# Addis Ababa University

Addis Ababa Institute of Technology
School of Electrical and Computer Engineering



# Amharic Named Entity Recognition Using Neural Word Embedding as a Feature


**By: Dagimawi Demissie**


**A Thesis Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment of Masters of Science in Computer Engineering**


**October, 2017**

**Addis Ababa, Ethiopia**

# Amharic Named Entity Recognition Using Neural Word Embedding as a Feature

By: Dagimawi Demissie

| | |
|---|---|
| Thesis Advisor | Examiner |
| **Dr. Surafel Lemma** | **Dr. Nune Sreenivas** |
| Chairman of Department | Examiner |
| **Dr. Yalemzewud Negash** | **Mr. Fitsum Assamnew** |

Submitted in Partial Fulfillment of the Requirements
for Masters of Science in Computer Engineering
Addis Ababa Institute of Technology
School of Electrical and Computer Engineering
October, 2017

# Declaration

This thesis is a presentation of my original research work. Wherever contributions of others are involved, every effort is made to indicate this clearly, with proper citation of sources. I, the undersigned, declare that this thesis has not been presented for a degree in any other university.

---

Dagimawi Demissie

# Acknowledgments

First and foremost, praises and thanks to God, for his countless blessings throughout my life and this research.

I would like to express my deep and sincere gratitude to my research advisor, Dr. Surafel Lemma for his valuable time, guidance, constructive ideas, comments which enabled me to gain good research experience.

I would also like to thank Mikiyas Tadele who gave me text corpus for my research. This whole research would not be successful with out his support.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. They have been my source of strength next to God. Last but not least, I would like to thank Ato Menore Tekeba, Ato Bisrat Derebssa and all my colleagues.

## Abstract

In this paper, Amharic Named Entity Recognition problem is addressed by employing a semi-supervised learning approach based on neural networks. The proposed approach aims at automating manual feature design and avoiding dependency on other natural language processing tasks for classification features. In this work potential feature information represented as word vectors are generated using neural network from unlabeled Amharic text files. These generated features are used as features for Amharic Named entity classification.

SVM, J48, random tree, IBk(Instance based learning with parameter k), attribute selected and OneR(one rule) classifiers are tested with word vector features. Additionally BLSTM(bi-directional long short term memory), LSTM(long short term memory) and MLP(multi layer perceptron) deep neural networks are also tested to investigate the impact of proposed approach.

From the experiments the highest F-score achieved was 95.5% using the SVM classifier. Relative to state-of-the-art approaches (SVM and J48) an average F-score improvement of 3.95% was achieved. The results showed that, automatically learned word features can substitute manually designed features for Amharic named entity recognition. Also these features has given better performance while reducing the effort in manual feature design.



**Keywords:** *Amharic Named Entity Recognition, Neural Word Embedding, Deep Neural Networks, Word2vec, Skip-gram Model, Natural Language Processing.*

# Contents

# List of Figures

# List of Tables

# Abbreviations

| | |
|---|---|
| **ANER** | Amharic Named Entity Recognition |
| **BLSTM** | Bidirectional Long Short Term Memory |
| **CBOW** | Continous Bag Of Words |
| **CNN** | Convolutional Neural Network |
| **CoNL** | Conference on Natural Language Learning |
| **CR** | Coreference Resolution |
| **CRF** | Conditional Random Field |
| **FN** | False Negative |
| **FP** | False Positive |
| **HMM** | Hidden Markov Model |
| **IE** | Information Extraction |
| **LSTM** | Long Short Term Memory |
| **MEMMS** | Maximum Entropy Markov Models |
| **ML** | Machine Learning |
| **MLP** | Multi Layer Perceptron |
| **MUC** | Message Understanding Conference |
| **NER** | Natural Language Processing |
| **NER** | Named Entity Recognition |
| **NN** | Neural Network |
| **POS** | Part Of Speech |
| **ReLU** | Rectified Linear Units |
| **RNN** | Recurrent Neural Network |
| **SGD** | stochastic Gradient Decent |
| **SMOTE** | Synthetic Minority Over-sampling Technique |
| **SSL** | Semi Supervised Learning |
| **STP** | Scenario Template Production |
| **SVM** | Support Vector Machine |
| **TDNN** | Time Delay Neural Network |
| **TEC** | Template Element Construction |
| **TP** | True Positive |
| **TRC** | Template Relation Construction |
| **WEKA** | Wakiato Enviroment for Knowledge Analysis |
| **WIC** | Walta Information Center |

# Chapter 1

# Introduction

This chapter discusses about basic definition of Named entity recognition, the objective, problem statement, methodology and contributions of this research.

## 1.1 Background

Named entity recognition (NER) is defined as the process of identification and classification of all proper nouns in a given text document in to predefined categories such as person, organization, location, date, address and time expressions (Kaur and Verma, 2014). The term named entity is first used in the sixth conference of message understanding (Marrero et al., 2013). It is intended to refer to unique identifier of entities.

The process of named entity recognition can be viewed as a two step process, which are detection (identification) and classification processes. The detection stage is concerned in marking a word or a phrase as named entity. Classification stage on the other hand is concerned with classifying detected named entities in to their respective classes (Gupta and Arora, 2009). NER is one of the major tasks in Natural Language Processing (NLP) (Sharnagat, 2014). It serves as a crucial component of many NLP applications such as Semantic Annotation, Question Answering, and Opinion Mining (Marrero et al., 2013). It also plays a vital role in natural language processing applications such as machine translation, information retrieval, text summarization and text clustering.

Named entity recognition is a challenging task especially for languages having low resource and complex linguistic structure like Amharic. Small amount of resources like training data for NER highly impacts the accuracy of the system. Complex structure of languages also needs design of important features and the best combination from these features. Amharic as one of low resourced and morphologically rich languages shares the above challenges.

Approaches to development of named entity recognition can be broadly classified in to three. These are rule based (linguistic approach), machine learning (statistical approach) and hybrid approach (Tadele, 2014). Rule based approaches are based on handcrafted rules by language experts. Machine learning approaches use a collection of training examples to build a model from training data and hybrid approach uses combination of the two. Machine learning approaches are further divided in to supervised, unsupervised and semi-supervised. Supervised approach uses labeled training data for building NER models. Unsupervised approaches do not need any structural information about the data (Alemu, 2013). Semi-supervised approaches use both labeled and unlabeled data for training.

Amharic named entity recognition got less attention and only few researches are done. Considering critical role of named entity recognition in many NLP tasks, the development of ANER will greatly enhance the performance of Amharic NLP tasks.

## 1.2 Problem statement

Due to its critical role in many NLP applications Named entity recognition is one of major research areas in NLP and has been an active research area for past twenty years (Sharnagat, 2014).

Many researches have been done on named entity recognition systems. Unfortunately most of these researches are specific to high resourced languages like English, French and other European languages. Despite its large number of speakers, only few research has been done.

One of the main challenges in NER systems is an ambiguity of Named entities in text. These ambiguities reside at two different levels: detection and classification. The first involves disambiguating Named Entities from non- Named Entities in text. The second involves classifying them into Named entity classes, e.g. person or location. Detection and classification levels of Named Entity recognition are sequential and the correctness of the identification phase is a prerequisite for the classification phase.

Amharic Named entity recognition (ANER) has many challenges due to structure of the language. Among these challenges ambiguity, spelling variations and nested named entities are common ones. Researches in ANER (Ahmed, 2010) (Alemu, 2013) (Tadele, 2014) tried to deal with these challenges in different ways. Their approaches can be summarized as selection of features, building feature extractor, preparing training data and testing it on a classifier algorithms with different combinations of features to get best performance. These approaches, however, have several limitations

The first limitation is selection of features which are critical for high accuracy of a classifier. In all proposed ANER approaches feature selection is manual and entirely

depends on Amharic language knowledge of the researcher. After selection of features, one has to go through a trial and error process to come up with the best feature set that gives higher performance. Another related problem is the need for complex feature extractors. To extract a feature for single word many operations have to be performed by feature extractor. Complexity of feature extractor has an effect in response time of the ANER system and makes it difficult to use in practical NLP applications.

Finally previous ANER approaches are based on the output of other NLP tasks like part of speech tagger and morphological analyzer. Being dependent on other NLP creates performance bottleneck in case of low performing POS tagger or morphological analyzer.

This thesis aimed at simplifying the process of manual feature design by using automatically generated features for classification. Therefore this study will answer the following research questions:

- **RQ1 [Word embedding for ANER]**

  Is it possible to use automatically generated Amharic text word features for named entity recognition task ?

- **RQ2 [Deep neural networks for ANER]**

  Can we improve performance of Amharic named entity recognition using deep neural networks ?

## 1.3 Objectives

### 1.3.1 General objective

The general objective of this research is to investigate the impact of automatically generated word vector features in Amharic named entity recognition task.

### 1.3.2 Specific objectives

- Generating neural word embedding that can be used as features for named entity classifier input.
- Testing state of the art algorithms in ANER (SVM, J48 decision tree) with automatically generated features.
- Testing different group of classifiers with automatically generated features.
- Building deep neural network classifiers that use word vector features as input.
- Evaluating performance of new developed models.

## 1.4 Methodology

- **Literature Review :** Literature survey of theoretical bases and existing Amharic Named Entity Recognition Systems.
- **Data Collection & Preparation :** Labeled and unlabeled corpora needed for the experiments are collected and prepared.
- **Propose Automatic Feature Extraction :** By investigating existing systems an automatic feature extraction using neural networks is proposed.

- **ANER Model Development :** Using automatically generated features different classifiers are used to build ANER models.
- **ANER Model Evaluation :** Models developed using automatically learned features are evaluated using precision, recall and F1-score metrics.

## 1.5 Scope

The aim of this research is to develop Amharic named entity recognition system. The scope of our study is limited to detection and classification of named entities of person, organization and location (ENAMEX) only.

## 1.6 Contributions

The main contributions of this research can be summarized as :

- We have developed ANER system that uses automatically generated features for Amharic named entity recognition. Our system is independent of other NLP tasks like POS tagger for features.
- We have substituted a complex feature extractor needed in case of manual design of features by very simple feature extractor which performs only lookup operation.
- We have prepared large unlabeled Amharic corpus that can be used for future researches.

## 1.7 Thesis organization

The rest of this document is organized as follows. Theoretical backgrounds in NER are explained in Chapter Two. State-of-the-art and challenges in ANER are discussed in Chapter Three. Our proposed approach is explained in Chapter Four. The experimental procedures followed and results are discussed in Chapter Five. Finally Chapter Six presents conclusions from experimental observations and future works to show further areas of improvement on ANER systems.

# Chapter 2

# Theoretical background

This chapter explains basic theoretical foundations used for solving the problem of Amharic Named entity recognition. It contains explanations of approaches for named entity recognition and brief description of neural word embedding features, which are bases for this research.

## 2.1 Named entity recognition

Named entity recognition is defined as the process of identification and classification of all proper nouns in a given text document in to predefined categories such as person, organization, location, date, address and time expressions (Kaur and Verma, 2014). The term named entity is first used in the sixth conference of message understanding. It is intended to refer to unique identifier of entities (Marrero et al., 2013).

Researchers in named entity recognition have given different definitions for named entities. These definitions can be categorized in to four, which are grammatical, rigid designation, unique identification and domain of application (Marrero et al., 2013). Grammatical category defines named entities as proper nouns or common names. Rigid designator category defines named entities as universally defined or accepted names. Unique identifier category on the other hand defines named entities as unique identifier of something. The last category defines that, purpose and domain of application defines

named entities (Marrero et al., 2013). From these definitions we have used unique identifier category to refer to named entities. Therefore, throughout this thesis named entities refers to unique identifiers of something.

The sixth message understanding conference defined named entity labels in to three categories (Sharnagat, 2014). The categories were defined as follows:

- **Named entities (ENAMEX):** person, organization, location
- **Temporal Expressions (TIMEX):** date, time
- **Number Expressions (NUMEX):** money, percentage, quantity

An example of MUC named entity categories as described on (Sharnagat, 2014) is given below:

The <ENAMEX TYPE=”LOCATION”>U.K</ENAMEX> satellite television broadcaster said its subscribers base grew <NUMEX TYPE=”PERCENT”>17.5 percent</NUMEX> during <TIMEX TYPE=”DATE”> the past year</TIMEX> to 5.35 million.

## 2.2 Applications of Named entity recognition

Named Entity Recognition (NER) is one of the major task in Natural Language Processing (NLP)(Sharnagat, 2014). It serves as a crucial component of many natural language applications such as Semantic Annotation, Question Answering, and Opinion Mining (Marrero et al., 2013). It also plays a vital roles in natural language processing applications such as machine translation, information retrieval, text summarization and text clustering.

### 2.2.1 Semantic annotation

Semantic annotation is the process of attaching additional information to various concepts (e.g. people, things, places, organizations etc.) in a given text or any other content. Unlike classic text annotations for reader's reference, semantic annotations are used by machines as a reference. Semantic annotation brings two major benefits for these systems which are information retrieval and improved interoperability (Marrero et al., 2013).

### 2.2.2 Question and answering

Question and answering systems provide answers to given user queries. Most question and answering systems work by gradually reducing amount of data they need to consider in several phases. After getting the user question it selects relevant documents and starts to filter out irrelevant documents (Tadele, 2014). In the process of filtering irrelevant documents named entity recognizer plays an important role by identifying text fragments that do not contain an entity compatible with expected answer (Marrero et al., 2013).

### 2.2.3 Opinion mining

Most people look online for opinions related to their interest. Opinion mining systems use named entity recognition as one of the preprocessing technique to provide the users more related opinions. After named entities are identified in the process related opinions can be fetched easily (Marrero et al., 2013).

### 2.2.4 Machine translation

Machine translation is the use of computers to automate the process of converting a content in one language to other language (Keselj, 2009). In the process of translation,

named entity recognition helps to disambiguate some words with similar surface form as of named entities (Tadele, 2014).

### 2.2.5 Information retrieval

Information retrieval is the task of retrieving relevant documents for user query. According to (Oudah and Shaalan, 2012) information retrieval benefits from named entity recognition in two ways. The first one is in the process of recognizing named entities in user's query. The second benefit is recognizing named entities within the documents to extract only relevant ones based on identified named entities.

### 2.2.6 Text summarization

Text summarization processes a single or multiple documents content into a summary which fully describes the content in the documents (Tadele, 2014). Named entities are information dense tokens and can define the domain of the text. Therefore named entity recognition can enhance identification of important text segments that can be used by automatic text summarizer (Hassel, 2003).

### 2.2.7 Text clustering

Text clustering is grouping of textual documents in a way that, documents having similar content will be clustered as same group. Named entity recognition can be used in text clustering for ranking resulted clusters based on ratio of entities associated with each cluster. It enhances the process of analyzing the nature of clusters and also to improve clustering approach using selected features (Oudah and Shaalan, 2012).

## 2.3 Overview of Amharic language

Amharic is one of the Semitic languages spoken in Ethiopia. It is the second most spoken Semitic language next to Arabic. The number of speakers are estimated 27 million, which makes it the second most spoken language in Ethiopia and one of the five most spoken languages in Africa(Tadele, 2014).

Amharic writing system uses a script which is originated from ancient language called Gee'z (Tadele, 2014). In modern Amharic script each syllable patterns comes in seven different forms and there are 33 basic Amharic characters called "Fidel".

### 2.3.1 Grammatical arrangement

As stated on (Tadele, 2014) Amharic words can be classified in to eight categories (parts of speech). These are noun, pronoun, preposition, adjective, adverb, conjunction, interjection and exclamation. Since the main focus of this research is on Amharic proper nouns, the noun class is briefly discussed.

#### Amharic nouns

Amharic nouns are words used to name or identify any of a class of things, people, places or ideas (Ahmed, 2010). Amharic nouns can be classified in to tangible (countable nouns), intangible (non-countable), collective nouns, common nouns and proper nouns (Tadele, 2014).

Tangible nouns are nouns which represent something that can be seen or touched. For example ሰው ፣ ወንበር ፣ ቤት (human, chair, house) denote things that can be counted. On the other hand intangible nouns represent something that cannot be seen or touched, for example nouns like ጨለማ ፣ ልጅነት (dark, childhood) represent things that cannot be

counted (Tadele, 2014). Collective nouns are used to denote class of things or concepts instead of particular entities. For example ወፍ ፣ አሳ ፣ መንጋ (bird, fish, herd) are names for class of things. Another class of nouns is common nouns which denote particular group of things and concepts in real world. For example በግ ፣ ፍየል (Sheep, Goat) refer particular group.

The last class which is main focus of this research is proper nouns. Proper nouns are nouns that identify specific person, organization, location and others. For example አበበ ፣ አለሙ ፣ አስቴር (Abebe, Alemu, Aster) are names of a person, አዲስ አበባ ፣ ጎጃም ፣ ጎንደር (Addis Ababa, Gojjam, Gonder) are names of locations and አዲስ አበባ ዩንቨርሲቲ ፣ ጠቅላይ ፍርድ ቤት (Addis Ababa University, Supreme Court) are names of organizations.

## 2.4 Challenges of Amharic named entity recognition

One of the main challenges of named entity recognition is that some proper nouns can belong to open class category. For example new names of a person or organization can be added to the classes. Another challenge is ambiguity. In Amharic same name may refer to different entities, for example Ethiopia can be the name of a person or location (Tadele, 2014). Additionally challenges like lack of capitalization, spelling variations, nested named entities and ambiguity are exhibited in Amharic texts.

### 2.4.1 Lack of capitalization

In languages like English capitalization rules for proper nouns enhance the process of named entity detection. Unfortunately there is no capitalization rule for proper nouns of Amharic language. This makes the process of detecting named entities much harder (Tadele, 2014).

### 2.4.2 Spelling variations

In Amharic language words having same meaning can be spelled differently. This problem emerges from duplication of some characters. For example the word ፀሀይ can be also written as ጸሀይ and አመት can be written as አሞት. The language has its own rules on how to use this characters but nowadays people tend to use them wrongly. This will create confusion in classification stage of named entities (Alemu, 2013).

### 2.4.3 Nested named entities

Nested named entities are formed by combination of two or more names. This leads to classification error if contextual information is not taken in to account. This problem is common in Amharic language. For instance if we consider nested named entity አዲስ አበባ ዩኒቨርሲቲ, the first two words can represent person name if considered individually and the combination of the two can represent location name. If the third word included in the context it becomes clear that it denotes organization name.

### 2.4.4 Ambiguity

Some named entities in Amharic have more than one named entity class. For example the name ፀሀይ which means (sun) can be also used as a name of a person. This ambiguity needs contextual features to resolve the correct context of the word.

## 2.5 Named entity recognition approaches

Approaches to development of named entity recognition can be broadly classified in to three. These are rule based (linguistic approach), machine learning (statistical approach) and hybrid approach (Mansouri et al., 2008).

### 2.5.1 Rule based approaches

Rule based approaches rely on handcrafted language rules prepared by language experts (Kaur and Verma, 2014). Generally rule based approaches consists of a set of patterns using grammatical, syntactic and orthographic features. These kind of systems can perform better for restricted domains. They can detect complex entities that might be difficult for learning models. But the main disadvantages of this approach are its lack of portability, robustness, and high cost of maintenance in slight change of data (Mansouri et al., 2008).

### 2.5.2 Machine learning Approaches

Machine learning approaches use a collection of training examples to automatically induce rules. These rules are sequence labeling algorithms for categorizing named entities (Nadeau and Sekine, 2007). Machine learning approaches are defined as statistical models to make predictions about named entities in a given text (Kaur and Verma, 2014).

There are three main approaches to machine learning. These are supervised learning, unsupervised learning and semi-supervised learning.

#### Supervised learning

Supervised learning method uses a labeled training data to enable the classifier develop a model that can correctly classify named entities. The performance of such approach depends on the number of training examples. As stated on (Mansouri et al., 2008) these systems cannot achieve high performance in the presence of small training data.

Now a days supervised learning approaches are the most dominant approaches in named entity classification tasks. Popular supervised learning approaches include Hidden Markov, Maximum Entropy Models, Conditional random fields, Support vector machines, decision tree, etc. (Tadele, 2014) .

### Semi-supervised approaches

Semi supervised learning methods use both labeled and unlabeled corpus for training. The motivation behind semi supervised learning is to overcome scarcity of labeled training data by using large unlabeled corpus (Sharnagat, 2014). The main method in Semi Supervised Learning is called bootstrapping which includes small measure of control at the beginning of learning process. The model is trained on an initial set of labeled data then prediction are made on separate set of unlabeled data. The performance of the model is then improved using predictions of previously developed models (Kaur and Verma, 2014).

### Unsupervised approaches

Many languages do not have large annotated data to train supervised learning algorithms. To solve this problem unsupervised learning approach is proposed (Sharnagat, 2014). Unsupervised learning methods do not need any structural information about the data. Common unsupervised learning tasks include, clustering, where the goal is to separate the n instances into groups; novelty detection, which identifies the few instances that are very different from the majority; and, dimensionality reduction, which aims to represent each instance with a lower dimensional feature vector (Alemu, 2013).

## 2.6 Named entity recognition features

Supervised learning approaches for named entity recognition is extremely sensitive to selection of features (Tkachenko and Simanovsky, 2012). Features are descriptors or characteristic attributes of words designed for algorithmic consumption. Features can be represented as Boolean, numeric or nominal values. Features used for named entity recognition can be classified in to three categories which are Word-level features, List lookup features and Document and corpus features (Nadeau and Sekine, 2007).

### 2.6.1 Word level features

Word level features describe word case, punctuation, numerical value and special characters. Example word level features are given in Table 2.1.

Table 2.1: Word level features

| **Features** | **Example** |
|---|---|
| Case | Starts with a capital letter, Word is all uppercased, The word is mixed case |
| Punctuation | Ends with period, has internal period, Internal apostrophe, hyphen or ampersand |
| Digit | Digit pattern, Cardinal and Ordinal, Roman number, Word with digits |
| Character | Possessive mark, first person pronoun, Greek letters |
| Morphology | Prefix, suffix, singular version, stem, Common ending |
| Part of speech | Proper name, verb, noun, foreign word |
| Function | Alpha, non-alpha, n-gram, lowercase, uppercase version, pattern, summarized pattern, token length, phrase length |

### Digit patterns

Special attention must be given to some digit patterns as they can give useful information such as dates, percentages, intervals, identifiers, etc. For example a two digit number and four digit number can be date or a year (Nadeau and Sekine, 2007). Also digits followed by "s" may stand for decade and digits followed by units may stand for quantity (Sharnagat, 2014).

### Common word ending

Common word endings are suffixes attached to the end of a word and they can change grammatical functions of a word. Understanding common word endings can help to identify some entities. For example a human profession often ends with "ist" (Journalist, cyclist) or nationality and language often ends with "ish" and "an " (Spanish, Danish, Ethiopian) (Tkachenko and Simanovsky, 2012).

### Patterns and summarized patterns

Pattern features is to map words on to a small set patterns over character types. For instance, a pattern Feature might map all uppercase letters to "A", all lower case letters to "a", all digits to "0" and punctuation to "-" for example:

- A.A.i.T= A-A-a-A
- Model-123=Aaaaa-000

These pattern features are then condensed further in which consecutive similar pattern types which are repeated are removed. For example the patterns given above can be summarize as follows:

- A.A.i.T=A-a-A
- Model-123= A-a-0

### 2.6.2 List lookup features

Lists are privileged features in Named entity recognition. Entities included in a list signifies an "is a" relationship for entities in the list (Sharnagat, 2014). List look up features can be divided in to three significant categories (Nadeau and Sekine, 2007) . These categories are summarized in Table 2.2.

Table 2.2: List lookup features

| **Feature** | **Example** |
|---|---|
| General List | general dictionary, stop words (function words), capitalized nouns (e.g., January, Monday), common abbreviations |
| List of entities | organization, government, airline, educational, first name, last name, celebrity, astral body, continent, country, state, city |
| List of entity cues | typical words in organization, person title, name prefix, post-nominal letters, location typical word, cardinal point |

### 2.6.3 Document and corpus features

Document features depend on both the content and structure of the documents. Having a large corpus can be used as an excellent source of features (Nadeau and Sekine, 2007). Document and corpus features are summarized in Table 2.3.

Table 2.3: Document and corpus features

| Feature | Example |
|---|---|
| Multiple occurrences | other entities in the context, uppercased and lowercased occurrences, anaphora, co reference |
| Local Syntax | enumeration, apposition, position in sentence, in paragraph, and in document |
| Meta Information | uri, email header, XML section, bulleted numbered lists, tables, figures |
| Corpus frequency | word and phrase frequency, co-occurrences, multi word unit permanency |

## 2.7 Distributed representation of words

Word representations are mathematical objects associated with each word. The representations are often in vector form. The contents of this vector representation are the features of each word. The value of each dimension represents the value for a specific feature (Sharnagat, 2014).

Traditionally one of the most obvious way of representing words is one hot vector representation. Each word (w) in a vocabulary (V) has a unique index. Then the words are represented by a vector of size |V|, in which the index of the word is one and the rest is zero (Sharnagat, 2014).

- Home = [0,0,1,0,0,0,....,0,0,0,0]
- House= [1,0,0,0,0,0,....,0,0,0,0]

One hot representation is easy to understand and implement, but it only considers local context and has many flaws. One of the problem with such representation is it fails to show correlation between words due to its local nature. For example consider the following two sentences for named entity tagging problem:

- “Abebe slept on the bed”
- “Aster slept on the couch”

If the above sentences are represented as one hot vector algorithm trained with the first sentence “Abebe” as person name might fail to tag the word “Aster” in the second unseen sentence. The reason is that the representation of the two person names doesn't capture the relation between the two names.

On the other hand distributed representations use lower dimensional real valued vectors to represent words. Each dimension represents latent feature about that specific word. The representation may look like, as given below:

- Home = [0.543, 0.1133,-0.300, 0, 0, 0,....,-0.210,0.1223]
- House= [0.111,-0.2103, 0.390,...., 0.129,0.4478,0.612]

This kind of representation is hoped to capture semantic and syntactic relation between words. Similar words are expected to be mapped to near vectors in the vector space. For example in the above two sentences if they are represented by distributed representation even if the tagger didn't see the word “Aster” in training data, it will get very similar vector as the word “Abebe” which makes the decision easy.

### 2.7.1 Skip-gram model

This model accepts a word Wi and predicts the words around a given word (Wi), which are context words (Wi-2, Wi-1, Wi+1, Wi+2). Context words does not need to be immediate words. Some words can be skipped within a given window size to look forward and backward from target word. Skip gram model has one hidden layered

neural network. The input layer consists of one-hot encoded vector of the vocabulary (Rong, 2014). Skip gram model neural network is depicted in Figure 2.1.

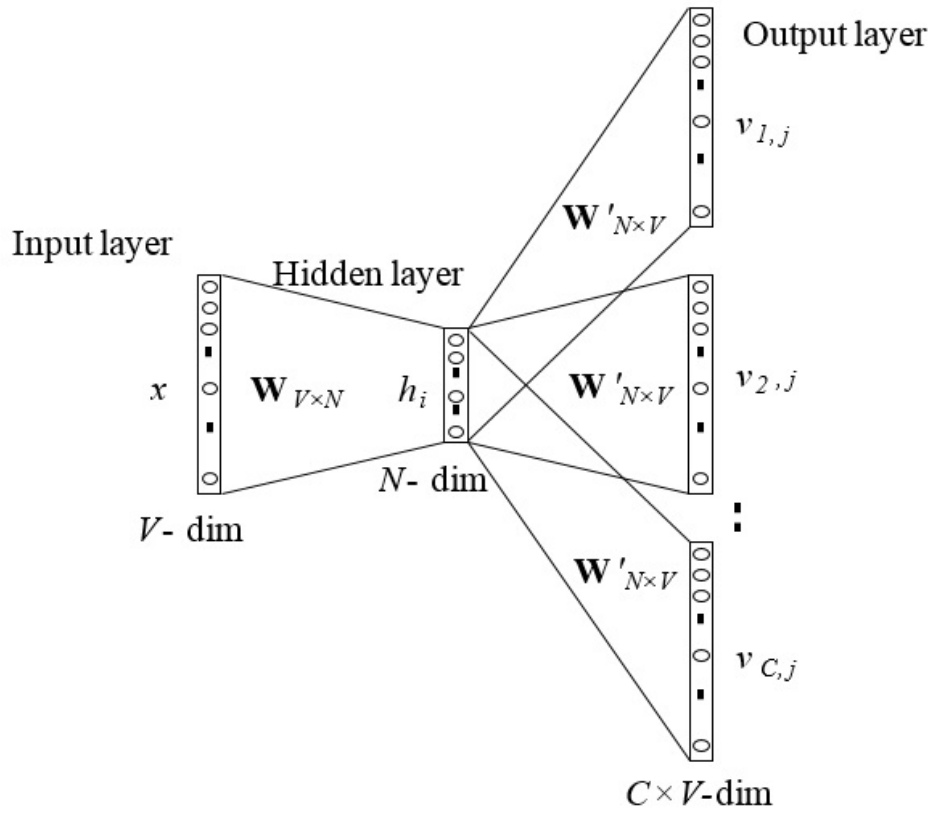


Figure 2.1: Skip gram model

### 2.7.2 Continuous bag of words model

Continuous bag of words is reverse of skip gram model. Given the context ( Wi-2, Wi-1, Wi+1, Wi+2 ) the task is to predict the word. Continuous bag of words model (CBOW) takes the average of the vectors of the input context words to compute the output of hidden layer, and use the product of the input layer hidden layer weight matrix and the average vector as the output(Rong, 2014). CBOW model neural network is depicted on Figure 2.2 .

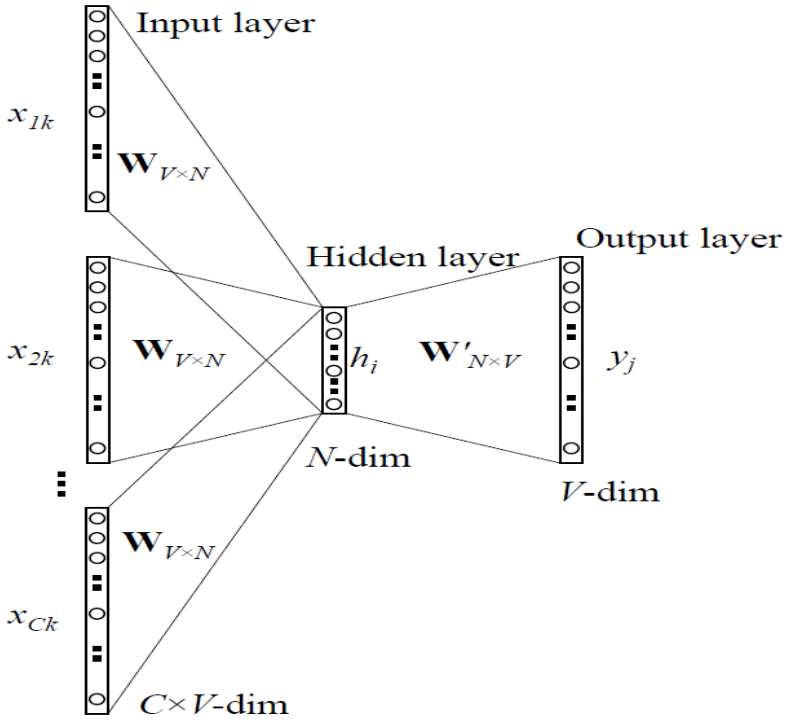


Figure 2.2: CBOW model

## 2.8 Semantic and syntactic information in neural word vectors

Models discussed in the previous sections can capture many linear dependencies between words. Syntactic properties of words like inflectional forms of a word are represented nearest to each other on the continuous vector space. Distributed vector representation of words shows state of the art accuracy on a test set for measuring syntactic and semantic word similarities (Mikolov et al., 2013) . More complex similarity checks can be performed by simple algebraic operations with vector representation of words. For example to find a word similar to “small” in same sense as “biggest ” is for “big”, it can be simply computed as :

X=vector(“biggest”)-vector(“big”)+vector(“small”)

Then searching the closest vector to the result vector X gives the answer which is "smaller". This result is by assuming that the model is well trained on very large data set. Additionally these vectors can be used to answer very subtle semantic relations between words such as a city and a country it belongs to, e.g. France is to Paris and German is to Berlin. Five types of semantic word relationships and nine types of syntactic relationship tests were conducted by (Mikolov et al., 2013). The results are shown in Table 2.4.

Table 2.4: Syntactic and semantic word relations test result

| **Type of relationship** | **Word pair 1** | | **Word pair 2** | |
|---|---|---|---|---|
| Common capital city | Athens | Greece | Oslo | Norway |
| All capital cities | Astana | Kazakhstan | Harare | Zimbabwe |
| Currency | Angola | kwanza | Iran | Rial |
| City-in-state | Chicago | Illinois | Stockton | California |
| Man-Woman | brother | sister | grandson | Granddaughter |
| Adjective to adverb | apparent | apparently | rapid | Rapidly |
| Opposite | possibly | impossibly | ethical | Unethical |
| Comparative | great | greater | tough | Tougher |
| Superlative | easy | easiest | lucky | Luckiest |
| Present Participle | think | thinking | read | Reading |
| Nationality adjective | Switzerland | Swiss | Cambodia | Cambodian |
| Past tense | walking | walked | swimming | Swam |
| Plural nouns | mouse | mice | dollar | Dollars |
| Plural verbs | work | works | speak | Speaks |

Based on these results these feature rich vectors could be used to improve many NLP tasks by substituting manually designed task specific features. Therefore the main intention of this research is to exploit the potential of word vectors for Amharic named entity recognition task.

## 2.9 Deep learning

Deep learning also known as deep structured learning or hierarchical learning is a set of machine learning algorithms that attempt to learn layered model of inputs commonly known as neural nets. It allows computational models composed of multiple processing layers to learn representations of data with multiple levels of abstraction. Deep learning algorithms are capable of discovering complex structures from large datasets using back propagation algorithm to update their internal parameters (Sharnagat, 2014).

Researchers in this area,like (Athavale et al., 2016) attempt to create models that learn representations from large unlabeled data . Various deep neural network architectures such as convolutional deep neural networks, deep belief networks and recurrent neural networks have been applied to many areas and shown impressive results. These areas include natural language processing computer vision speech recognition and bioinformatics (LeCun et al., 2015).

There are many types of deep neural networks and deep neural network architectures. This paper briefly discusses networks used in our research.

### 2.9.1 Recurrent neural networks

Traditional neural networks have a major limitation in considering sequential relation of inputs and outputs. It is assumed that each inputs and outputs are independent of each other. To overcome this limitation recurrent neural networks (RNN) are proposed. The idea behind recurrent neural networks is making use of sequential information. For example if we want to predict named entity tag of current word we have to know the tag of words occurred before it. Recurrent neural networks have memory about what has been calculated so far and uses it on current output computation. Theoretically RNN

make use of information in arbitrarily long sequence but in practice it is limited to few time steps (Britz, 2015). Typical recurrent neural network is shown in Figure 2.3:

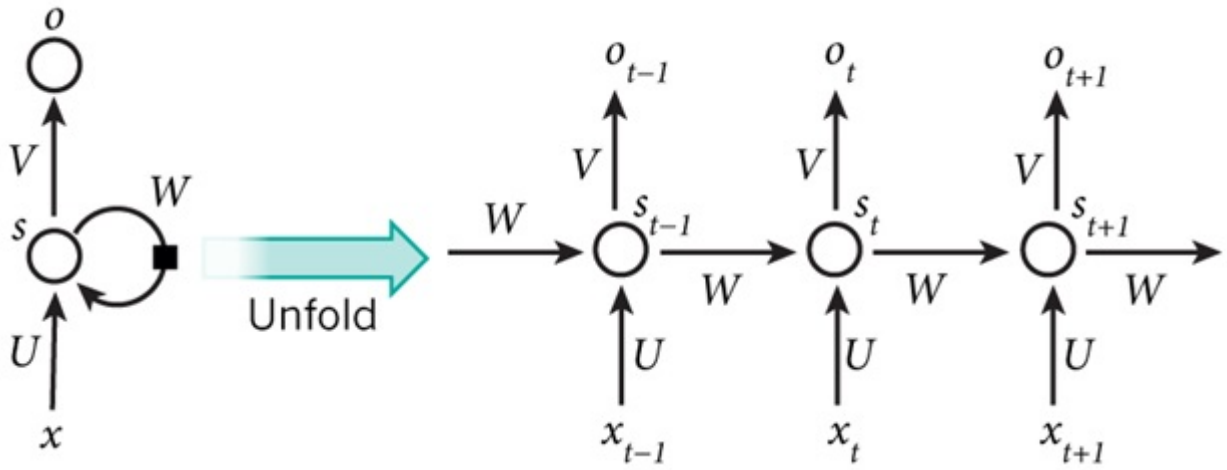


Figure 2.3: A recurrent neural network and the unfolding in time of its forward computation

Where $x_t$ is the input at time step t, $s_t$ is hidden state at time t, and it is the memory of the network which is calculated based on the previous hidden state and the input at the current step $s_t = f\left(ux_t + ws_{t-1}\right)$. The function f is usually non-linear function such as tanh or ReLu. Ot is the output at time t (Britz, 2015).

Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (U, V, W above) across all steps. This reflects that it is performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters the network need to learn (Britz, 2015).

RNN have shown great success in many NLP tasks. The most commonly used type of RNNs are long short term memory recurrent neural networks (LSTM RNN), which are much better at capturing long-term dependencies than typical recurrent neural network discussed before.

**Long short-term memory neural networks (LSTM)**

Sometimes it might be sufficient to remember recent information to perform recent task. But there are also cases where we need more context information. As the gap between relevant information and the point where it is needed becomes very large RNNs become unable to learn to connect the information. Long Short Term Memory networks usually just called “LSTM ” are a special kind of RNN, capable of learning long-term dependencies. They are explicitly designed for handling the problem of long term dependency where longer context information is needed for current task. All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer (Olah, 2015).

LSTM also shares this repeating structure, but contains four neural network layers interacting in a special way. The LSTM have the ability to remove or add information to the cell state, which is carefully regulated by structures called gates. Gates control the flow of information. They are composed of sigmoid neural network layer and a point wise multiplication operation. The sigmoid layer outputs numbers between 0 and 1 which describes how much of each component should be let through LSTM's has three such gates to protect and control the cell state. The first step in LSTM is to decide what information to throw away from the cell state. This decision is made by forget gate sigmoid layer. It looks at $h_{t-1}$ and $x_t$, and outputs a number between 0 and 1 for each number in the cell state $C_{t-1}$. A 1 represents “completely keep this” while a 0 represents “completely get rid of this ” (Britz, 2015). LSTM decision process diagram on what information to throw is shown in Figure 2.4.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad \textbf{(5)}$$

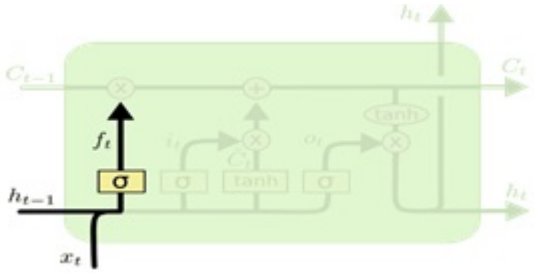


Figure 2.4: LSTM decision process on what information to throw

The next step is to decide what information we store in the cell state. First input gate sigmoid layer decide which values to update. Then a tanh layer creates a vector of new candidate values, $C_t$, that could be added to the state (Britz, 2015). LSTM decision process diagram on what information to store is shown in Figure 2.4.

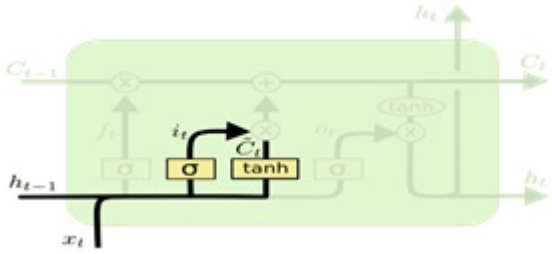


Figure 2.5: LSTM decision process on what information to store

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \quad \textbf{(6)}$$

$$C_t = tanh\left(W_c \cdot [h_{t-1}, x_t] + b_C\right) \quad \textbf{(7)}$$

Multiplying the old state by $f_t$, forgetting the things decided to forget earlier. Then adding $i_t * C_t$. This is the new candidate values, scaled by how much decided to update each state value. Finally the output will be based on the cell state. Sigmoid layer decides what part of cell state to output, then puts cell state through tanh and multiply it by output of sigmoid gate. Figure 2.6 shows LSTM cell state output.

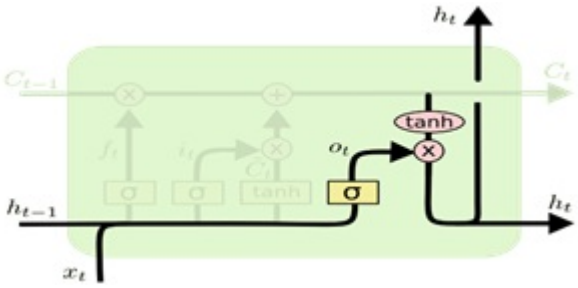


Figure 2.6: LSTM cell state output

$$O_t = \sigma\left(W_0\left[h_{t-1}, x_t\right] + b_0\right) \quad \textbf{(8)}$$

$$h_t = O_{t^*}\tanh\left(C_t\right) \quad \textbf{(9)}$$

**Bidirectional long short-term memory neural networks (BLSTM)**

Bidirectional long short memory recurrent neural networks are extended version of LSTM. BLSTM network is designed to capture information of sequential data set and maintain contextual features from past and future. This network processes input sequences in both directions with two sub layers in order to account for the full input context. The two sub layers compute forward and backward hidden sequences. Then these are combined to compute output sequence y (Mousa and Schuller, 2017). Architecture of BLSTM is shown in Figure 2.7.

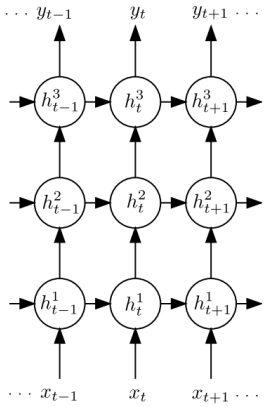


Figure 2.7: Architecture of a Bidirectional Long Short Term Memory RNN.

# Chapter 3

# Literature review

This chapter contains review of literatures on Ethiopian languages named entity recognition and foreign languages named entity recognition researches. From foreign languages reviewed, state-of-the-art approaches are briefly discussed.

## 3.1 Named entity recognition on local languages

From the literatures reviewed, we have found three researches on Amharic named entity recognition and two researches on Afan Oromo. The first research on Amharic was done by Moges Ahmed in 2010. The second research was done in 2014 by Besufekad Alemu and the third one done by Mikiyas Tadele in 2014. The first research on Afan Oromo was done by Mandefro Leggese in 2010 and the second one done by Abdi Sani in 2015.

Conditional random field machine learning approach is used for ANER in (Ahmed, 2010). Feature sets used by the researcher were word and tag context features, part of speech tag of tokens, prefix and suffix. The researcher conducted different combination of these features to determine best performing feature sets. For determining these features four different scenarios were conducted. In the first scenario all features were used. The second scenario used all feature sets except POS tag. In the third and fourth scenarios all features were considered except prefix and suffix respectively. From the experiments on these scenarios the third scenario achieved highest F-score of 74.61%. Based on

the result the researcher concluded that important combination of features for Amharic named entity recognition is the third scenario which considers all features except prefix feature. Based on the results he recommended use of large corpus and including rule based component (hybrid approach) to improve the performance of Amharic named entity recognition system.

(Alemu, 2013) also used conditional random field machine learning approaches. Previous and next word, named entity tag of a word, word pairs, word shape , prefix and suffix features were used for the experiments. The highest F-measure achieved was 80.66%. The features used were previous word, next word, prefix, suffix, previous word NE tag and next word NE tag. The researcher recommended using Stanford and ling pipe tools, and using large corpus and adding rule based component to improve performance of Amharic named entity recognition.

Hybrid approach is used for ANER in (Tadele, 2014). Decision tree (J48) and support vector machine (SVM) are used for classification. Also a rule based component having two rules based on presence of trigger words was included in the model. The features used are words, NE tag of words with window size of two, prefix, suffix, POS tag and nominal feature which indicates whether a word is noun or not. From the experiments the highest F-score achieved was 96.1% for J48 decision tree and 85.9% for SVM.

Hybrid approach which contains CRF machine learning algorithm and rule based component for Afan Oromo is used in (Kejela, 2010). A corpus of size 23,000 words have been used for training. The training data contains person, location, organization and miscellaneous. The miscellaneous category includes date and time, monetary value and percentage. Average performance of 77.41%, 75.80% and 76.60% for precision, recall and F1-score are achieved.

Also in (Genemo, 2015) Afan Oromo Named entity recognition is done using hybrid approach. The rule based component has parsing, filtering, grammar rules, white list gazetteers, blacklist gazetteers and exact matching components. The same corpus developed by the first researcher is used for training. The best performance achieved from this research was 84.12% precision, 81.21% recall and 82.52% F-Score. The summary of NER researches on Ethiopian languages is given in Table 3.1 .

Table 3.1: Summary of researches on Ethiopian languages NER

| **Research** | **Approach** | **Features used** | **F1 score (%)** |
|---|---|---|---|
| Named Entity Recognition for Amharic Language, 2010, Moges Ahemed | CRF | context features, POS tag, prefix, suffix | 74.61 |
| A Named Entity Recognition for Amharic, 2013, Besufekad Alemu | CRF | context features, word pairs, word shape, prefix, suffix | 80.66 |
| Amharic Named Entity Recognition Using a Hybrid approach, 2014, Mikiyas Tadele | SVM, J48 | context features, POS tag, prefix, suffix, nominal feature | 85.9, 96.1 |
| Named Entity Recognition for Afan Oromo , 2010, Mandefro Legesse | CRF, Rules | position features, Normalized features, words shape, POS tag, prefix, suffix | 76.60 |
| Afaan Oromo Named Entity Recognition Using Hybrid Approach, 2015, Abdi Sani | CRF, Rules | morphological, POS tag, word length, dot flag, capitalization, Gazetteers flags, nominal, NE tag | 82.52 |

We have used machine learning algorithms to develop ANER, similar to all the above researches. We have also used the same training data with the researches done on ANER. Our approach differs from all researches in the process of feature generation. All the researches described in Table 3.1 used hand crafted (manual) features. But in our

research we have used only automatic features generated by neural network other than NE tag feature. Additionally previous systems are dependent on other natural language processing tasks, like POS tagger for additional features. Whereas, our approach is not dependent on any NLP tasks for features.

## 3.2 Named entity recognition on foreign languages

Most of state of the art researches on NLP tasks use the power of deep neural networks and neural word embedding. Some of these researches which are closely related to our research are discussed below.

One of such researches is (Yao et al., 2015) which develops biomedical named entity recognition system based on deep neural networks. In this paper huge potential feature information represented as word vectors are generated by neutral networks based on unlabeled biomedical text files. Their system achieved F-score 71.01% on GENIA regular test corpus. From their experiments they have concluded that deep learning approach can effectively perform better on biomedical NER.

Similarly a research done by (Wang et al., 2015) proposes a way to use bidirectional long short term memory recurrent neural networks with word embedding for tagging problem. In this approach they have used automatically generated word features and a recurrent layer to process the sequence and predict the tag. They have achieved 89.64% F-score from this approach. Similarly (Lample et al., 2016) make use of CRF and BLSTM. The model rely on two sources of information about words: character-based word representations learned from the supervised corpus and unsupervised word representations learned from unannotated corpora. They have used English, Dutch, German, and Spanish corpus to evaluate their approach. The results achieved were 90.94%,

81.74%, 78.76%, 85.75% F-scores respectively on four languages. Another approach proposed in (Chiu and Nichols, 2015) uses neural network architecture that automatically detects Word and character-level features using a hybrid bidirectional LSTM and CNN architecture. They have used lookup tables to transform discrete features such as words and characters into continuous vector representations, which are then concatenated and fed into BLSTM network. To induce character-level features, a constitutional neural network is used. They have achieved 91.65% F-score on CoNLL data.

Most of the researches done on these area used same approach with some modifications to improve the accuracy of their model. Summary of researches done and their corresponding results are presented in Table 3.2.

We have used word vectors as a feature similar to these researches. But in our research we have also tried to show the results of using word vectors on different groups of classifiers. From these literatures we have seen that, impact of using word embeddings and deep neural network classifiers is not investigated for ANER system. Therefore this research tries to show the results from such approach on ANER.

Table 3.2: Summary of researches on foreign languages NER

| **Research** | **Approach** | **Features** | **F-score(%)** |
|---|---|---|---|
| Neural Architectures for Named Entity Recognition, Guillaume Lample, et al. 2016 | BLSTM, CRF | character based word representation, word vectors | 90.32 |
| Towards Deep Learning in Hindi NER: An approach to tackle the Labelled Data Scarcity, Vinayak Athavale, et al. ,2016 | BLSTM | word vectors, POS tag | 90.32 |
| Biomedical Named Entity Recognition based on Deep Neutral Network, Lin Yao et al. ,2015 | CNN | word vectors, POS tag | 71.01 |
| A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding, Peilu Wang et al. ,2015 | BLSTM | word vectors | 89.64 |
| Named Entity Recognition with Long Short-Term Memory, James Hammerton, et al. Named Entity Recognition with Long Short-Term Memory, James Hammerton, et al. | LSTM | self-organizing maps for sequences | 76.37 |
| Improving Named Entity Recognition for Morphologically Rich Languages using Word Embeddings, Hakan Demir, et al. | Averaged perception neural network | word vectors, context features | 91.85 |
| Named Entity Recognition with Bidirectional LSTM-CNNs, Jason P.C. Chiu, et al. | LSTM, CNN | word vectors | 91.62 |

# Chapter 4

# Proposed approach

After a review of researches done on Amharic named entity recognition one problem identified was design and selection of best features. Our proposed approach aims at automating this process of feature design by automatically learning features from given unlabeled data. After learning word representations from large unlabeled data, generated word feature vectors are used for training. In this section we will describe an architecture proposed for ANER that uses automatic features. This architecture can be used with different machine learning algorithms to use word vectors as a feature for ANER.

The architecture proposed contains four main processes. These are preprocessing, automatic feature generation, training with word embedding, and finally named entity prediction with trained prediction model. The processes are briefly described in next sections. The architecture diagram is shown in Figure 4.1.

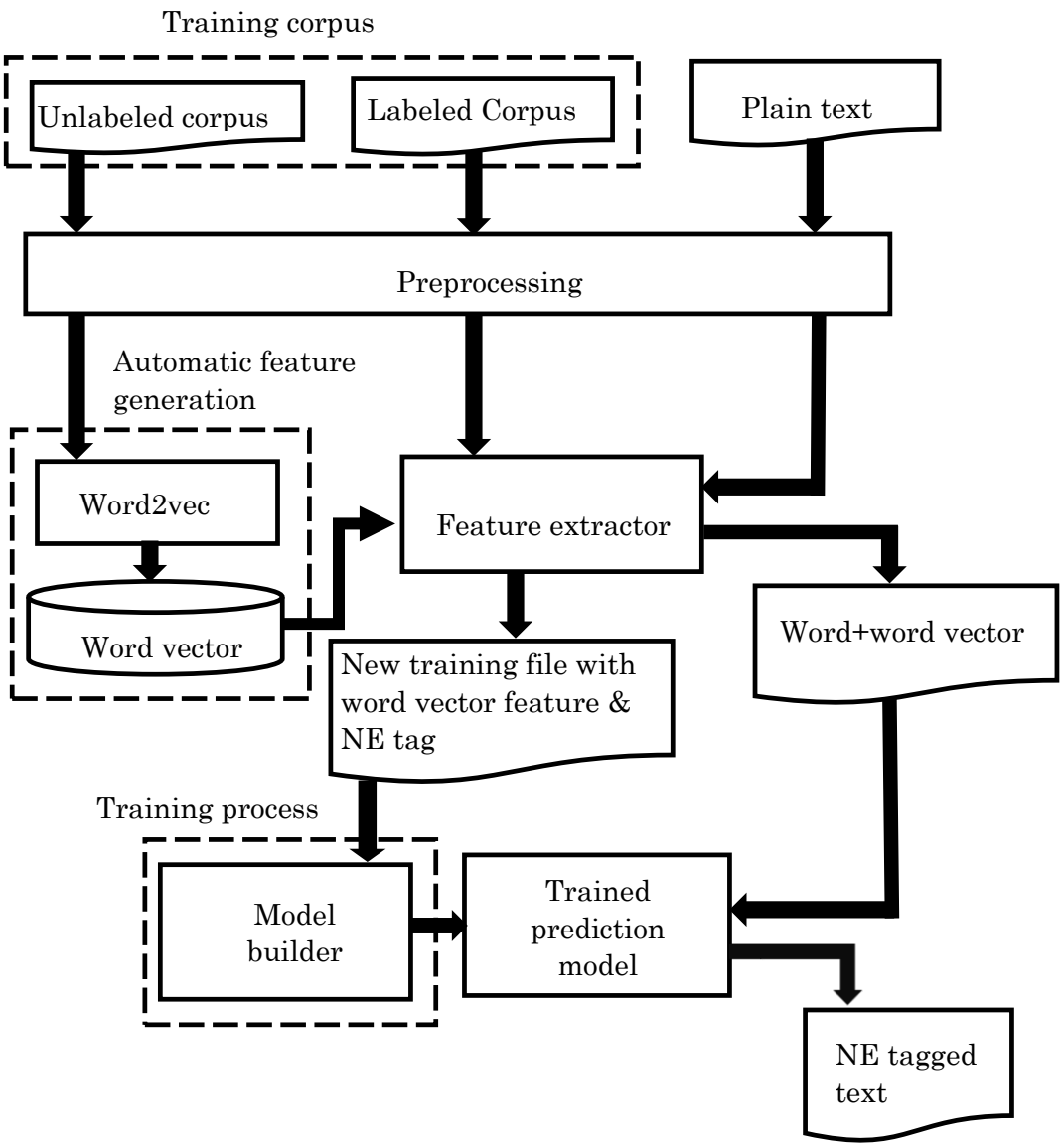


Figure 4.1: Architecture proposed for ANER

### Preprocessing

Preprocessing stage is the first process in our proposed approach. Two types of corpus are used for training which are unlabeled (for word vector generation) and labeled (for building NER model). Before proceeding to next stages these corpora have to pass through two main preprocessing stages. The first stage is cleaning process which

removes non-Amharic characters from the given text. This step is specially needed for unlabeled data because existence of non Amharic characters affects automatic feature generation. After this step is completed the next stage in preprocessing is transliteration. Transliteration is the process of converting Amharic characters to their English character equivalent. This step solves the problem of duplicated characters having same role in Amharic scripts. After transliteration, characters having same role but different notation will be mapped to same English characters. This process avoids confusion between two same words which are spelled differently.

### Automatic feature generation

The process of automatic feature generation uses preprocessed text in the first step as an input. The input is large unlabeled data, and this data will be tokenized before it's used for training. This stage is where the words semantic and syntactic relations are learned. The output from this stage is fixed sized vector representation for each word. Word2vec with skip-gram model is used for generating word vectors. After the training process is finished, all the words with their corresponding features will be logged to an output file. This output file will be the source of features in next stages of our architecture for ANER.

### Feature extractor

The main operation in our feature extractor is lookup operation. It takes tokenized words from preprocessing stage and retrieves corresponding feature vector from the output of Word2vector tool. For training purpose it reads tagged words and retrieves their word vector, then combines it with its named entity tag for training. For plain text it also reads tokenized words and retrieves their feature vector. Sample training data prepared from feature extractor output is given in Appendix B.

### Training process

Training process is the main part of our architecture. Training process uses different machine learning algorithms to build a model. The input for this process is a training file containing words with their feature vector and named entity tag. After the training data is fed, the model building process starts to form a model considering the features and their named entity tag. Features are represented in the form of floating numbers, therefore different machine learning algorithms could be used.

### Trained prediction model

The prediction model is the final trained model from the training. The input for prediction is the output of feature extractor which is a file containing the word vector of words in a given plain Amharic text. By taking this input it predicts the named entity tag of a word, therefore the output is target words with their predicted tag.

# Chapter 5

# Experiments

In this chapter experimental procedures, datasets, tools and the experimental scenarios used for evaluating our hypothesis are discussed.

## 5.1 Data collection

For this research, we used the corpus originally prepared by Ethiopian language Research Center of Addis Ababa University in a project called " the manual annotation of Amharic news documents ". This corpus contains manually annotated Amharic words in their context with appropriate POS (part of speech) tag by using WIC's (Walta Information Center) 1065 news articles which contains approximately 210,000 words. We have used 10,190 words tagged with their named entity tag which is taken from WIC by (Ahmed, 2010). The same training data was used by (Alemu, 2013) (Tadele, 2014) .

To generate word feature vectors, we built a corpus from WIC website archive. This corpus contains Amharic news text with 611,294 tokens. Its unlabeled data that will be used for feature generation purpose only.

## 5.2 Data preparation

The corpus from (Tadele, 2014) contains 22 manually designed features for every word, in this study the word and its named entity tag are used.

The corpus from Walta news contains English words and pictures. This corpus is also cleaned and prepared for training using Java tool. After generating automatic features Weka training data consisting of the word and its feature vector is prepared in WEKA .arff data format.

## 5.3 Development tools

Many development tools are used in the process of this research. The tools that has been used include Deeplearning4j library, WEKA, Tensorflow deep learning library, Keras deep learning library, Scikit learn machine learning library and JAVA with netbeans.

### 5.3.1 Deep learning 4j

Deep learning 4j is a Java based deep learning library. It is an open source product designed for adaptability in industry, where Java is very common. The framework currently interfaces with both Java and Scala (Fox et al., 2016). Deep learning 4j is used with intellij idea IDE for generation of neural word embedding (word feature vector) for this research.

### 5.3.2 WEKA

WEKA stands for Waikato Environment for Knowledge Analysis. It is a machine learning library which includes a collection of state-of-the-art algorithms and data processing tools. WEKA includes all the features that are needed for generic machine learning tasks. It has features for data preparation, building machine learning models, evaluating the models built using different evaluation metrics and visualizing the input data and the final result using different graphical tools (Singhal and Jena, 2013). We

have used WEKA for testing different types of classifiers, including baseline classifiers (SVM and J48).

### 5.3.3 Tensor flow

Tensor flow is Google's open source deep learning library released on November 2015. It includes C++ and Python API's. It has plenty of abstraction power but users might also be working with computational primitive wrappers such as matrix operations, element-wise math operators, and looping control. Tensor flow considers networks as a directed graph of nodes with data flow computation and dependencies encapsulated in it (Fox et al., 2016). Tensor flow is used as a back end for Keras library which is used for development of deep neural network classifiers.

### 5.3.4 Keras

Keras is a Python machine learning library for deep learning that can run using Theano or Tensor Flow as a back end. It's main focus is enabling implementation of deep learning models as fast and easy as possible for research and development. It runs on Python 2.7 or 3.5 and can execute on GPUs and CPUs given the underlying frameworks. It is released under the permissive MIT license (Fox et al., 2016). Keras is used for the development of deep neural network classifiers for proposed ANER system.

### 5.3.5 Scikit learn

Scikit-learn is a Python machine learning library which includes a wide range of state-of-the-art machine learning algorithms for supervised and unsupervised problems. It focuses on bringing machine learning to non-specialists by providing a general-purpose high-level language. It is easy to use, has high performance and contains detailed

documentations (Pedregosa et al., 2011). Scikit learn is used to evaluate deep neural classifiers by calculating performance metrics.

## 5.4 Evaluation metrics

A common approach for evaluating machine learning models is through comparison of predicted outputs of the model with that of labeled data by humans. Depending on the similarity between the two, a standard measure used in most researches called F-score can be calculated. F-score is combined measure of precision and recall. Precision is number of items correctly labeled as belonging to positive class divided by total number of elements labeled as belonging to positive class. Recall is defined as number of true positives divided by total number of elements that actually belong to positive class.

$$\text{Precision= TP/FP+TP} \quad \textbf{(1)}$$

$$\text{Recall= TP/FN+TP} \quad \textbf{(2)}$$

True positive (TP) measures NE tags which are produced by the model that match the true labels. False Positives (FP) measures NE tags returned by the model that are not in the true labels. And lastly, False Negatives (FN) are NE tags in the true labels but they are missed by the model (Alemu, 2013).

$$\text{F-Measure} = (\beta^2 + 1)\ \text{PR}/\ (\beta^2\ \text{R} + \text{P}) \quad \textbf{(3)}$$

$\beta$ is a measure of weighting between precision and recall. An evenly weighted F-measure can be calculated by setting $\beta$=1. Then the previous equation becomes as follows:

$$\text{F-Measure} = (2^*\ \text{Precision} * \text{Recall})\ /\ (\text{Precision} + \text{Recall}) \quad \textbf{(4)}$$

For all experiments in this research F-measure with $\beta$=1 is used as a performance evaluation metrics. The overall precision, recall and F-score of a classifier is calculated by using a weighted average. The weighted average is calculated by summing, the multiplication of score with number of instances for each class and dividing the result to the total number of instances.

## 5.5 Experimental setup

Two desktop machines are used for the experiments. The first machine is used for experimenting on Weka tool on windows environment and the other one for experiments done on deep neural networks using Linux environment. Hardware and software specifications of the two machines are given in Tables 5.1 and 5.2.

Table 5.1: Specifications of machine used for WEKA experiments

| | |
|---|---|
| Manufacturer | DELL |
| Model | OPTIPLEX 7010 |
| Processor | Intel® Core(TM) i7-3770 CPU @ 3.40 GHz |
| Memory(RAM) | 4.00 GB (3.89GB usable) |
| Operating System | Windows 10 |

Table 5.2: Specifications of machine used for deep neural network experiments

| | |
|---|---|
| Manufacturer | DELL |
| Model | OPTIPLEX-760 |
| Processor | Intel ® Core(TM) 2 Duo CPU E8600 @ 3.33 GHz |
| Memory | 4.00 GB (3.70 GB usable) |
| Operating System | Ubuntu 16.04 LTS |

Two types of corpora are used for training purpose. The first one is unlabeled news text for the purpose of word vector generation which has a total of 611,294 words. This corpus is prepared from WIC corpus containing 210,000 tokens and additional news text from WIC archive. The second corpus is labeled training data containing about 12,196

tokens taken from (Tadele, 2014). The labeled training data we got is highly imbalanced containing 9,899, 389, 1,267, 636 instances from others, person, organization and location respectively. In case of imbalanced data the classifier mostly classifies entities as a majority class members (Akbani et al., 2004). We used a WEKA preprocessing tool called SMOTE (synthetic minority oversampling technique) to avoid this problem. SMOTE is a means of increasing the sensitivity of a classifier to the minority class by balancing number of instances between the classes. The number of instances for each class after balancing is given in Table 5.3.

Table 5.3: Balanced data used for WEKA experiments

| **Class** | **Number of instances** |
|---|---|
| PERSON | 8,640 |
| ORGANIZATION | 8,552 |
| LOCATION | 8,367 |
| OTHERS | 8,314 |
| Total = 33,873 instances | |

## 5.6 Baseline experiment

To compare our proposed system, we selected the state of the art system by (Tadele, 2014). The reason for selecting this research experiments as baseline is because, it is the best performing ANER system. SMOTE class balancer tool is used to balance the training data used by baseline experiments. Corpus used and baseline experiment results are shown in Tables 5.4 and 5.5.

Table 5.4: Training data used by baseline experiment

| **Class** | **Number of instances** |
|---|---|
| PERSON | 6,224 |
| ORGANIZATION | 10,136 |
| LOCATION | 5,088 |
| OTHERS | 9,899 |
| Total = 31,347 instances | |

Table 5.5: Baseline experiment results

| **Classifier** | **Precision(%)** | **Recall(%)** | **F1(%)** |
|---|---|---|---|
| J48 | 96.1 | 96.1 | 96.1 |
| SVM | 86.3 | 86.1 | 85.9 |

## 5.7 Experimental scenarios

To evaluate our hypothesis different experiments are conducted. Generally these experiments can be grouped in to three. The first group of experiments are aimed at evaluating the quality of word vectors generated from Word2vec tool. The second group of experiments are to evaluate the performance of different classifiers using word embedding as a feature. The last group of experiments is conducted on deep neural networks using same training data. The first and second group of experiments are used to answer the first research question. The third group of experiments are used to answer the second research question.

To answer RQ1 [Word embedding for ANER], the initial step was generation of neural word embedding from unlabeled news text. For this purpose skip gram model of Word2vec is used. A vector with a fixed length of hundred is generated for each word in the corpus. The next step was an experiment to check the quality of word vectors before they are used as a classifier input. To evaluate the quality we used a simple similarity test between words. For this purpose top ten closest words by using cosine distance, are retrieved for given target word. The target words are representative instances from person and location.

The next step was actual usage of word vectors as a feature for named entity classification. Feature vector of each word in labeled training data is retrieved and training

data containing a word and its word vector is prepared using WEKA data format. As explained in the previous section class instance balancing using SMOTE is performed and final balanced data is used for the experiments.

By using automatically generated features an initial experiment on word vectors is conducted to evaluate the information contained in neural word embedding. Next set of experiments were on baseline classifiers (SVM, and J48). To check the validity of our approach on different classifiers, we conducted experiments on random tree, IBk, attribute selected and OneR classifiers in WEKA. These classifiers are selected from each group of classifiers available in WEKA. Therefore from lazy, meta, rule, and tree groups one classifier from each group is tested.

To answer RQ2 [Deep neural networks for ANER], we have used three types of deep neural networks which are BLSTM, LSTM and MLP (refer Appendix A for implementations). Balanced training data prepared for previous experiments is used for training. By changing the recurrent layer neuron cells (LSTM & BLSTM) our neural net is trained with 100 iterations for each case. For evaluation train test split using 80% of corpus as training data and 20% (1,780, 1,663, 1,722, 1,610) instances from person, organization, location and others is used as test data. Categorical cross entropy objective function with adam optimizer is used for training process. Finally a traditional multi-layer perceptron deep neural net is developed to check the consistency of our results. Deep MLP neural network with input layer having 120 neurons, a hidden layer containing four layers having 120 neurons with rectified linear unit activation (ReLu) and an output layer having 4 neurons with soft-max activation. Stochastic gradient decent (SGD) optimizer with categorical cross entropy objective function is used for training. To train this network we used 100 iterations and train test split evaluation with same ratio as the previous experiments.

## 5.8 Results

### 5.8.1 Word embedding for ANER

Table 5.6: Word similarity test results on word vectors

| **Word** | **Ten closest words** |
|---|---|
| Alemu | Kumsa, Sultan, Biftu, Habtamu, Wendwesen, Wendmu, Alene, Amare, Seyfu, Knfe |
| Gojam | Welo, Yemetekel, Awi, Besemenena, Besemen, Behadya, Waghmra, Msraqawi, Bedebubena, Shewa |

Table 5.6 shows that word vectors can capture word relations based on their role. We took two target words from person and location names to show the ability of word vectors in capturing role of words. Closest words to given person name are all names of a person and closest words to given location name are mostly location names. ***From this result we can see that automatically generated features are capable of capturing contextual word relations from a given text.***

The second experimental scenario is usage of word vectors as classifier input. Baseline experiment approach is tested with our new feature and four additional classifiers to check the consistency of our result.

Table 5.7: Results from SVM and J48 classifiers, ( scale 100% )

| Classifier | Class | Precision | Recall | F-Score |
|---|---|---|---|---|
| **SVM** | Person | 99.9 | 98.3 | 99.1 |
| | Organization | 91.4 | 93.4 | 92.4 |
| | Location | 94.1 | 97.2 | 95.6 |
| | Others | 96.6 | 92.9 | 94.7 |
| | **Weighted averages** | **95.5** | **95.5** | **95.5** |
| **J48** | Person | 95.6 | 99.4 | 97.4 |
| | Organization | 93.5 | 93.2 | 93.4 |
| | Location | 92.8 | 97.2 | 95.0 |
| | Others | 96.2 | 87.8 | 91.9 |
| | **Weighted averages** | **94.5** | **94.5** | **94.4** |

Weighted average F-scores from our classification result with word vectors as a feature, and baseline experiments with manually designed features are summarized in Figure 5.1 .

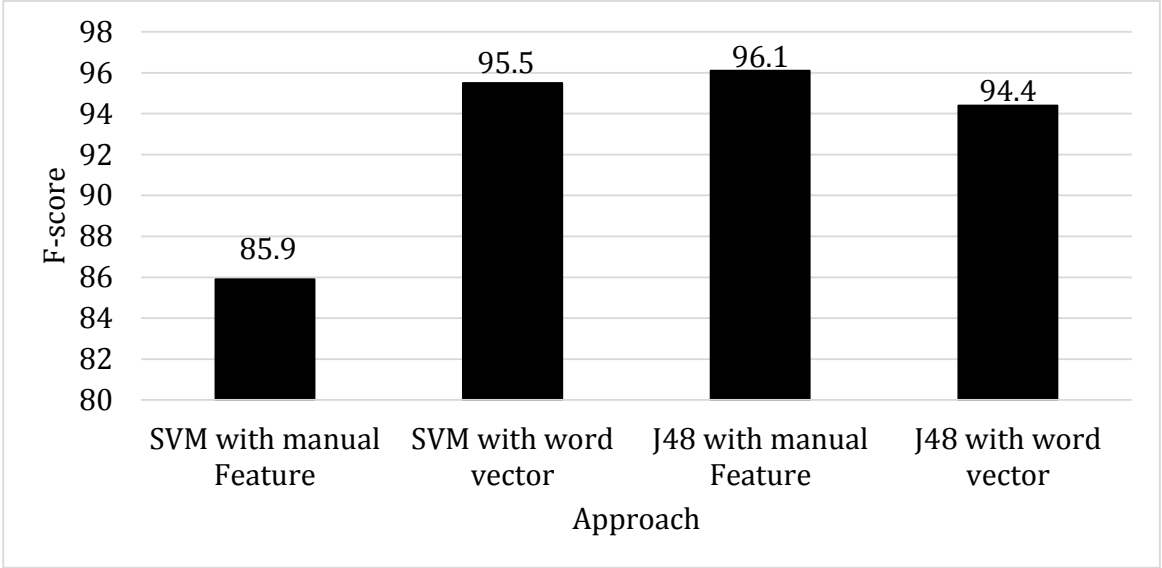


Figure 5.1: F-scores of state-of-the-art classifiers on manual and automatically generated features

The results we got on state-of-the-art classifiers shows that, neural word embedding can substitute manually designed features for named entity classification. Our approach

show 9.6% F-score improvement on SVM classifier. J48 classifier achieved lower F-score by 1.7%. One possible reason for low F-score could be, word vectors features are mapped as a continuous variable and J48 decision tree is better with discrete attributes (Quinlan, 1996). The features used by (Tadele, 2014) contain nominal features which are better for J48.

Results of word vectors while used with random tree, IBk, attribute selected and OneR classifiers are shown in Table 5.9.

Table 5.8: Results from different classifiers with automatically generated feature input (scale (100%))

| **Classifier** | **Class** | **Precision** | **Recall** | **F-Score** |
|---|---|---|---|---|
| **Random tree** | Person | 84.7 | 99.2 | 91.4 |
| | Organization | 93.6 | 94.1 | 93.8 |
| | Location | 94.9 | 96.9 | 95.9 |
| | Others | 96.4 | 76.7 | 85.4 |
| | **Weighted averages** | **92.3** | **91.8** | **91.6** |
| **IBk** | Person | 95.9 | 99.0 | 97.5 |
| | Organization | 92.1 | 94.1 | 93.1 |
| | Location | 92.6 | 97.3 | 94.9 |
| | Others | 96.7 | 86.3 | 91.2 |
| | **Weighted averages** | **94.3** | **94.2** | **94.2** |
| **Attribute selected** | Person | 75.1 | 99.8 | 85.7 |
| | Organization | 93.7 | 92.4 | 93.0 |
| | Location | 94.1 | 97.2 | 95.6 |
| | Others | 97.9 | 62.6 | 76.4 |
| | **Weighted averages** | **90.1** | **88.2** | **87.7** |
| **OneR** | Person | 75.0 | 99.8 | 85.6 |
| | Organization | 93.6 | 92.2 | 92.9 |
| | Location | 94.1 | 97.2 | 95.6 |
| | Others | 97.9 | 62.6 | 76.4 |
| | **Weighted averages** | **90.0** | **88.1** | **87.7** |

A summary of results for RQ1 is depicted in Figure 5.2.

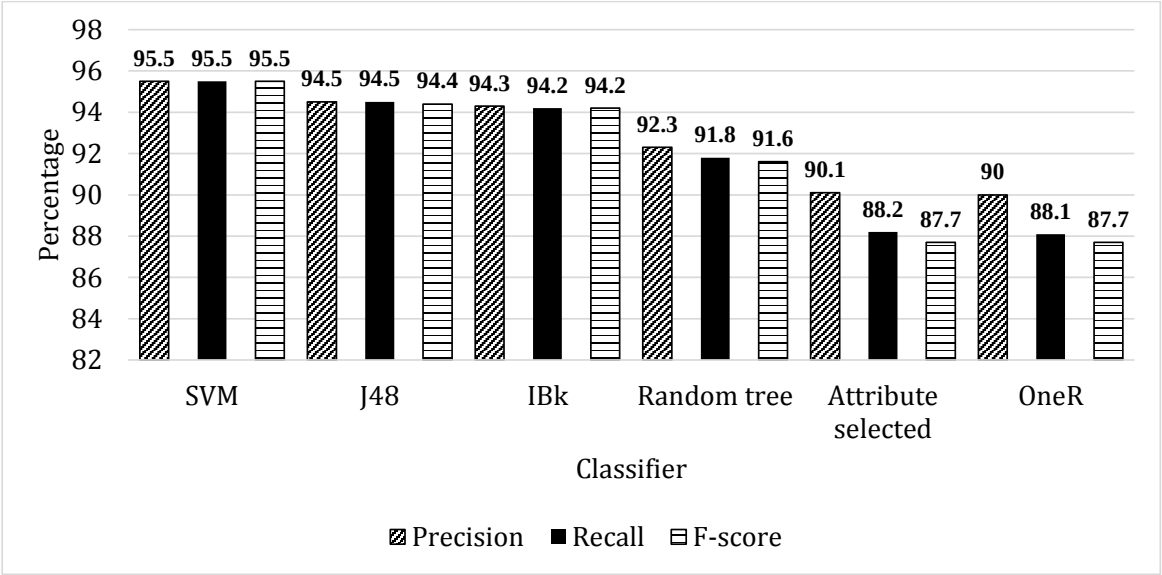


Figure 5.2: Summary of results for RQ1

***The results obtained from experiments conducted to answer RQ1, demonstrated that word embedding features can improve Amharic named entity recognition. Word vector features can substitute manually designed features while maintaining high performance for ANER. These results also show that, we can use the advantage of large unlabeled Amharic text to generate features and train named entity recognition system with available tagged data.***

### 5.8.2 Deep neural networks for ANER

Table 5.9: Results from deep neural networks with word embedding features (scale (100%))

| Classifier | Class | Precision | Recall | F-Score |
|---|---|---|---|---|
| **BLSTM** | Person | 98 | 94 | 96 |
| | Organization | 94 | 90 | 92 |
| | Location | 94 | 96 | 95 |
| | Others | 85 | 90 | 87 |
| | **Weighted averages** | **92.9** | **92.6** | **92.6** |
| **LSTM** | Person | 85 | 99 | 91 |
| | Organization | 93 | 91 | 92 |
| | Location | 95 | 95 | 95 |
| | Others | 92 | 76 | 83 |
| | **Weighted averages** | **91.2** | **90.6** | **90.4** |
| **MLP** | Person | 98 | 89 | 93 |
| | Organization | 93 | 89 | 91 |
| | Location | 92 | 96 | 94 |
| | Others | 80 | 88 | 84 |
| | **Weighted averages** | **91** | **90.5** | **90.6** |

A summary of results for RQ2 is depicted in Figure 5.2.

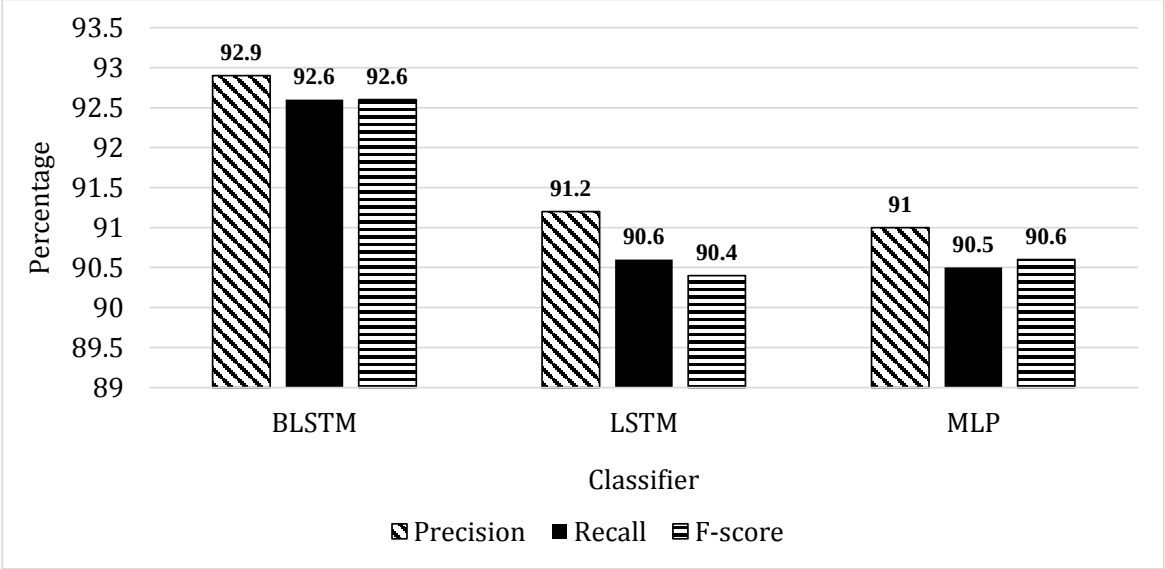


Figure 5.3: Summary of results for RQ2

***Three of the classifiers have shown high performance with average F-score of 92.6%, 90.4%, 90.6% for BLSTM, LSTM and MLP respectively. BLSTM outperforms MLP and LSTM by 2.2% and 2% respectively. Compared to results from SVM these scores are lower by 2.9%, 5.1%, 4.9% respectively. The lower F-measures could be due to three possible reasons, the first one is train test split used in these experiments discards 20% of training data for test. This affects the accuracy of the model. The other reason is that, deep neural networks need very large amount of data to give better performance and the data set we used is not large enough. Finally the network parameters used in our experiments might not be in their optimized value.***

## 5.9 Threats to external validity

### 5.9.1 Construct threat to validity

Construct threat to validity in our research may arise from the selection of feature size and classifiers. The length of word embedding selected is 100 and there might be another length of word vector that will result in a better performance. Also the classifiers we used in case of our WEKA experiments are selected from each group of classifiers to check the consistency of results. Hence there might be other classifiers that will result in a better performance using same approach. Additionally the deep neural network and WEKA classifiers parameters are mostly set to default configuration. Using the default configuration avoids over tuned parameters specific to some data. We have used different deep classifiers, but there might be a different deep learning classifiers with different set of parameters to achieve high performance.

### 5.9.2 Conclusion threat to validity

Conclusion threats to validity can be caused from small data and errors in measurement. The data set we used was not that much large. To decrease the effect of data size, we have used class balancer tool that increases number of training samples while maintaining balance between classes. Classifiers we used are also few in number. But we have tried to include additional classifiers to show the generalizablity of our results. The classifiers are selected from differnt group of classifiers which are lazy, meta, rule and tree classifiers. Finally, when we evaluate our models, we believe we didn't incur significant error on our results because performance metrics are calculated by automated process.

# Chapter 6

# Conclusion and future works

This chapter is conclusion of observations from our research. It also contains future works to show further researches that can be done in the future.

## 6.1 Conclusion

Named entity recognition systems are critical components of many NLP applications. Proper identification of named entities will improve the performance of other NLP tasks that rely on it. Different approaches have been used for solving NER problem which are rule based (linguistic approach), machine learning (statistical approach) and hybrid approach.

Different researches have been done on NER for local languages. All of these researches used machine learning approach and some of them used hybrid approach. The features used for classification are manually designed. Designing these features for a given language needs good understanding of language structure. In their training process combinations of features are tested to come up with best feature set with high performance.

In this research we have automated the process of manual feature designing. We trained Word2vec model to generate word vectors that can capture syntactic and semantic relations of words and used it as a feature for our experiments. We have proposed

an architecture that uses word embedding as a feature by avoiding usage of manually designed features.

To evaluate our hypothesis we have conducted different experiments. Each set of experiment is aimed at answering our research questions which investigates usage of word embedding for ANER and deep neural nets for ANER.

We have built Amharic named entity classification models from different classifiers. The first experiments were conducted on state-of-the-art classifiers which are, J48 and SVM. In comparison to the baseline experiments our system achieved better result by 9.6% on SVM and 1.7% lower F-score on J48. Additionally all other classifiers used also showed improved F-score than baseline SVM model. BLSTM, LSTM, and MLP deep neural networks are also used for experiments and achieved 92.6%, 90.4%, 90.6%.

Finally from our observations of experimental results we have concluded the following points.

- Neural word embedding (word vectors) can capture syntactic and semantic relations of Amharic words.
- Word embedding learned from large unlabeled data can substitute manually designed features and can be used as feature input for ANER systems.
- Different classifiers can use word vectors as a feature and can build ANER model with high performance.
- From the two types of LSTM cells, bidirectional LSTM outperforms unidirectional LSTM.
- Among all classifiers used in our experiments SVM achieved the highest F-score.

## 6.2 Future works

This research demonstrated that automatically generated neural word embedding can be used as a feature for NER task. It also shows that high performance NER system can be build from word embedding features. But to have full fledged ANER further researches could be conducted. Based on our observations we recommend the following future research areas.

- When conducting our experiments we have used small amount of corpus for word vector generation and to build the models. By using a very large data to train both neural embedding and the classifiers, we believe better results could be achieved. In the future, we aim to test our approach with large data..
- Our approach can be used for similar labeling tasks like part of speech taggers with out changing the approach. By only changing the training data Amharic part of speech tagger could be developed.
- We believe the deep neural network we used can be further refined by changing the network type and architecture.
- Our research focused on only basic named entity classes (Person,organization and location), further researches could consider additional categories of named entities to have complete ANER system.

# Bibliography


Ahmed, M. (2010). Named entity recognition for amharic language. Master's thesis, Addis Ababa University.

Akbani, R., Kwek, S., and Japkowicz, N. (2004). Applying support vector machines to imbalanced datasets. *Machine learning: ECML 2004*, pages 39–50.

Alemu, B. (2013). A named entity recognition for amharic. Master's thesis, AAU.

Athavale, V., Bharadwaj, S., Pamecha, M., Prabhu, A., and Shrivastava, M. (2016). Towards deep learning in hindi ner: An approach to tackle the labelled data scarcity. *arXiv preprint arXiv:1610.09756*.

Britz, D. (2015). Recurrent neural networks tutorial, part 1–introduction to rnns.

Chiu, J. P. and Nichols, E. (2015). Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Fox, J., Zou, Y., and Qiu, J. (2016). Software frameworks for deep learning at scale. *Internal Indiana University Technical Report*.

Genemo, A. S. (2015). Afaan oromo named entity recognition using hybrid approach. *Computer Science, Addis Ababa University, Addis Ababa*.

Gupta, P. K. and Arora, S. (2009). An approach for named entity recognition system for hindi: an experimental study. *Proceedings of ASCNT–2009, CDAC, Noida, India*, pages 103–108.

Hassel, M. (2003). Exploitation of named entities in automatic text summarization for swedish. In *NODALIDAâĂŹ03–14th Nordic Conferenceon Computational Linguistics, Reykjavik, Iceland, May 30–31 2003*, page 9.

Kaur, D. and Verma, A. (2014). Survey on name entity recognition used machine learning algorithm. *(IJCSIT) International Journal of Computer Science and Information Technologies*, 5(4):5875–5879.

Kejela, M. L. (2010). Named entity recognition for afan oromo. *Computer Science, Addis Ababa University, Addis Ababa*.

Keselj, V. (2009). Speech and language processing daniel jurafsky and james h. martin (stanford university and university of colorado at boulder) pearson prentice hall, 2009, xxxi+ 988 pp; hardbound, isbn 978-0-13-187321-6, 115.00.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., and Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Mansouri, A., Affendey, L. S., and Mamat, A. (2008). Named entity recognition approaches. *International Journal of Computer Science and Network Security*, 8(2):339–344.

Marrero, M., Urbano, J., Sánchez-Cuadrado, S., Morato, J., and Gómez-Berbís, J. M. (2013). Named entity recognition: fallacies, challenges and opportunities. *Computer Standards & Interfaces*, 35(5):482–489.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mousa, A. E.-D. and Schuller, B. (2017). Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis. In *Proceedings EACL*.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Olah, C. (2015). Understanding lstm networks. 2015. *URL http://colah. github. io/posts/2015-08-Understanding-LSTMs/img/LSTM3-chain. png*.

Oudah, M. and Shaalan, K. F. (2012). A pipeline arabic named entity recognition using a hybrid approach. In *Coling*, pages 2159–2176.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Quinlan, J. R. (1996). Improved use of continuous attributes in c4. 5. *Journal of artificial intelligence research*, 4:77–90.

Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Sharnagat, R. (2014). Named entity recognition: A literature survey. *Center For Indian Language Technology*.

Singhal, S. and Jena, M. (2013). A study on weka tool for data preprocessing, classification and clustering. *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, 2(6):250–253.

Tadele, M. (2014). Amharic named entity recognition using a hybrid approach. Master's thesis, AAU.

Tkachenko, M. and Simanovsky, A. (2012). Named entity recognition: Exploring features. In *KONVENS*, pages 118–127.

Wang, P., Qian, Y., Soong, F. K., He, L., and Zhao, H. (2015). A unified tagging solution: Bidirectional lstm recurrent neural network with word embedding. *arXiv preprint arXiv:1511.00215*.

Yao, L., Liu, H., Liu, Y., Li, X., and Anwar, M. W. (2015). Biomedical named entity recognition based on deep neutral network. *International Journal of Hybrid Information Technology*, 8(8):279–288.

# Appendix A

# Deep Classifiers

```
##### ANER with BLSTM RNN ######
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn import model_selection
from sklearn.metrics import classification_report
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation,LSTM,Bidirectional
from keras.optimizers import SGD
import numpy as np
from sklearn.model_selection import train_test_split
import pandas
from keras.utils.np_utils import to_categorical
from sklearn.metrics import confusion_matrix
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
# load dataset
dataframe = pandas.read_csv("Training.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:100].astype(float)
Y = dataset[:,100]
Y = to_categorical(Y)
#split training and test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.20, random_state=seed)
# reshape Xtrain and X test
X_train=np.reshape(X_train, (len(X_train), 1, 100))
X_test=np.reshape(X_test, (len(X_test), 1, 100))
model = Sequential()
model.add(Bidirectional(LSTM(120,return_sequences=True),
input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Bidirectional(LSTM(120,return_sequences=True)))
model.add(Bidirectional(LSTM(120)))
model.add(Dense(Y_train.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='rmsprop',
metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, batch_size=32, verbose=1)
score = model.evaluate(X_test,Y_test, batch_size=32)
Predict=model.predict(X_test)
Decoded=np.argmax(Predict,axis=-1)
Ytrue=Y_test.argmax(1).flatten()
Ytrue=np.asarray(Ytrue,dtype='int32')
report = classification_report(Ytrue, Decoded)
print '\nDetailed␣Report\n'
print (report)
print'\n␣Confusion␣Matrix\n'
conf=confusion_matrix(Ytrue,Decoded)
print conf ,'\n'
print 'Score␣',score[1]*100
```

```
##### ANER with unidirectional LSTM #####
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn import model_selection
from sklearn.metrics import classification_report
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation,LSTM
from keras.optimizers import SGD
import numpy as np
from sklearn.model_selection import train_test_split
import pandas
from keras.utils.np_utils import to_categorical
from sklearn.metrics import confusion_matrix
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
# load dataset
dataframe = pandas.read_csv("Training.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:100].astype(float)
Y = dataset[:,100]
Y = to_categorical(Y)
#split training and test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.20,
random_state=seed)
# reshape Xtrain and X test
X_train=np.reshape(X_train, (len(X_train), 1, 100))
X_test=np.reshape(X_test, (len(X_test), 1, 100))
########################################
model = Sequential()
model.add(LSTM(120,return_sequences=True,
input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(LSTM(100,return_sequences=True))
model.add(LSTM(100))
model.add(Dense(Y_train.shape[1], activation='softmax'))
model.compile(loss='categorical_crossentropy',
 optimizer='adam', metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=100, batch_size=32, verbose=1)
score = model.evaluate(X_test,Y_test, batch_size=32)
Predict=model.predict(X_test)
Decoded=np.argmax(Predict,axis=-1)
Ytrue=Y_test.argmax(1).flatten()
Ytrue=np.asarray(Ytrue,dtype='int32')
report = classification_report(Ytrue, Decoded)
print '\nDetailed Report\n'
print (report)
print'\n Confusion Matrix\n'
conf=confusion_matrix(Ytrue,Decoded)
print conf ,'\n'
print 'Score ',score[1]*100
```

```
###### ANER with MLP ############################
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn import model_selection
from sklearn.metrics import classification_report
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD
import numpy as np
from sklearn.model_selection import train_test_split
import pandas
from keras.utils.np_utils import to_categorical
from sklearn.metrics import confusion_matrix
# fix random seed for reproducibility
seed = 7
np.random.seed(seed)
# load dataset
dataframe = pandas.read_csv("Training.csv", header=None)
dataset = dataframe.values
X = dataset[:,0:100].astype(float)
Y = dataset[:,100]
Y = to_categorical(Y)
#split training and test data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
 test_size=0.20, random_state=seed)
model = Sequential()
model.add(Dense(120, activation='relu', input_dim=100))
model.add(Dense(120, activation='relu'))
model.add(Dense(120, activation='relu'))
model.add(Dense(120, activation='relu'))
model.add(Dense(120, activation='relu'))
model.add(Dense(5, activation='softmax'))
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9,
nesterov=True)
model.compile(loss='categorical_crossentropy',
optimizer=sgd,metrics=['accuracy'])
model.fit(X_train, Y_train, epochs=300,batch_size=32)
score = model.evaluate(X_test,Y_test, batch_size=32)
Predict=model.predict(X_test)
Decoded=np.argmax(Predict,axis=-1)
Ytrue=Y_test.argmax(1).flatten()
Ytrue=np.asarray(Ytrue,dtype='int32')
report = classification_report(Ytrue, Decoded)
print '\nDetailed␣Report\n'
print (report)
print'\n␣Confusion␣Matrix\n'
conf=confusion_matrix(Ytrue,Decoded)
print conf ,'\n'
print 'Score␣',score[1]*100
```

# Appendix B

# Sample training data

```
yegambEla,-0.153305,0.116933,0.214152,0.025864,-0.08117,ORGANIZATION
mrmr,0.042483,0.061404,0.038219,-0.038649,-0.074938,ORGANIZATION
maIkel,-0.186347,0.091409,-0.172896,0.106773,0.070097,ORGANIZATION
hulet,0.203535,0.217793,0.133808,-0.196595,0.373735,O
yetexaxalu,-0.039288,-0.016551,0.058609,0.09986,0.023633,O
yebeqolona,0.050875,0.028409,0.080038,0.065379,0.019249,O
yemaxla,-0.012882,0.016072,0.075467,0.063509,-0.030381,O
zerocn,-0.014679,0.011532,0.019815,0.050414,0.062673,O
bemrmr,-0.035704,0.017807,0.057679,0.020972,-0.107615,O
magNetun,-0.013611,0.077893,0.222668,0.205491,-0.036908,O
astaweqe,-0.128656,0.149551,0.421336,-0.177356,-0.020566,O
::,0,0,0,0,0,O
astebabari,-0.035247,0.042429,0.025423,0.151584,-0.193012,O
ato,-0.417835,0.125718,0.188247,-0.455521,-0.276809,O
belayneh,-0.045846,-0.000772,0.014171,0.033052,-0.103195,PERSON
ayalEw,-0.057266,0.005746,0.020755,0.043034,-0.179062,PERSON
lewalta,-0.326168,-0.064239,0.150459,0.067694,-0.383392,ORGANIZATION
InformExn,-0.32616,-0.186347,0.091409,-0.172896,0.106773,ORGANIZATION
IndegeleSut,0.10778,0.076591,-0.166279,0.270082,-0.100033,O
maIkelu,0.02583,-0.006896,-0.103767,-0.017523,-0.087931,O
zer,0.037752,0.002819,0.037215,-0.105298,-0.081748,O
balefew,-0.068184,-0.137138,-0.148893,0.056637,-0.253954,O
and,0.022233,-0.027359,-0.05673,-0.091969,0.23855,O
amet,-0.189924,-0.246649,0.12719,0.137355,0.080149,O
bakahEdew,-0.199476,0.02655,0.073176,0.021214,-0.320723,O
mrmr,0.042483,0.061404,0.038219,-0.038649,-0.074938,O
new,-0.061831,-0.016131,-0.30506,-0.22305,0.369784,O
::,0,0,0,0,0,O
```