextual, morphological , and shallow syntactic features. They have experimented with two sets of data, the standard Automatic Content Extraction(ACE) data called ANERcorp and a manually prepared data sets called UPV-corpus. They have experimented each feature in isolation and in combination and found that their system performs better when using the all features in combination and scored an F1 measure of 82.71% .

There are also a total of four researches conducted in the area of NER, two of which are for Afan Oromo and the other two are for Amharic.

NER for Afan Oromo[8] is the first work made for Afan Oromo language which is according to [5] is a mother tongue for Oromo people who are the largest ethnic group in Ethiopia compromising 33.5% (22.5 million) people according to 2008 census. Afan Oromo also faces some of the challenges like other languages which are resource poor languages, which[8] mentioned thoroughly throughout his thesis. [8] Proposed a hybrid approach for his work but limited his research to machine learning approach mainly due to lack of available time but nevertheless their architecture vividly shows a NER system which incorporates both machine learning and rules which assists the machine learning

component in recognition phase. They have used beginning, outside, and inside (BOI) scheme for tagging their data and followed CoNLL 2002 for tokenizing and tagging their data. They have used CRF learning algorithm for building their model among the features they have used are language independent features which includes word-shape features, position features, part-of-speech tags, normalized tokens, and prefixes and suffixes of a token each of these features are represented by a binary value of 1 or 0. They have prepared manually Afan Oromo corpus which consists of 23,000 words out of 3600 are NEs they have divided their data into 8000 tokens for training and 4000 for testing and obtained recall, precision, and $F_1$ measure of 77.41%, 75.80% ,76.60% respectively. The conclusions reached that features play a great part in boosting the performance of the NER and POS feature has least contribution among the features chosen.

Another work which is the first work for Amharic language is the work of[6],which have used a pure machine learning approach to the problem of NER. [6] used the CRF algorithm to train and build NER model . They have stated that feature selection plays a key role when using a CRF algorithm as it was done by[8].[6] have experimented the performance of their model in isolation and in combination of features to determine the best features for NE tagging. The corpus they have used is the one from WIC which are prepared from 1,065 news articles which constitutes 210,000 words out of which they have cleaned and tagged only 10405 tokens and they have only used 8005 for training and 961 for testing. They have included both language and language independent features and when combining all the features they have obtained a better performance as compared to other experiments with precision, recall, and F1 measure of 72%, 75% and 73.47% respectively. They concluded that word context features, POS tags of tokens and suffix are important features in NE recognition and classification for Amharic text.

Recently the work done by [9] is another work for Amharic language. They have used the same corpus and machine learning algorithm used by [6] but includes an extensive experiment on various features to determine the best feature set for Amharic NER. But due to using different tool from that of [6] they are forced to set up a base line for themselves and also exclude some features from theirs, like End of sentence( EOS) and Begging of sentence(BOS) features and POS features. The highest performance they have achieved is the one with a window size of two on both sides, previous and next tag

of a current word and prefix and suffix with length four of F1 measure of 80.66% which is a significant improvement as compared with that of [6].

### 2.8.3 Hybrid Approach

By following the same trend as we have done for the other approaches we will see some of the research done using the hybrid approach which tries to combine the rule-based approach with the machine learning approach by taking strong side from each of the two approaches.

As we have seen previously in the machine learning approach the work of[8] is the only work, which is done locally for Afan Oromo language, using the hybrid approach even though due to time constraint they have decided to experiment their work using the ML component of their system. According to their system architecture they have incorporated the rule-based approach during the recognition process of NE. In the recognition phase there exists two main components the model built using CRF and stored rules which are supposed to be build using pattern matching techniques. The rules focus on the general structure and organization of Afan Oromo NEs which tried to cover some of the knowledge not known by the Model. Generally most of the rules identified by[8] are list of clues used in identifying the NE types.

The work of [37] tried to build NER for five Indian languages namely Hindu, Bengali, Urdu, Oriya, and Telugu. Among these five languages only two of them are known by the researchers for that reason they are able to build rules and extract context patterns for Bengali and Hindu. The rest of the language are experimented using the Maximum Entropy machine learning algorithm which acted as the base line for the five languages. They have build their model by using various features which are both language dependent and independent but POS and morphological features are only included for Bengali and Hindu. They have tried to tag twelve NE types defined in the IJCNLP-08 NER Shared Task for SSEAL among which only three classes are chosen to build a rule as they have claimed that those classes are better identified by rules rather than by MaxEnt method. The classes are number, measure and time. And also they have build a semi-automatically extraction of context patterns those patterns, which tries to identify a pattern for a specific NE class, which are presumed to aid the identification process. Finally for Bengali and Hindu they have achieved an f-value of 65.96% and

65.13% respectively and concluded that their work has outperformed some of the other works in the IJCNLP-08 NER Shared Task for SSEAL and the rules and gazetteer list would be more effective if there exist a large amount of training data. Most of the works presented in Proceedings of the IJCNLP-08 Workshop on NER for South and South East Asian Languages have used a hybrid approach to tackle the problem of NER. Even though the languages covered are resource poor languages most of the researches agree that using language specific rules to augment the models build using the ML approach can improve the overall performance of the NER system.

Most of the hybrid work that are covered in IJCNLP-08 Workshop on NER for South and South East uses the rule-based component as post-processing the machine learning model so that they can recognize those entities that are missed by the model or resolve some ambiguities that are not resolved. But the next work[4] that we are going to review is different because it uses the NEs recognized by the rule-based component as a feature for the machine learning component, meaning that they have included the rule-based component as part of the feature extraction component of the system.

[4] Have developed a NER system for Arabic language which is a Semitic language in which Amharic is also part of. They stated that there is only one research which have applied the hybrid approach in building a NER system. This system have identified only three named entity types namely Person , Organization and location. The rule-based part of their component is simply a reimplementation of the work we have reviewed in the rule-based approach section using GATE. The machine learning algorithm they have used is decision tree and they have incorporated the rule-based decisions as a feature with other future of vectors. The other features considered are word's length, POS tag, Noun flag (i.e. a binary feature to indicate whether POS tag is Noun or not), gazetteers, statement-end flag, prefix and suffix features according to [10] the previous has outperformed some of the researches using the same data set.

But the work of [10] differ from that of the previous work first this work recognizes 11 deferent entities namely Person, Location, Organization, Date, Time, Price, Percent, Phone Number, Measurement, ISBN and File Name. Unlike the previous work which have used only the ANERcorp which a freely available data set for research purposes but this work in addition to ANERcorp they have used the Automatic Content

Extraction (ACE) corpora and Arabic Treebank (ATB) Part1 v 2.0 dataset, and also prepared some data sets specially for file names, phone numbers and ISBN numbers as they have thought that it is not sufficiently covered in the other data sets. They have tagged their 11 types NEs using an XML tag.

They have taken from the previous research gazetteers for the person, organization, and location names and they have prepared by themselves gazetteers for the rest of the other NE types all in all a total of 19,328 keywords in the gazetteer list. Like the previous work the rule-based component is a reimplementation of the[34] work using the GATE tool but the rule-based part in this research has been improved to accommodate the additional NE types. For experimenting purposes they have chosen three widely used ML algorithms namely; SVM, decision tree, and logistic regression. According to [4] they have chosen SVM and decision tree algorithms for their high performance in NER in general and Arabic NER in particular; whereas, the third technique is a new investigation that has never been tested before. They have used WEKA and for the decision tree algorithm is applied using the J48 classifier, SVM with the LibSVM classifier, and Logistic Regression with the Logistic classifier.

Unlike the previous research they have used a comprehensive list of features which are both language dependent and independent. They have setup their experiment by dividing their test into three; in the first one is all features are considered and in the second one without the rule based component and the last one without the morphological features and they have summarized their work using a table depicted below. Concluded that the hybrid approach outperforms the pure Rule-based approach and the pure ML-based approach and when applied to ANERcorp it outperforms the state of the art ANER systems with an f-measure of 94.4% for Person named entities, f-measure of 90.1% for Location named entities, and f-measure of 88.2% for Organization named entities. The above work is the main motivation for doing our thesis ,NER for Amharic language using the hybrid approach.

# Chapter Three
# 3. Methodology

This chapter describes the methods and the techniques used to collect and prepare the data which are used to conduct the experiment. It also describes the tools used to prepare the data and to run those data on the machine learning algorithms. Finally, it enlists the features set used as an input to the ML (Machine Learning) algorithms and the performance analysis metric used to evaluate the NER system.

## *3.1 Research Method*

Generally our research method can be considered as being part of a quantitative method, in which the answer to the research question can be quantified numerically. But specifically our research belongs to the sub category of quantitative research method, i.e. experimental quantitative research method. In experimental research method the answer to research questions is obtained by conducting various experiments. We have conducted different experiments to explore the performances of NER models built using decision tree and SVM algorithms by incorporating simple rules used to identify named entities.

## *3.2 Data Collection*

Any kind of researche conducted in NLP requires resources and among those resources the most important and vital one is data. Since most of the state of the art researches are conducted using the corpus-based approach using various ML algorithms it is obvious why data is vital. But in Amharic, which is a resource poor language, shortage of standard corpus is a major problem encountered when conducting NLP research.

The corpus that we have used for our research is the one used by Besufkad [9], which he has acquired it from the Ethiopian language Research Center of Addis Ababa University. The project called "the manual annotation of Amharic news documents" was engaged in manually annotating each Amharic word in its context with appropriate POS (part of speech) tag by using WIC's (Walta Information Center) 1065 news articles which contains approximately 210,000 words[53].

Since the algorithms that we have used in our rule-based component heavily depends on POS information of a given word we have used the original WIC corpus in

transliterated form to train and build Amharic POS model. Furthermore, we have used the POS information predicted from the model as part of the feature sets we use for our ML algorithms.

We have used a total of 12196 words obtained from [9] to train and test The NER model. But for training the POS tagging model we have used all WIC corpus.

## *3.3 Data Preparation*

Data preparation is the most difficult and tiresome task in NLP research since the performance of the system is highly dependent on the quality of data. The data prepared by [9] is important to our research but the format required by the tool we have used is different. So we are forced to modify the format of data that we have obtained from [9].

More specifically for building the NER model we have used the WEKA tool (described in section 4.4). WEKA requires that data should be prepared in a specific format, namely ARFF (Attribute-Relation File format). The arff file format has two distinct sections namely the header information section and the data information section. The header section of the ARFF file contains the name of the relation, a list of the attributes along with their data types. The data information section is simply a comma separated list of values that corresponds to the attributes declared with respect to their sequence. An excerpt taken from our training data in ARFF format can be found in Appendix A

But for the POS tagging we have used a different tool than WEKA and it is called openNLP (described in section 4.4). openNLP requires that the data used for POS tagging should be prepared in a format which involves the word that is to be tagged separated by a specific separator character specified in the property file followed by the POS tag. Each line of the file contains one sentence. The property file that we have used for our experiment is the default one which includes those features which are language dependent and that is generated by the application but we have modified the property file so that our separating character should be "_". An excerpt taken from our training data using the openNLP format can be found at Appendix B.

Since the approach that we are going to follow for building the NER learning model involves the use of rule-based component as a feature, the POS tagger will be one of its components. We have also added a new POS class that did not exist in the original

WIC corpus, the proper noun tag (NNP). The reason we have included the NNP tag is that in the original WIC corpus proper nouns are tagged as nouns (N), noun with preposition (NP) , and noun with conjunction (NC) which can be used to tagging other type of nouns which are not person, organization , and location names.

We have found it to be tiresome to manually tag proper nouns in WIC. So we have written a program that maps those words tagged as NEs in the data we have used for building the NER model with that of the training data we have used for POS tagging. After mapping those words we have changed their POS tag from N, NP, and NC to NNP.

The data we have used for our rule-based component consists of wordsthat represent trigger words and these words are prepared as a list that will be used in the algorithms that we have used to detect and classify NEs. Some of the trigger words list prepared can be found at Appendix C.

## *3.4 Tools*

In this study we have used WEKA for building our NER model and Apache openNLP is used for building our POS tagger model.

**WEKA**:-stands for Waikato Environment for Knowledge Analysis. As is defined by [41] WEKA is a collection of state-of-the-art machine learning algorithms and data processing tools. It has all the features that are needed for generic machine learning tasks including: data preparation, building ML models, evaluating the models built using different evaluation metrics and visualizing the input data and the final result using different graphical tools.

In WEKA the user is not expected to write any program as long as the data is prepared in a format that is required by the tool. Once there are data it can be preprocessed using the different features available in preprocessing stage of the tool as depicted below in fig 3.1

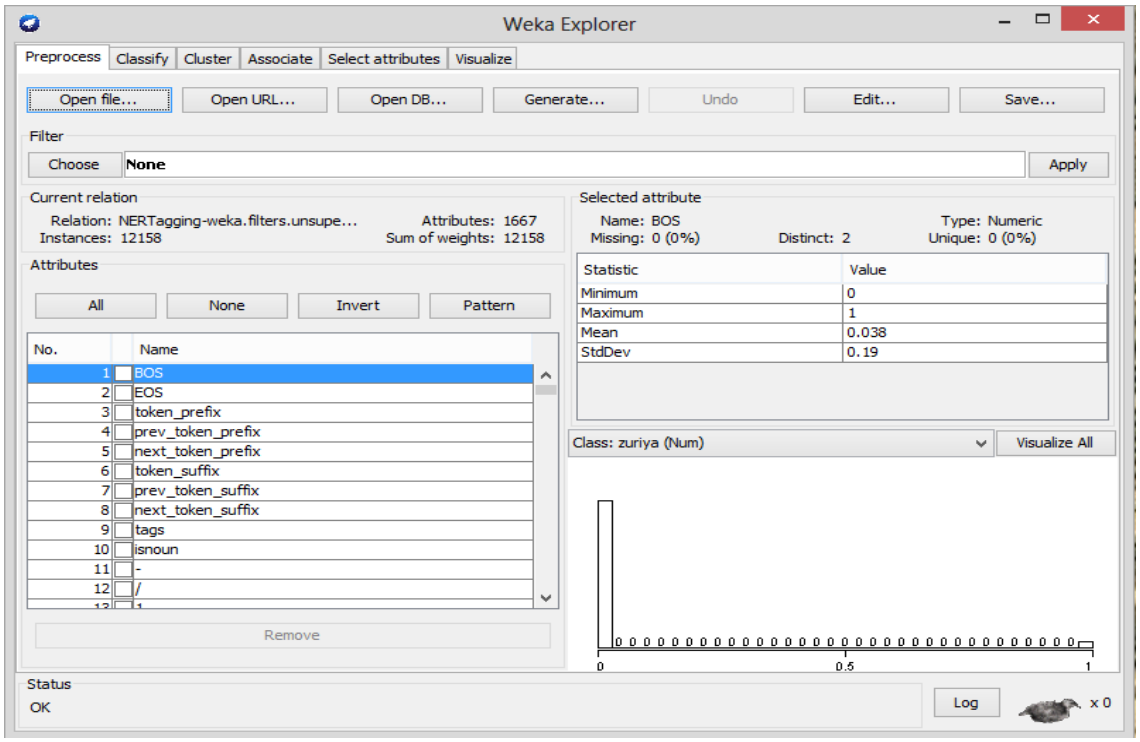


**Figure 3.1 Data preprocessing in WEKA**

After preprocessing we can use the different types of ML algorithms to build a model for different applications. Since the problem of a NER is a classification problem we have used two algorithms that are available in WEKA. J48 which is a decision tree algorithm already preinstalled with WEKA.WEKA allows third-party developers to develop an implementation of ML algorithms and include it with its set of ML algorithms using the package manager. Using the package manager we have installed the Lbsvm library that has an implementation of SVM, which is one of the algorithms that we have conducted our research.

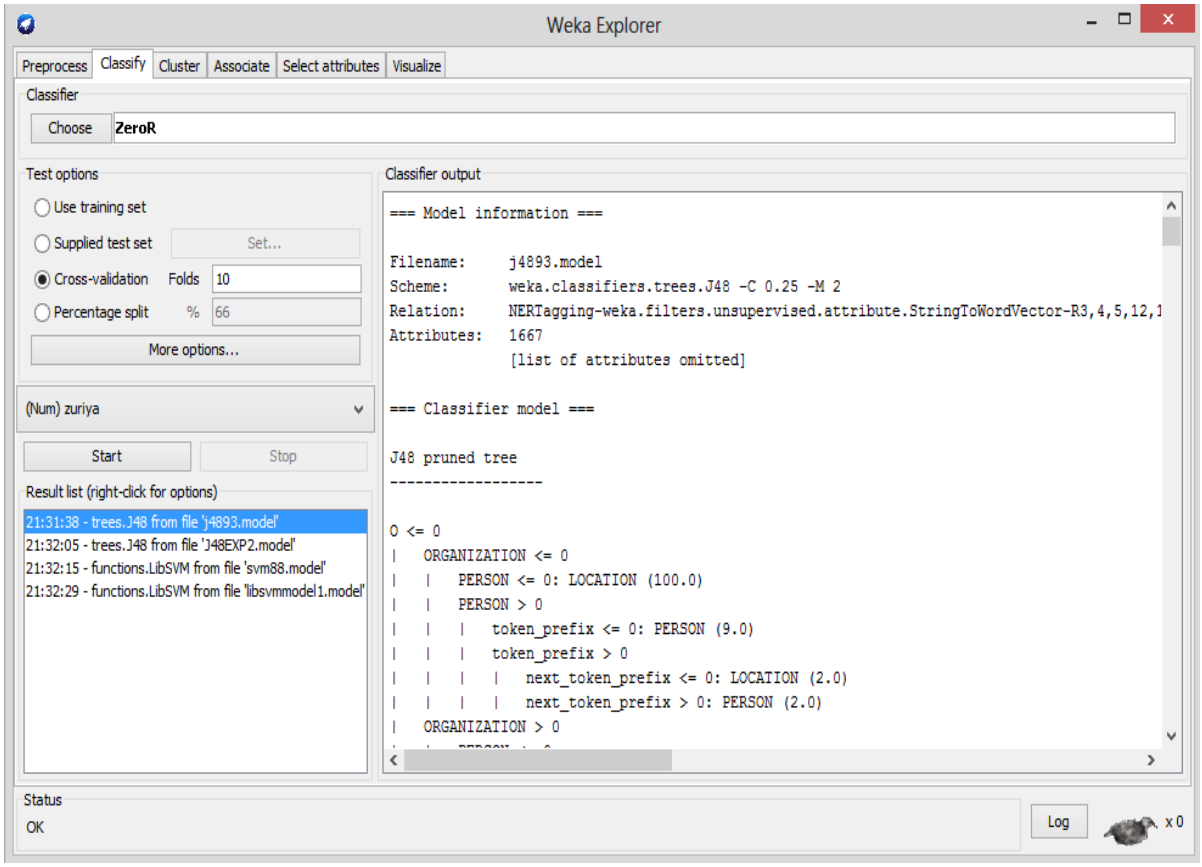


**Figure 3.2 model building in WEKA**

When it comes to evaluating the performance of the model built, WEKA provides options to evaluate the model by using three different kinds of mechanisms to supply test data. The first one is to simply supply the test data which is prepared by the user. The second one is to apply cross-validation, which is appropriate when having a small amount of data to split it into training and testing data. The third mechanism is to split the data in percentage into training and testing. The version that we have used for our experiment is WEKA 3.7.10 [54].

**Apache openNLP**:-The documentation [55] describes Apache openNLP as machine learning based toolkit for the processing of natural language text. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and co-reference resolution. These tasks are usually required to build more advanced text processing services. But unlike WEKA, openNLP supports a small set of machine learning algorithms and also do not have pre-processing and visualization components.

We have used Apache openNLP to build a POS tagger model that we have used as part of our features sets. Apache openNLP has a tool called POS tagger that builds POS tagging model given the data is prepared in a format required by the tool. The ML algorithm used is maximum entropy Markov model (MEMM).

Finally, we have written a Java program to build a feature extractor, the rule-based component, and to prepare the data in ARFF format that can be used in WEKA. The feature extractor and rule based component part is described in detail in the next chapter. We have also written a java program to prepare data for Apache openNLP.

## *3.5 Feature set*

From the literature review, we have learned the impact that features have during the learning phase of NER. Since we have chosen to conduct our experiment on algorithms and on tool that have never been tried previously by researchers for Amharic NER so choosing feature sets has been a problem. But since [9] have obtained a better performance than [6] we have chosen to use [9]'s optimal feature sets as a base line, which are words and NE tags of words of window size two and prefix and suffix of words. POS tag features are not used by [9] but we have decided to include it since state of the art NER systems have used them extensively. Moreover, a binary feature called nominal flag feature which indicates whether a given word is a noun or not have been used as a feature.

## *3.6 Performance Measures*

The WEKA tool has a built-in performance analysis component that allowed us to evaluate our model in an easy and efficient manner. As we have discussed in section 2.2.3 the evaluation metric that we have used in order to analyze the performance of our systems are recall, precision, and F1-measure, which are typically used metrics in information retrieval. WEKA calculates these performance metrics automatically after the models have been built.

# Chapter Four
# 4. Design and Implementation

In this chapter, we discuss the design and implementation of our Amharic Named Entity Recognition (ANER). We start with the overall design of our system, which describes the approach chosen for building the ANER. The architecture that vividly depicts the process involved along with the main responsibility of each process is presented. Then we proceed to the implementation section which explains the algorithm used to implement the rule-based component and the feature extractor.

## *4.1 Design of ANER*

### 4.1.1 The approach used

The approach that we have used to experiment NER for Amharic is the hybrid approach. The hybrid approach consists of two major components in which by themselves can be complete NER systems. The first component is the rule-based component which we have built using two algorithms that predicts the NE tag of a given word based on the presence of trigger words such as ኦቶ,ወይዘሮ etc. The second component is the ML component which uses different features that we have extracted from the corpus obtained from [9] including using as a feature the NE prediction from the rule based component. This component applied two ML algorithms separately to experiment their performance for the hybrid ANER.

The integration of the rule-based component to that of the ML component can be done in two ways as discussed in section 2.2.2.3. We have chosen to use the predictions of the rule-based component as part of the feature set we have used for building our NER model.

Our ANER system also employs another major application of NLP, that is Part of Speech (POS) Tagging. According to [9] the use of POS information as feature has very limited significance, especially when using Stanford NLP named entity recognition tool as most of the features used for POS has already been used for NER. But [6] uses POS information as a feature and in his experiment the lowest performance recorded for his ANER is when the experiment conducted without POS information. So we have used POS information as feature in our experiment.

The approach we have used for building the POS tagger model is purely a ML approach using the Maximum Entropy Markov Model (MEMM) ML algorithm.

### 4.1.2 ANER Architecture

In chapter two Section 2.2.1, we have seen a generic architecture of NER. It shows the major tasks involved in NER and how data flows from one component of the system to the other. But in this section we are going to present a more specific architecture that we have used to build our ANER system.

It is known that the major tasks involved in NER are the detection and classification of words into their perspective named entity classes. Any architecture should be based on these two major tasks of NER. Furthermore, before detecting and classifying a given data, the data must be pre-processed that is it should be cleaned and prepared in a format that is appropriate to the task at hand. Additionally, after the prediction algorithm classifies the given data it must be used to detect and classify NEs.

It is clear from the above paragraph that there are at least five major components for our ANER. They are: the pre-processing component, the learning component (since detection of our ANER is done using ML algorithms), the rule-based component, the classification or recognition component, and finally the component that uses the model built.

We have adopted the architecture of [4] which has used a hybrid approach for building NER for Arabic language. The reason we have adopted their architecture is that it uses a similar approach of integrating the rule-based component as ours. But we have elaborated more on ours by adopting the architecture of [6] and [9] which is depicted below in Fig4.1

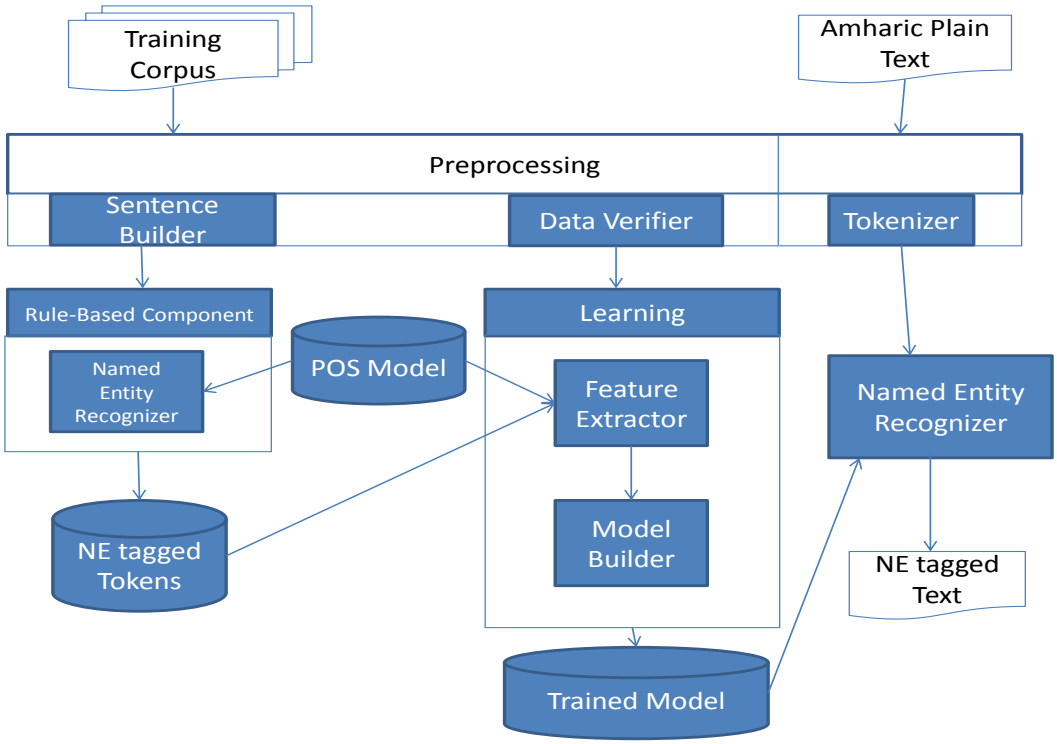


**Figure 4.1 A hybrid approach NER architecture for Amharic.**

The diagram in Fig 4.1depicts how the ANER is structured. The diagram shows that each of the components of the ANER can function on its own as long as input data is provided in a format suitable to those components. One advantage of this layering approach is that the system can be reconstructed without necessarily building the whole system from scratch. Only those components that require improvement are given attention. Below is the description each of the components of the architecture:

- **Pre-Processing:-** The pre-processing step is the most crucial and important step in all Natural Language Processing (NLP) applications, specifically those that apply ML algorithms. Since our ANER bases its prediction and classification on the data provided, the quality of its prediction and classification depends on the quality of the data. The WEKA tool that we have used for our experiment has a built in pre-processing functionality that simply transform our data into the type of data required by the ML algorithm. In addition to WEKA's pre-processing feature, we have included the following three components into our ANER system.

- ***Tokenizer:-*** The tokenizer component reads a plain Amharic text sentence by sentence and split it into words. For our purpose since Amharic sentences constitute words separated by a space we have used a space as a separating character to split words into tokens. But a more comprehensive tokenizer for Amharic should also consider splitting compound words which our tokenizer was unable to do.

- ***Data Verifier:-*** This component reads the already tokenized training data and checks to see that each line consists of a word and a tag separated by a space. It also checks to see the data are tagged as PERSON, LOCATION, ORGANIZATION, and O. Those lines which are tagged wrongly are rejected from the list.

- ***Sentence Builder:-***This component reads the training data and separates the words from their tags. It uses the words to construct Amharic sentences using the four dots (::), question mark(?), and exclamation(!) as sentence ending punctuation marks for each sentences. Since those punctuation marks are included in the original corpus along with the words.

- **Rule-based Component:-**This is the component that given a sentence classifies words in the sentence into their named entity types based on list of trigger words and POS information of words. The output of this component is a list of words along with their named entity types, which can be used later in the feature extraction step to determine whether a given word is a member of a particular named entity type or not.

- **Learning:-** The learning component is the component that builds a ML model that detects and classifies words into their named entity categories. The learning method that we have used for our ANER task is supervised ML. Supervised ML requires data already labeled with appropriate classes so that the algorithm learns using different mathematical formulas or rules from those annotated data and

classify or predict a given data into appropriate classes. The learning component consists of two subcomponents.

- ***Feature Extraction:-***The feature extraction component given a word extracts those features that are found important to classify the word into its named entity type. This component uses POS information predicted by our POS model and also the prediction from our rule-based component. The feature extraction component finally produces data in a form appropriate for the ML algorithm.

- ***Model Builder:-*** By taking the data from the above component the ML algorithms builds model for the recognition and classification of words into their named entity types. Our experiment uses two widely used ML algorithms in order to build a learning model. They are WEKA's J48 implementation of decision tree and libsvm's SVM in WEKA.

- **Named Entity Recognizer:-** This component accepts tokenized words from the tokenizer and uses the model built from the learning component to detect and classify words into their respective NE types.

## *4.2 Implementation of ANER*

### 4.2.1 Rule construction

Before the popular usage of a corpus-based statistical approach to NLP the most popular and widely used way of solving different NLP problems is through the use of Rule-Based approach. The rule-based approach consists of a handcrafted linguistic rules used to solve a given NLP application problems. For example, when considering the problem of NER which is a classification problem those handcrafted rules are used to detect and classify words into named entity classes. One of the biggest disadvantages of the rule-based approach is that it requires an expert in linguistic to maintain those systems. Nevertheless, they are accurate when it comes to accomplishing their tasks.

Most of the NLP researches done in Amharic uses a corpus-based approach since it doesn't require a linguist to manually prepare rules. And also by the time research on NLP become popular in Ethiopia which is some 20 years ago the popular approach was

the corpus-based statistical approach. But the problem with the Corpus-based approach is that this approach requires a huge corpus size. To compensate the shortage of large corpus size, most researches focus on optimizing the learning algorithms using only the available corpus.

The ANER system we have built also tries to optimize the learning algorithms by using the prediction of the rule-based component as a feature which gives more information during the learning process. Due to time constraint we were unable to consult a linguist and develop an extensive list of rules. In spite of this we have managed to use two rules which classify words into NE types based on the occurrence of trigger words in a given sentence. The algorithms that we have used for implementing our NEs using trigger words are adopted from the works of [32].

The work of [32] recognizes person names using a list of trigger words that appear before and after NEs. But we have adopted their algorithms to identify location and organization names too.The source code for the Rule-based component can be found at appendix E.

| Algorithm 4.1 | Rule to identify NEs that occur after trigger words |
|---|---|

```
Rule1(S:Senetence)
  P=IsIPWL(S) // get position of IPWL in S
  If P=0 Then
          Exit // no IPWL keywords in S
  Else
          For each  W in S From(P+1)
                  If Wϵ IPWL Then
                          S=Copy String(S,P+1)
                          NameRule1(S)
                  Else
                          If POS(W)=NNP then//NNP is used to identify proper nouns
                              Label(W)=Named Entity type// which can be PERSON,LOCATION,
                         ORGANIZATION.
                          Else
                              EXIT.
           P=P+1
End Rule1
```

| Algorithm 4.2 | Rule to identify NEs that occur before trigger words |
|---|---|

```
Rule2(S:Senetence)
  P=IsIPWL(S) // get position of IPWL in S
   If P=0 Then
          Exit // no IPWL keywords in S
   Else
         For each  W in S From(P-1 to 1)
                 If Wϵ IPWL Then
                          S=Copy String(S,P+1)
                          NameRule1(S)
                 Else
                          If POS(W)=NNP then
                               Label(W)=Named Entity type// which can be PERSON,LOCATION,
                         ORGANIZATION.
                          Else
                               EXIT.
                P=P-1
End Rule
```

The first rule, for example, can recognize the person names ደሳለኝ ሀይለ ማሪያም , አስማን አልበሽር , and ቴድሮዎስ አድኣኖም in the following two sentences as PERSON named entities given the words ክቡር , ጠቅላይ ,ሚንስተር , ፕሬዝዳንት, የውጭ ጉዳይ, ዶክተር , are included in the lists.

ክቡር ጠቅላይ ሚንስተር ደሳለኝ ሀይለ ማሪያምለ ሱዳኑ ፕሬዝዳንት አስማን አልበሽር አቀባበል አደረጉ።።

የውጭጉዳይ ሚኒስቴር ክቡር ዶክተር ቴድሮዎስ አድኣኖምለጉብኝት ወደ አሜሪካ ጉዞ አደረጉ።።

The second rule which start searching a given sentence from the end of the sentence to find trigger keywords and checks to see the word after the keyword is of a named entity or not. The following two sentences are suitable for the second rule considering ከተማ and ድርጅት exists in the lists.

የአዲስ አበባ ከተማነዋሪዎች የውሃ እጥረት እንደገጠማቸውአስታወቁ።።

የተባበሩት መንግስታት ድርጅት ዛሬ አመታዊ በአሉን አከበረ።።

### 4.2.2 Feature Extraction

As we have already discussed in chapter two Section 2.2.3 features extraction is an important step in most of the machine learning applications. The two previously done research in ANER [6][9] focuses solely on finding the most pertinent features that can improve the performance of ANER. Our research also involves optimizing the performance of ML algorithms. But unlike other works we have included the prediction of our simple rule-based system as one feature to the learning algorithm to build a better model.

But after examining the features that [6] and [9] to optimize the performance of their system we were convinced that the addition of other features is needed. As we are going to include the prediction of the rule-based system as a feature we have also included a feature that tells whether a word is a noun or not. Below are the features that we have used in our experiment. The source code used to extract those features can be found at appendix D.

- words including previous and next words.
- suffix and prefix of words.
- POS tag of words.
- NE tag predicted from the rule-based system.
- previous and next NE tag from the corpus.
- nominal flag which indicates whether a given word is noun or not.

The pseudo code used for the feature extraction algorithm is shown below.

| Algorithm 4.3 | Feature Extraction |
|---|---|

```
FeatureExtraction(word:Words,tag: Tags)
        // current word features
        Concatenate current word feature

        Concatenate current word NE tag

        Concatenate current word POS tag

        Concatenate current word's NE information =Rule-based(word)

          If current word length greater than one then

              Concatenate prefix(n) of current word

              Concatenate suffix(n) of current word
          End if
      // Context features of current word

      Concatenate window(n) of previous words

      Concatenate window(n) of next word

      Concatenate previous word NE tag

      Concatenate next word NE tag

      Concatenate previous word POS tag

      Concatenate next word POS tag

      If current word is noun then
      Concatenate 1
      Else
      Concatenate 0
      End if


End FeatureExtraction
```

# Chapter Five
# 5. Experimentation

In this chapter we are going to discuss the experiments that we have conducted to explore the performances of the decision tree and SVM machine learning (ML) algorithms using NE information from the rule-based component as a feature with other additional ones. We start this chapter by describing the environment in which we have conducted our experiments. Then we will show the experiments conducted using different scenarios. Finally we will discuss the results of the experiments.

## *5.1 Experimental Setup*

When conducting our experiments we found out that the Machine learning(ML) algorithms that we have used performs differently depending on the specification of the Hardware we have used. Table 5.1depicts the specification of hardware and software.

| **System:** | |
|---|---|
| Manufacturer | Toshiba |
| Model | Satellite L55-A |
| Processor | Intel(R) Core(TM) i5-3337U CPU @ 1.80GHz 1.80GHz |
| Memory | 8.00GB (7.88 GB usable) |
| System type | 64-bit Operating system, x64 based processor |
| Operating System | Windows 8 |

**Table 5.1 Hardware and software used**

The data set that we have obtained from [9] is imbalanced, that is some classes have higher instances whereas others have low. According to [51] most classifiers perform poorly with imbalanced data since machine learning algorithms generalize sample data and output the simplest hypothesis that best fits the data. [51] also stated that with imbalanced data the simplest hypothesis most of the time classifies all instances as classes which have higher number of instances. For these reasons we have decided to balance our data using SMOTE analysis which is available as a preprocessing tool in WEKA. [52] discusses the use of SMOTE for imbalanced datasets.

The data set that we have obtained from [9] consists of a total of 12191 words (i.e. tokens) which are tagged as PERSON, ORGANIZATION,LOCATION, and O (i.e. others). Each class has the following number of instances: 9899, 389, 1267, 636 for O,

PERSON, ORGANIZATION, LOCATION respectively. And after we have performed SMOTE using the default parameters as provided in WEKA 10 times, to make sure that each class have better representative than before, we have obtained a total of 31347 data sets. And each class has the following number of instances: 9899, 6224, 10136, 5088 for O, PERSON, ORGANIZATION, LOCATION respectively.

We have conducted a total of 6 experiments including the baseline for each of the two algorithms that we have used, J48 and Libsvm, with their default parameters as provided by WEKA. The rule based NE prediction is used as a feature together with other features with these algorithms in our experiments to explore the performance of the hybrid ANER. The POS tagger model built using apache's openNLP has an accuracy of 77% as it is calculated by the tool. We have calculated the performance of the rule-based component using a total of 3962 words which is almost 40% of the corpus we have used for the hybrid ANER. The rule-based component has an f-measure of 50% performance and The source code used for calculating the performance evaluation can be found at appendix F.

## *5.2 Experimental Scenarios and Results*

The two previous researches conducted for Amharic language [6] and [9] explored different feature sets to analyze their impacts on the performance of the ML algorithm used to train and build a NER model. They both concluded that features play a major role in having a better performing NER model. Based on their arguments we have also decided to conduct our experiments by having different experimental scenarios based on the feature sets we have used.

Before dividing the experiments into different scenarios we have chosen feature sets for the baseline experiment which can act as a benchmark to compare other experiments conducted. We have chosen to use [9]'s optimal feature sets as it has performed well compared to [6]. These are the features we have used as baseline feature sets:

- previous and next words.
- previous and next words' NE tags from the corpus.
- previous and next words' prefix and suffix with maximum length 2.

| Algorithms | Performance | | |
|---|---|---|---|
| | Recall % | Precision % | F1 % |
| J48 | 94.5 | 94.5 | 94.5 |
| SVM | 85.7 | 85.9 | 85.6 |

**Table 5.2 Baseline experiment**

After we have selected the baseline feature sets that we have used throughout our experiments. The next experimental scenario included the NE information predicted from the rule-based part as a feature to the baseline feature sets. We have chosen to use the previous and next word's NE type predicted.

| Algorithms | Performance | | |
|---|---|---|---|
| | Recall % | Precision % | F1 % |
| J48 | 93.2 | 93.3 | 93.3 |
| SVM | 81.2 | 81.2 | 80.9 |

**Table 5.3 Experiment with rule**

In the second experimental scenarios we have chosen to experiments the hybrid NER model built in previous experiments by including POS information for the previous and next word and also including the binary feature that indicates whether a given word is a noun or not. Following this experiments we have excluded the binary feature from the second experiment feature sets to explore which feature sets ( the POS or binary feature) play a major role in the identification of NEs.

| Algorithms | Performance | | |
|---|---|---|---|
| | Recall % | Precision % | F1 % |
| J48 | 95 | 95 | 95 |
| SVM | 82.9 | 82.9 | 82.6 |

**Table 5.4 Experiment with rule, POS, and nominal flag**

| Algorithms | Performance | | |
|---|---|---|---|
| | Recall % | Precision % | F1 % |
| J48 | 94 | 94 | 94 |
| SVM | 81.3 | 81.3 | 80.9 |

**Table 5.5 Experiment with rule and POS features**

In the fourth and fifth experimental scenarios we have dropped the rule-based feature to compare the performance of the machine learning components using the POS and the binary feature without the rule-based feature. The first experiment includes both the POS information and the nominal flag value feature and the second excludes the nominal flag value feature.

| Algorithms | Performance | | |
|---|---|---|---|
| | Recall % | Precision % | F1 % |
| J48 | 96.1 | 96.1 | 96.1 |
| SVM | 86.1 | 86.3 | 85.9 |

**Table 5.6 Experiment without rule but with POS and nominal features**

| Algorithms | Performance | | |
|---|---|---|---|
| | Recall % | Precision % | F1 % |
| J48 | 94.8 | 94.9 | 94.8 |
| SVM | 85.6 | 85.7 | 85.4 |

**Table 5.7 Experiment without rule but with only POS feature**

We will now use bar charts to clearly show the results obtained from the experiments. For having a better insight into the performance of each of the algorithms the bars depicts the performance of the algorithms from the precision, recall, and F-measure perspective and the class classification accuracy perspective taken from the confusion matrix generated by WEKA.

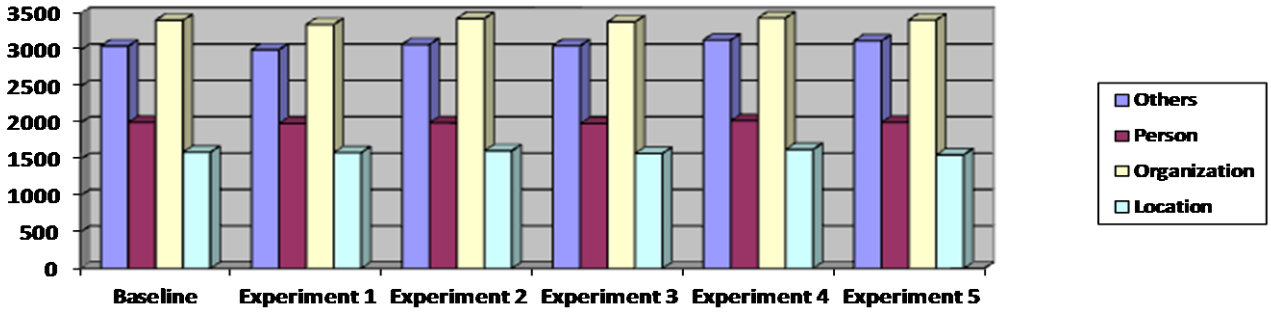


Figure 5.1 J48's classification accuracy for each class

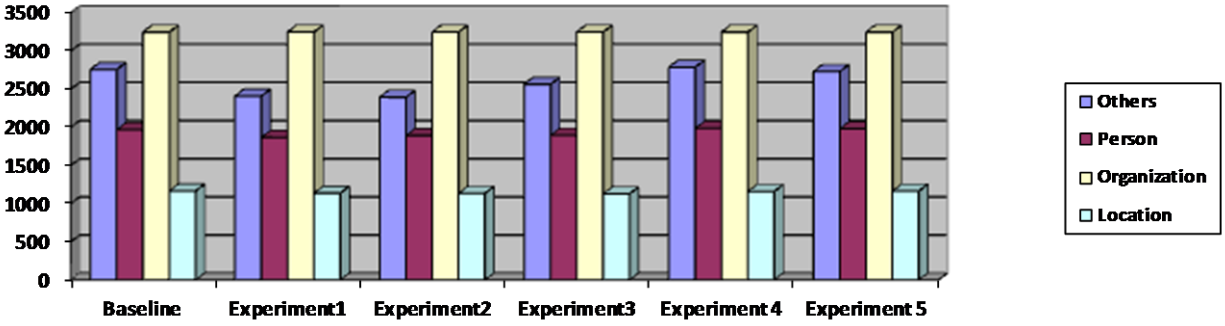


Figure 5.2 Libsvm's classification accuracy for each class

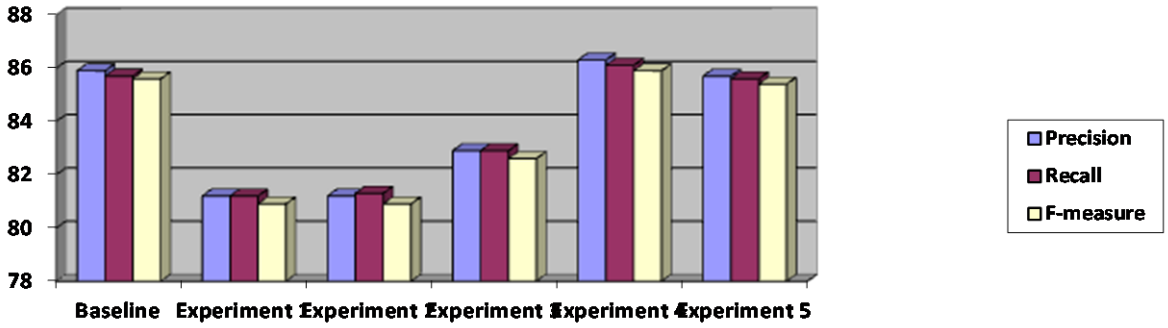


Figure 5.3 J48's Precision, Recall, and F-measure

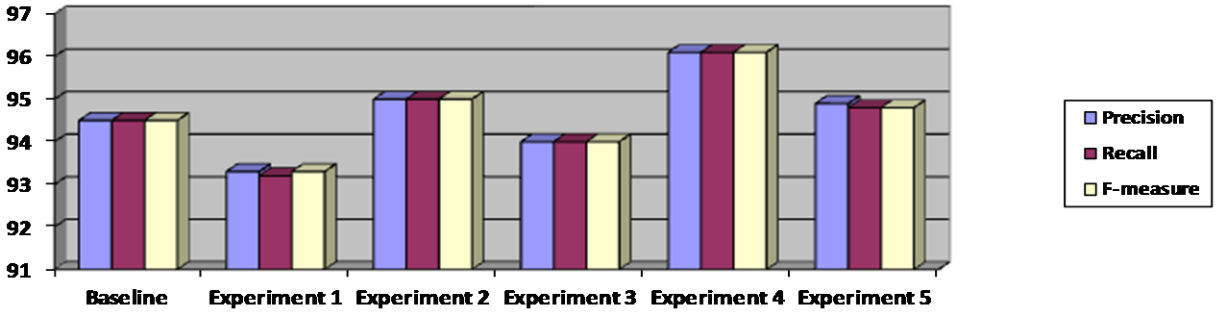


**Figure 5.4 Libsvm's Precision, Recall, and F-measure**

## *5.4 Discussion*

Five different kinds of experiments are conducted to evaluate the performance of the proposed hybrid NER system. The experiments were conducted on a total of 31347 words which consists of 9899,6224, 10136,vand 5088 number of words tagged as O (i.e. others), PERSON,ORGANIZATION, and LOCATION respectively. These data are obtained after performing SMOTE analysis 10 times to balance the dataset.

The results obtained from the baseline experiment was significant in terms of the precision, recall and F1 measure as compared with the two previously conducted researches, which has obtained an f-measure of 74.61% and 80.66%[6] and [9].

Following the baseline experiment we have conducted five different experiments to compare the feature sets that we have used for the machine learning NER model and the hybrid model. For the machine learning and also for the hybrid experiments J48 outperforms the libsvm algorithm. As we can see in the charts in the previous section J48 tried to classify accurately each of the instances according to their respective classes.

When we look closely on the performance of the hybrid NER model the results showed that the rule-based feature we have used in both of the algorithms does not allow the performances to increase. Even though the results obtained for both J48 and libsvm algorithms performs better when compared to previous works but the other feature sets we have used outperformed it significantly. The reason why the performance of the

hybrid NER model is low as compared with the ML model is that the algorithms that we have used to build the rule-based component is simple because it bases its prediction only on the presence of trigger words. But the algorithms can perform better if we include word lists that goes together with proper nouns.

The experiments conducted for the pure machine learning NER model scored a higher performance for ANER. For the ML experiments two features played a major role on the performance of the models, they are POS information feature and the nominal value flag feature. The nominal value flag is not used in previous researches and this is the first research which has attempted to use this feature. As we can see from the charts and tables this feature played a significant role for both approaches (ML and hybrid). Experiments 2 and 3 and also experiments 4 and 5 is conducted to test the performance of NER systems that use the nominal value flag feature with or without POS information and rule-based feature. As it is shown in the experimental results depicted using tables and charts our best performing model is obtained when this feature is used with that of the POS information. [9] dropped the POS information from his experiments for the reason that POS model are built with the same feature as that of NER model but in our case POS information, like the work of [6], played a major role in performance improvement of ANER model.

Due to time limitation we have not conducted further experiments considering some of the features for the current word such as whether a given word is at the beginning or ending position of a sentence together with the feature sets that we have used throughout our experiments. But these features should be further explored using these ML algorithms as we have seen from this and previous researchers' experiences a different combination of features should be explored to find the optimal feature sets for ANER. Moreover we can say that using rule-based feature predicted from the algorithms we have built has not shown a significant result as compared with the other feature sets used without it as a result of the poor performance of our rule-based system. But the nominal value flag feature together with POS features has shown a significant increase in performance for the ML based ANER model.

# Chapter Six
# 6.Conclusions and Recommendations

## *6.1 Conclusions*

NER is the task of detecting and classifying NEs, which are unique identifiers of proper nouns, times and measurements. NER has found many application areas in NLP, among them are IE, Question and Answering, Machine Translation, Text Summarizations and others.

NLP is one of the fields that is getting popularity among academicians in Ethiopia. Most of the works that has being conducted are for Amharic, Afan Oromo and Tigrigna languages as most resources are available for these languages. Among the researches done for Amharic language is NER by [6] and [9] that recognize and classify NEs using the conditional random field (CRF) machine learning algorithm. The highest score they have obtained using the F-measure are 74.61% and 80.%, which can be considered as promising considering the size of the corpus used.

Our research also explored Amharic NER (ANER) but unlike the works that have already been done in this area our works explored ANER by using the hybrid approach. We have used the Walta Information Center (WIC) corpus[53] which we have obtained from [9]. The corpus is already annotated with Person, Organization, Location and Other (i.e. refers to any word that is not the other three NE types.) Named entity types and consists of a total of 12191 tokens.

The rule-based component of our system is built by adopting the two rules used for building a rule-based NER for Arabic [32] and has a performance f-measure of 50%.We have also built a part of speech(POS) tagging model using apache's openNLP tool with an accuracy of 77%. The POS tagger is used extensively with the rule-based component for identifying and classifying NEs.

In order to use the rule-based algorithm we have retagged the WIC corpus to incorporate a tag for proper nouns ( i.e. NNP). We have also extracted features from the given corpus to train and evaluate our learning algorithms. Before building our model the data that we have obtained from [9] is imbalanced so by using WEKA's SMOTE

preprocessing tool we have tried to balance our data by performing SMOTE 10 times on the data sets.

In order to conduct our experiments we have divided our experiments into different scenarios based on the feature sets used. We have adopted the feature sets that we have used from previous researchers with the exception that we have used a rule-based feature and also experimented with one new feature, which is the nominal value flag feature. After selecting our baseline feature sets we have conducted different experiments to explore the performance of our hybrid NER system in combination with different feature sets. Then finally we have discovered that the hybrid NER model has not made a significant performance improvement as compared with the pure ML NER model. Therefore, from the results of the experiments we have conducted we have concluded the following:

- The hybrid approach for ANER using the NE type predicted, from the rule-based component we have built, as a feature has not improved the performance of ANER.
- The POS feature used in our experiment has increased the performance of ANER.
- The nominal flag value feature when used in combination with the POS information has shown a significant performance improvement.
- The two ML algorithms namely SVM and J48 we have used in our experiments has resulted in a better performance as compared with the CRF ML algorithms which was used by previous researchers.
- Among the two ML algorithms J48 outperformed SVM in terms of accuracy.

## *6.2 Recommendation*

This study showed that NER can be solved using the hybrid approach but in order to have a robust ANER systems further works are required in the field..

- The rule-based component we have built has low accuracy and from the literatures reviewed for resource poor languages, linguistic rules plays a vital role in obtaining high performing Natural Language Processing (NLP) applications.

So there is a need to prepare list of gazetteers and rules for identification of NEs. For these works to be successful linguist should participate in these tasks.

- The datasets we have used is imbalanced. We have tried to balance it using WEKA's SMOTE tool. But a more balanced data in which each class has a better representative instances should be prepared and used for building a better and more robust NER.
- Our research focuses only on the detection and classification of person, organization, and location names but time and measurement are also essential NE types. So further research should be conducted by including not only these NE types but also other application specific NE types.
- From the literature reviewed, NLP applications such as Question and Answering and Machine translation have found out that the use of NER is vital to the performance of their system. Therefore, researches should be conducted for Amharic that uses NER with applications that benefit from it.
- We have integrated the rule-based output as a feature to be used with other feature sets by the machine learning algorithms. But the integration can be done after the ML built a classification model to disambiguate some of the misclassified NEs. So further research should be conducted by applying this approach to hybrid NER.
- Having a robust high performing POS tagger can increase the performance of the hybrid ANER and also the ML based NER.

# References

[1]. Peter Jackson and Isabelle Moulinier, 2002, Natural language processing for online application text retrieval, extraction and categorization,2nd revised edition.

[2]. Daniel Jurafsky & James H. Martin, 2009, Speech and language processing, An Introduction to Natural Language Processing, Computational Linguistic, and Speech Recognition, 2nd Edition, Pearson Prentice Hall, New Jersey.

[3]. Nadeau, D. and Sekine, S, 2007, A Survey of Named Entity Recognition and Classification. *Lingvisticae Investigationes*, 30(1), pages 3-26.

[4]. Mai Mohamed Oudah1 Khaled Shaalan,2012 A Pipeline Arabic Named Entity Recognition Using a Hybrid Approach, Proceedings of COLING 2012: Technical Papers, pages 2159–2176.

[5]. Desalegn Abebaw Zeleke, 2013, LETEYEQ (ልጠየቅ)-A Web Based Amharic Question Answering System for Factoid Questions Using Machine Learning Approach, MSc Thesis,Addis Ababa University, Addis Ababa.

[6]. Moges Ahmed, 2010**,** Named Entity Recognition For Amharic Language, MSc Thesis,Addis Ababa University, Addis Ababa.

[7]. Solomon Assemu, 2011,Unsupervised machine learning approach for word sense disambiguation to Amharic words, MSc Thesis,Addis Ababa University, Addis Ababa.

[8]. Mandefro Legesse Kejela, 2010, Named Entity Recognition for Afan Oromo, MSc Thesis,Addis Ababa University, Addis Ababa

[9]. Besufkad Alemu,2013, A Named Entity Recognition system for Amharic, MSc Thesis,Addis Ababa University, Addis Ababa

[10]. Robert Gaizauskas and Yorick Wilks,Information Extraction: Beyond Document Retrieval: Computational Linguistics and Chinese Language Processing vol. 3, no. 2, August 1998, pp. 17-60

[11]. Douglas E. Appelt,1999, Introduction to information extraction: AI Communications, Volume 12 Issue 3, Pages 161 - 172, IOS Press Amsterdam, The Netherlands.

[12].Ralph Grishman, Information Extraction: Techniques and Challenges.

[13]. Cowie and Lehnert,1996, Information Extraction:Communications of the ACM, Volume 39 Issue 1,Pages 80-91.

[14]. Mónica Marrero, Julián Urbano, Sonia Sánchez-Cuadrado, Jorge Morato, Juan Miguel Gómez-Berbís,2013, Named entity recognition: Fallacies,Challenges, and Opportunities; Computer Standards & Interfaces (482-489).

[15]. Hsin–Hsi Chen, Yung–Wei Ding, and Shih–Chung Tsai,1998, Named Entity Extraction for Information Retrieval: COMPUTER PROCESSING OF ORIENTAL LANGUAGES, VOL. 11, NO. 4.
[16]. Alireza Mansouri, Lilly Suriani Affendey, Ali Mamat,2008,Named Entity Recognition Approaches: IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.2.
[17]. Dasgupta Soumi Sukhendu, Prof. Avani R. Vasant,2002, A Hybrid Named Entity Recognition System Using Natural Language Processing And Decision Tree Learning: Journal Of Information, Knowledge And Research In Computer Engineering Volume – 02.
[18]. Roman Klinger and Katrin Tomanek, 2007,Classical Probabilistic Models and Conditional Random Fields: Algorithm Engineering Report TR07-2-013 ,ISSN 1864-4503.
[19]. Lior Rokach and Oded Maimon, 2005,Data Mining And Knowledge Discovery Handbook, Springer-Verlag New York, Inc. Secaucus, NJ, USA .
[20]. Saher Esmeir and Shaul Markovitch, 2007,Anytime Learning of Decision Trees, Journal of Machine Learning Research 8 (2007) 891-933.
[21]. T. Mitchell, "Decision Tree Learning", in T. Mitchell, *Machine Learning*, The McGraw-Hill Companies, Inc., 1997, pp. 52-78.
[22]. T.Miranda Lakshmi, A.Martin , R.Mumtaj Begum ,and Dr.V.Prasanna Venkatesan,2013, An Analysis on Performance of Decision Tree Algorithms using Student's Qualitative Data: *I*.J.Modern Education and Computer Science, 2013, 5, 18-27.
[23]. Asif Ekbal and Sivaji Bandyopadhyay,2010, Named Entity Recognition using Support Vector Machine: A Language Independent Approach: International Journal of Electrical, Computer, and Systems Engineering 4:2 2010.
[24]. Joel Mickelin,2013, Named Entity Recognition with Support Vector Machines: MSc Thesis, Royal Institute of Technology School of Computer Science and Communication, Stockholm, Sweden.
[25]. Peter Harrington,2012, Machine Learning in Action.
[26]. Wondwossen Mulugeta and Michael Gasser,2012, Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming, Workshop on Language Technology for Normalisation of Less-Resourced Languages
[27]. A.A. Argaw and L. Asker, 2007,An Amharic Stemmer: Reducing words to their citation form, in proceedings of the 2007 workshop on computational approaches to Semitic languages: Common issue and resources, pp.104-110.
[28]. ጌታሁን አማረ ,1989, ዘመናዊ የአማርኛ ስዋስው በቀላል አቀራረብ
[29]. ብላታ መርስዔ ሐዘን ወልደ ቂርቆስ ,1948, የአማርኛ ስዋስው
[30]. ፕሮፌሰር ባየ የማም ,2004, አጭርና ቀላል የአማርኛ ሰዋስው
[31]. NERA, Named Entity Recognition For Arabic Languages, Journal Of The American Society For Information Science And Technology, 60(8):1652–1663, 2009
[32]. Mohammed Aboaoga and Mohd Juzaiddin Ab Aziz, Arabic Person Names Recognition by Using A Rule Based Approach, Journal of Computer Science 9 (7): 922-927, 2013
[34]. Asif Ekbal, Rejwanul Haque, Amitava Das, Venkateswarlu Poka and Sivaji Bandyopadhyay,vLanguage Independent Named Entity Recognition in Indian Language,

IJCNLP-08 Workshop On NER for South and South East Asian Languages Proceedings of the Workshop pages 33-40.
[35]. Asif Ekbal and Sivaji Bandyopadhyay, Bengali Named Entity Recognition using Support Vector Machine, IJCNLP-08 Workshop On NER for South and South East Asian Languages Proceedings of the Workshop pages 51-58.
[36]. Praneeth M Shishtla, Karthik Gali, Prasad Pingali and Vasudeva Varma ,Experiments in Telugu NER: A Conditional Random Field Approach,IJCNLP-08 Workshop On NER for South and South East Asian Languages Proceedings of the Workshop pages 105-111.
[37]. Sujan Kumar Saha et al A Hybrid Approach for Named Entity Recognition in Indian Languages, IJCNLP-08 Workshop On NER for South and South East Asian Languages Proceedings of the Workshop pages 17-24.
[40]. P. Thompson and C. Dozier,1997, "Name searching and information retrieval," in Proceedings of Second Conference on Empirical Methods in Natural Language Processing, Providence, Rhode Island.
[41]. Martin Hassel,2003, Exploitation of Named Entities in Automatic Text Summarization for Swedish, In Proceedings of NODALIDA 03 - 14 th Nordic Conference on Computational Linguistics, May 30-31.
[42]. Kashif Riaz,2010,Proceedings of the 2010 Named Entities Workshop, ACL 2010, pages 126–135,Uppsala, Sweden,
[43] Abhilash Kumar Srivastava, Krishnendu Ghosh,2013, An English Name Entity Recognition System, International Conference on Computer Science and Information Technology; ISBN: 978-93-82208-87-7.
[44]. Bowen Sun,2010, Named entity recognition Evaluation of Existing Systems, MSc Thesis In Information Systems, Norwegian University of Science and Technology.
[45]. Munish Minia,2012,Named Entity Recognition: Literature Survey,Clift.
[46]. Ethem Alpaydin,Introduction to Machine Learning,2010, MIT press
[47]. Rehan Akbani1, Stephen Kwek1, Nathalie Japkowicz2, 2009,Knowledge and Data Engineering, IEEE Transactions on Volume:21 , Issue: 9
[48] http://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/
[49] Mr. Brijain R Patel and Mr. Kushik K Rana, 2013,A Survey on Decision Tree Algorithm for Classification, International Journal Of Engineering Development And Research.
[50]. http://www.mcit.gov.et/web/english/ict-sector-development-in-ethiopia
[51] Rehan Akbani, Stephen Kwek,2004, Nathalie Japkowicz, Applying Support Vector Machines to Imbalanced Datasets, Lecture Notes in Computer Science Volume 3201,pp 39-50.
[52]. Nitesh V. Chawla, 2010, Data Mining For Imbalanced Datasets:An Overview, Data Mining And Knowledge Discovery Handbook, 2nd ed pp 853-867.
[53]Girma Awgichew Demeke, et.al. (2006). Manual Annotation of Amharic News Items with Part-of-Speech Tags and its Challenges. Ethiopian Languages Research Center of Addis Ababa University.
[54] htttp://www.cs.waikato.ac.nz/ml/weka/ [55].
[55]https://opennlp.apache.org/documentation/1.5.2-incubating/manual/opennlp.html

# APPENDICES

## *Appendix A: Data Prepared to train the ANER*

```
%AmharicNER
%arff file created by:Mikiyas Tadele from Besufkad's manually tagged Training data
@relation NERTagging


%current word features
@attribute cword String
@attribute cwordPrefix String
@attribute cwordSuffix String
@attribute cwordNETag String
@attribute cwordPOSTag String
@attribute cwordNEFromRule String
%previous words of window size 4
@attribute pword1 String
@attribute pword2 String
@attribute pword3 String
@attribute pword4 String
%previous words of window size 4
@attribute nword1 String
@attribute nword2 String
@attribute nword3 String
@attribute nword4 String
%previous and next NE tag
@attribute pwordNETag String
@attribute nwordNETag String
% previous and next POS tags
@attribute pwordPOSTag String
@attribute nwordPOSTag String
%class
@attribute class {O,PERSON,ORGANIZATION,LOCATION}

@data

yegambEla, ye, la, ORGANIZATION, NNP, LOCATION, 0, 0, 0, 0, mrmr, maIkel,
hulet, yetexaxalu, 0, ORGANIZATION, 0, NNP, ORGANIZATION
mrmr, mr, mr, ORGANIZATION, NNP, O, yegambEla, 0, 0, 0, maIkel, hulet,
yetexaxalu, yebeqolona, ORGANIZATION, ORGANIZATION, NNP, NNP,
ORGANIZATION
```

## *Appendix B: Data prepared to train the POS tagger*

likesetu_V Indemiclu_VP gnzabE_N asCebTWal_V ::_PUNC
Indihum_PRONP yesratu_NP adegawoc_N likesetu_V yemiclubetn_VREL qedadawoc_N lemedfenna_NPC beqeTay_ADJP ametat_N wsT_NNP drjtu_N lemiyakenawnacew_VP srawoc_NNP aqTaCa_N yeteyazebet_VREL mehonun_VN amelktewal_V ::_PUNC
gubaEw_N bedrjtu_NP wesT_PREP tekesto_V yeneberewn_VREL zqTet_N feTan_ADJ behone_VP menged_N lemeqrefna_NPC drjtu_N leyazacew_VP hzbawi_ADJ programoc_N meseret_N yeTale_VREL Indehonem_VPC tequmewal_V ::_PUNC
begubaEw_NP lay_PREP bekbr_NP Ingdnet_N yetesatefut_VREL yemIrab_NNP herergE_NNP tmhrt_NNP memriya_NNP halafi_NNP ato_ADJ zeynu_NNP mehemed_NNP bebekulacew_NP yedrjtu_NP yeasr_NUMP amet_N hidet_N riport_N beteTenakerena_VPC glInet_N betemolabet_VP hunEta_N meqrebun_VN tequmew_V ;_PUNC bewsTu_NP betekatetu_VP neTboc_N lay_PREP abalatu_N yakahEdut_VREL krkrna_NC wyyt_N dEmokrasiyawi_NNP akahEd_N yetemolabet_VREL Indeneber_VP asredtewal_V ::_PUNC
drjtu_N beasr_NUMP amet_N hidet_N wsT_NNP kagaTemut_VP yeSere-dEmokrasiyawi_ADJ aserar_N yeamelekaket_NP Trat_N megWadelna_NC yemusna_NPC cgroc_N bemenesat_NP beqeTayu_ADJP hulet_NUMCR lemakenawen_NP yaqedacewn_VREL srawocm_NC yeohdEdn_NNP teTenakro_V mewTat_VN yemiyamelakt_VREL new_AUX blewal_V ::_PUNC
drjtun_N cgroc_N IndiyaweTut_VP Imnet_N yeteTalebacew_VREL temeraC_ADJ komitEwoc_N yekllun_NP mengstawi_NNP mewaqr_N keSedeqew_VP Iqd_N gar_PREP bemismama_VP melku_N mastekakelna_NC bedrjtu_NP wsT_NNP ytayu_V yeneberutna_VPC yeamerar_NP cgroc_N qeTaynet_N balew_VP melku_N meftat_VN IndemiTebeqbacew_VP ato_ADJ zeynu_NNP mekrewal_VN ::_PUNC
lElaw_ADJ yekbr_NP Ingdana_NC asteyayet_N seCi_NNP besemEn_NNP xewa_NNP zon_NNP yewha_NNP maIdnna_NNP Inerji_NNP memriya_NNP halafi_NNP ato_ADJ kebede_NNP gerba_NNP bebekulacew_NP beIyandandu_PRONP guday_NNP lay_PREP yegubaEw_NP tesatafiwoc_N yemeselacewn_VREL hesab_N beneSanet_NP yeseTubetna_VPC kalefew_VP shtet_N lememar_NP yalacew_VREL zgjunet_NNP drjtun_N yemiyaTenakrew_VREL blewal_V ::_PUNC
gubaEw_N yemereTacew_VREL komitEwoc_N bqat_N yalacewna_VPC betgl_NP wsT_NNP bzu_ADJ gizE_N yasalefu_VREL qedem_sil Indetayew_VP yemiyagaTm_VREL shtetn_N lEla_ADJ akal_N sayTebqu_VP bemaremna_NPC drjtun_N beras_PRONP metemamen_VN bemadreg_NP leoromo_NP hzb_N tqm_N betkkl_NP metagel_VN IndemiTebeqbacew_VP astawqewal_V ::_PUNC
be4Naw_NUMP yeihadEg_NNP gubaE_N abal_NNP drjtocu_NNP yeasr_NUMP amet_N hidetacewn_N gemgmewna_VC beageritu_NP yemigenebawn_VREL srat_N bemifelegew_VP menged_N kemaramed_NP anSar_N yehonu_VREL cgrocn_N leytew_V yemiyakahEdut_VREL bemehonu_NP kemngizEwm_NPC bebeleTe_VP teqami_ADJ wsanEwocn_N ytelalefalu_VREL blew_VP IndemiTebqu_VP asteyayet_N seCiwocu_N tenagrewal_V ::_PUN

## *Appendix C: Trigger words (in transliterated format) used for the Rule-based system*

| Person Trigger words | Location Trigger words | Organization Trigger words |
|---|---|---|
| Ato, gax, artist<br>Asr, amsa, meto, aleqa, shaleqa, artist, redat, profiser, ambasader, prEzidant,mister, temeramari, keato, teqlay, ministr, dokte, weyzero, yewCi, guday jEnEral, kbur, atlEt weTat, komixner,azegaj, gazETeNa, astedadari tekesax, ners, weyzerit | Ketema, kll, ager, zon, kfleketema. | Drjt, universti, mahkel, mestedader, biro,balesltan,komixn, mestedader. |

## *Appendix D: Source Code of Feature extraction*

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package ner_annotation;

import java.io.InputStream;
import java.util.*;
import java.util.ArrayList;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSSample;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.postag.WordTagSampleStream;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.TrainingParameters;

/**
 *
 * @author Mikiyas
 * This class is called using its default constructor. The class extract features for a given word like the word's POS tag, its prefix and suffix length
 * and so on. The class also uses the POS tag model generated using apaches's open NLP tool further documentation on how to use this tool can be found at
 * https://opennlp.apache.org/documentation/1.5.2-incubating/manual/opennlp.html
 */
public class NERFeatureExtractor {
    private InputStream modelIn = null;
    private POSModel model = null;
    private POSTaggerME tagger = null;

    public NERFeatureExtractor() {

        try {
            modelIn = new FileInputStream("C:\\Users\\Mikiyas\\Downloads\\Compressed\\apache-opennlp-1.5.3-bin\\apache-opennlp-1.5.3\\bin\\am-pos-maxent2
            model = new POSModel(modelIn);
        } catch (IOException e) {
            // Model loading failed, handle the error
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
                }
            }
        }

        tagger = new POSTaggerME(model);
    }

    public String opennlpPosTag(String word) throws IOException {
        //String sampleSentence=word;
        String[] wa = new String[1];
        wa[0] = word;
        String[] tags = tagger.tag(wa);
        modelIn.close();
        return tags[0];

    }

    public HashMap opennlpPosTag1(String[] word) throws IOException {
        //String sampleSentence=word;
        String[] wa = word;
        HashMap wt = new HashMap();
```

```
        String[] tags = tagger.tag(wa);
        modelIn.close();
        for (int i = 0; i < tags.length; i++) {
            wt.put(wa[i], tags[i]);
        }
        return wt;

    }

    public String getPrefix(int len, String word) {
        if (word.length() > len) {
            return word.substring(0, len);
        } else if (word.length() == len) {
            return word.substring(0, len - 1);
        } else {
            return "0";
        }


    }

    public String getSuffix(int len, String word) {
        if (word.length() > len) {
            return word.substring(word.length() - len, word.length());
        } else if (word.length() == len) {
            return word.substring(word.length() - (len - 1), word.length());
        } else {
            return "0";
        }
    }

    public String[] getPreviousWords(int windowSize, int index, ArrayList<String> corpus) {
        String[] words = new String[windowSize];
        int j=0;
     if (index == 0) {
         for (int i = 0; i < windowSize; i++) {
             words[i] = "0";
         }

     } else {
         index = index - 1;

         while(j<windowSize)
         {
             if(index>-1)
             words[j]=corpus.get(index);
             else
               words[j]="0";

             index=index-1;
             j=j+1;
         }
       }

     return words;

  }

  public String[] getNextWords(int windowSize, int index, ArrayList<String> corpus) {
      String[] words = new String[windowSize];
      int j=0;
      if (index==corpus.size()-1) {
          for (int i = 0; i < windowSize; i++) {
              words[i] = "0";
          }

      } else {
          index = index + 1;
```

```
          }

        return words;

    }

    public String[] getNextWords(int windowSize, int index, ArrayList<String> corpus) {
        String[] words = new String[windowSize];
        int j=0;
        if (index==corpus.size()-1) {
            for (int i = 0; i < windowSize; i++) {
                words[i] = "0";
            }

        } else {
            index = index + 1;

            while(j<windowSize)
            {
                if(index<=corpus.size()-1)
                words[j]=corpus.get(index);
                else
                  words[j]="0";

                index=index+1;
                j=j+1;
            }
          }

        return words;

    }
}
```

# *Appendix E: Source Code of Rule-Based Component*

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package ner_annotation;

import java.io.InputStream;
import java.util.*;
import java.util.ArrayList;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSSample;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.postag.WordTagSampleStream;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.TrainingParameters;

/**
 *
 * @author Mikiyas
 * This class is called using its default constructor. The class extract features for a given word like the word's POS tag, its prefix and suffix length
 * and so on. The class also uses the POS tag model generated using apaches's open NLP tool further documentation on how to use this tool can be found at
 * https://opennlp.apache.org/documentation/1.5.2-incubating/manual/opennlp.html
 */
public class NERFeatureExtractor {
    private InputStream modelIn = null;
    private POSModel model = null;
    private POSTaggerME tagger = null;

    public NERFeatureExtractor() {

        try {
            modelIn = new FileInputStream("C:\\Users\\Mikiyas\\Downloads\\Compressed\\apache-opennlp-1.5.3-bin\\apache-opennlp-1.5.3\\bin\\am-pos-maxent2
            model = new POSModel(modelIn);
        } catch (IOException e) {
            // Model loading failed, handle the error
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
                }
            }
        }

        tagger = new POSTaggerME(model);
    }

    public String opennlpPosTag(String word) throws IOException {
        //String sampleSentence=word;
        String[] wa = new String[1];
        wa[0] = word;
        String[] tags = tagger.tag(wa);
        modelIn.close();
        return tags[0];

    }

    public HashMap opennlpPosTag1(String[] word) throws IOException {
        //String sampleSentence=word;
        String[] wa = word;
        HashMap wt = new HashMap();
```

```java
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package ner_annotation;

import java.io.InputStream;
import java.util.*;
import java.util.ArrayList;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import opennlp.tools.postag.POSModel;
import opennlp.tools.postag.POSSample;
import opennlp.tools.postag.POSTaggerME;
import opennlp.tools.postag.WordTagSampleStream;
import opennlp.tools.util.ObjectStream;
import opennlp.tools.util.PlainTextByLineStream;
import opennlp.tools.util.TrainingParameters;

/**
 *
 * @author Mikiyas
 * This class is called using its default constructor. The class extract features for a given word like the word's POS tag, its prefix and suffix length
 * and so on. The class also uses the POS tag model generated using apaches's open NLP tool further documentation on how to use this tool can be found at
 * https://opennlp.apache.org/documentation/1.5.2-incubating/manual/opennlp.html
 */
public class NERFeatureExtractor {
    private InputStream modelIn = null;
    private POSModel model = null;
    private POSTaggerME tagger = null;

    public NERFeatureExtractor() {

        try {
            modelIn = new FileInputStream("C:\\Users\\Mikiyas\\Downloads\\Compressed\\apache-opennlp-1.5.3-bin\\apache-opennlp-1.5.3\\bin\\am-pos-maxent2
            model = new POSModel(modelIn);
        } catch (IOException e) {
            // Model loading failed, handle the error
            e.printStackTrace();
        } finally {
            if (modelIn != null) {
                try {
                    modelIn.close();
                } catch (IOException e) {
                }
            }
        }

        tagger = new POSTaggerME(model);
    }

    public String opennlpPosTag(String word) throws IOException {
        //String sampleSentence=word;
        String[] wa = new String[1];
        wa[0] = word;
        String[] tags = tagger.tag(wa);
        modelIn.close();
        return tags[0];

    }

    public HashMap opennlpPosTag1(String[] word) throws IOException {
        //String sampleSentence=word;
        String[] wa = word;
        HashMap wt = new HashMap();
```

```
        perkeywords.add("keato");
        perkeywords.add("teqlay");
        perkeywords.add("ministr");
        perkeywords.add("dokter");
        perkeywords.add("weyzero");
        perkeywords.add("yewCi");
        perkeywords.add("guday");
        perkeywords.add("jEnEral");
        perkeywords.add("kbur");
        perkeywords.add("atlEt");
        perkeywords.add("weTat");
        perkeywords.add("komixner");
        perkeywords.add("azegaj");
        perkeywords.add("gazETeNa");
        perkeywords.add("astedadari");
        perkeywords.add("tekesax");
        perkeywords.add("ners");
        perkeywords.add("weyzerit");
        perkeywords.add("ase");
    }

    private void populateLocKeyWords() {
        lockeywords.add("ketema");
        lockeywords.add("kll");
        lockeywords.add("ager");
    }

    private void populateOrgKeyWords() {
        orgkeywords.add("drjt");
        orgkeywords.add("universti");
        orgkeywords.add("maIkel");
        orgkeywords.add("mestedadr");
        orgkeywords.add("baleslTan");
        orgkeywords.add("komixn");
        orgkeywords.add("biro");

    }

    public void personRule1(String sen) throws IOException {
        String[] tokens = sen.split("\\s+");
        NERFeatureExtractor ne = new NERFeatureExtractor();
        int pos = isTriggerWordsRule1(tokens, perkeywords);
        if (pos == -1) {
            return;
        } else {
            for (int i = pos + 1; i < tokens.length; i++) {
                if (perkeywords.contains(tokens[i])) {
                    String newSen = getNewString(tokens, i);
                    personRule1(newSen);
                } else {
                    if (!(ne.opennlpPosTag(tokens[i]).equals("NNP"))) {
                        return;
                    } else {
                        word_namedEntityMap.put(tokens[i], "PERSON");
                    }
                }
            }
        }

    }

    private String getNewString(String[] tokens, int sIndex) {
        String newString = "";
        for (int i = sIndex; i < tokens.length; i++) {
            newString += tokens[i] + " ";
        }
        return newString.trim();
```

```
    }

    public void personRule2(String sen) throws IOException {

        String[] tokens = sen.split("\\s+");
        int pos = isTriggerWordsRule1(tokens, perkeywords);
        if (pos == -1) {
            return;
        } else {
            HashMap wt = nerf.opennlpPosTag1(tokens);
            for (int i = pos - 1; i >= 0; i--) {
                if (perkeywords.contains(tokens[i])) {
                    String newSen = getNewString(tokens, i);
                    personRule2(newSen);
                } else {
                    if (!wt.get(tokens[i]).equals("NNP")) {
                        return;
                    } else {
                        word_namedEntityMap.put(tokens[i], "PERSON");
                    }

                }
            }

        }
    }

    public void OrgRule1(String sen) throws IOException {

        String[] tokens = sen.split("\\s+");
        int pos = isTriggerWordsRule1(tokens, orgkeywords);
        if (pos == -1) {
            return;
        } else {
            HashMap wt = nerf.opennlpPosTag1(tokens);
            for (int i = pos - 1; i >= 0; i--) {
                if (orgkeywords.contains(tokens[i])) {
                    String newSen = getNewString(tokens, i);
                    OrgRule1(newSen);
                } else {
                    if (!wt.get(tokens[i]).equals("NNP")) {
                        return;
                    } else {
                        word_namedEntityMap.put(tokens[i], "ORGANIZATION");
                    }

                }
            }

        }
    }

    public void locationRule1(String sen) throws IOException {
        String[] tokens = sen.split("\\s+");
        int pos = isTriggerWordsRule1(tokens, lockeywords);
        if (pos == -1) {
            return;
        } else {
            HashMap wt = nerf.opennlpPosTag1(tokens);
            for (int i = pos - 1; i >= 0; i--) {
                if (lockeywords.contains(tokens[i])) {
                    String newSen = getNewString(tokens, i);
                    locationRule1(newSen);
                } else {
                    if (!wt.get(tokens[i]).equals("NNP")) {
                        return;
                    } else {
                        word_namedEntityMap.put(tokens[i], "LOCATION");
```

```
    public void locationRule1(String sen) throws IOException {
        String[] tokens = sen.split("\\s+");
        int pos = isTriggerWordsRule1(tokens, lockeywords);
        if (pos == -1) {
            return;
        } else {
            HashMap wt = nerf.opennlpPosTag1(tokens);
            for (int i = pos - 1; i >= 0; i--) {
                if (lockeywords.contains(tokens[i])) {
                    String newSen = getNewString(tokens, i);
                    locationRule1(newSen);
                } else {
                    if (!wt.get(tokens[i]).equals("NNP")) {
                        return;
                    } else {
                        word_namedEntityMap.put(tokens[i], "LOCATION");
                    }
                }
            }

        }
    }

    private int isTriggerWordsRule1(String[] sen, ArrayList<String> triggerList) {

        for (int i = 0; i < sen.length; i++) {
            if (triggerList.contains(sen[i])) {
                return i;
            }
        }
        return -1;
    }
}
```

## *Appendix F: Source Code of Rule-Based Component Performance Evaluation*

```
66          for(int i=0;i<sentences.size();i++)
67          {
68              String[] line=sentences.get(i).split("-");
69              if(line[1].equals(line[2]))
70              {
71                if(line[1].equals("PERSON"))
72                    tPerson=+1;
73                else if(line[1].equals("ORGANIZATION"))
74                    tOrganization=+1;
75                else if(line[1].equals("LOCATION"))
76                    tLocation=+1;
77                else
78                    tO=+1;
79              }
80              else if(!line[1].equals(line[2]))
81              {
82                  if(!line[1].equals("PERSON") && line[2].equals("PERSON"))
83                      fPerson=+1;
84                  else if(!line[1].equals("ORGANIZATION") && line[2].equals("ORGANIZATION"))
85                      fOrganization=+1;
86                  else if(!line[1].equals("LOCATION") && line[2].equals("LOCATION"))
87                      pLocation=+1;
88                  else if(!line[1].equals("O") && line[2].equals("O"))
89                      pO=+1;
90                  if(line[1].equals("PERSON") && !line[2].equals("PERSON"))
91                      fnPerson=+1;
92                  else if(line[1].equals("ORGANIZATION") && !line[2].equals("ORGANIZATION"))
93                      fnOrganization=+1;
94                  else if(line[1].equals("LOCATION") && !line[2].equals("LOCATION"))
95                      fnLocation=+1;
96                  else if(line[1].equals("O") && !line[2].equals("O"))
97                      fnO=+1;
 99  average_precision=(tPerson+tLocation+tOrganization+tO)/(tPerson+tLocation+tOrganization+tO+fPerson+pLocation+fOrganization+pO);
100  average_recall=(tPerson+tLocation+tOrganization+tO)/(tPerson+tLocation+tOrganization+tO+fnPerson+fnLocation+fnOrganization+fnO);
101  average_fmeasure=2*(average_precision*average_recall)/(average_precision+average_recall);
```