Addis Ababa University

College of Natural Sciences

Afaan Oromo Named Entity Recognition Using Neural Word Embeddings

Mekonini Kasu Taye

A Thesis Submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

(in Data and Web Engineering)

Addis Ababa, Ethiopia

October 26, 2020

Addis Ababa University

College of Natural Sciences

Mekonini Kasu Taye

Advisor: Yaregal Assabie (PhD)

This is to certify that the thesis prepared by Mekonini Kasu, titled: *Afaan Oromo Named Entity Recognition Using Neural Word Embeddings and* submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Name	Signature	Date
Advisor: Yaregal Assabie (PhD)	_____	_____
Examiner: Dida Midekso (PhD)	_____	_____
Examiner: Ayelaw Belay(PhD)	_____	_____

Abstract

Named Entity Recognition (NER) is one of the canonical examples of sequence tagging that assigns a named entity label to each of a sequence of words. This task is important for a wide range of downstream applications in natural languages processing. Two attempts have been conducted for Afaan Oromo NER that automatically identifies and classifies the proper names in text into predefined semantic types like a person, location, and organizations and miscellaneous. However, their work heavily relied on hand design feature.

We proposed a deep neural network architecture for Afaan Oromo Named Entity Recognition, based on context encoder and decoder models using Bi-directional Long Short Term Memory and Conditional Random Fields respectively. In the proposed approach, initially, we generated neural word embeddings automatically using skip-gram with negative subsampling from an unsupervised corpus size of 50,284KB. The generated word embeddings represent words in semantic vectors which are further used as an input feature for encoder and decoder model. Likewise, character level representation is generated automatically using BiLSTM from the supervised corpus size of 768KB. Because of the use of character level representation, the proposed model is robust for the out-of-vocabulary words.

In this study, we manually prepared annotated dataset size of 768KB for Afaan Oromo Named Entity Recognition. We split this dataset into 80% for training, 5% for testing and 15% for validation. We prepared totally 12,963 named entities from these 10,370.4 %, 648.15% and 1,944.45% are used for training, validation and test set respectively. Experimental results show that the combination of BiLSTM-CRF algorithms with pre-trained word embedding and character level representation and regularization techniques (dropout) perform better as compared to the other models such as Bi-LSTM, BiLSTM-CRF with only character level representation or word embeddings. Using Bi-LSTM-CRF model with pre-trained word embeddings and character level representation significantly improved Afaan Oromo Named Entity Recognition with an average of 93.26 % F-Score and 98.87 % accuracy.

Keywords: Afaan Oromo NER, Context Encoder and Tag Decoder, Distributed Representation, Deep Neural Networks

Acknowledgments

First and foremost, thanks to God, for His countless blessings throughout my life.

In particular, I would like to express my sincere gratitude to my adviser Yaregal Assabie (PhD) for his valuable time, guidance, constructive ideas, comments which enabled me to gain good research experience. Without his supervision and support, it would be difficult for me to reach to this final stage. His excitement and broad experience always assisted me in my work. I am very grateful that I could grow professionally under his supervision.

Last but not least, I would like to thank all of my partners, friends for giving me assistance and encouragements.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Acronyms and Abbreviations

Bi-LSTM	Bidirectional Long Short-Term Memory
CBOW	Continuous Bag of Words
CRF	Conditional Random Field
HMM	Hidden Markov Model
LSTM	Long Short-Term Memory
ML	Machine Learning
MUC	Message Understanding Conference
NE	Named Entity
NLP	Natural Language Processing
NER	Named Entity Recognition
NERC	Named Entity Recognition and classification
NN	Neural Network
RNN	Recurrent Neural Network
SVM	Support Vector Machines

Chapter One: Introduction

1.1 Background

Daniel and James [1] define Natural Language Processing (NLP) as computational techniques that take written or spoken human language and produce natural language data. It focuses on developing a system that enables the computer to communicate with peoples using everyday language. There is a large amount of unstructured data that exists on the Internet. This abundant data is useful only if suitable techniques are available to process the data and extract knowledge from it. This process needs the natural language processing application which is named as information extraction [1]. It transforms unstructured textual data into structured data that can be understood by machines. The starting point for most information extraction applications is Named Entity Recognition (NER). Sundheim and Grishman [2] organized the first NER task. To this end, they identify the main goal of NER that can be considered names from a set of documents and classify them into a predefined semantic category.

As an approach for NER, many researchers used rule-based, machine learning, and deep neural network. However, these approaches have their own limitations. According to Ghiasvand [3], the rule-based method for NER lack portability and robustness. To address these problems, machine learning approach is proposed, which can be further classified as supervised, semi-supervised and unsupervised. In supervised machine learning, the machine is presented with huge amounts of manually annotated data. However, the manual annotation of the data can be a difficult, time-consuming and expensive process which makes it hard to apply supervised models in such situations. Supervised machine learning has been developed based on a sequence labeling algorithm such as Hidden Markov Models [4], Conditional Random Fields [5], Support Vector machines [6] and Decision Tree and other as well. In contrast to supervised machine learning, semi-supervised NER is used both label and unlabeled text which expected to decrease the labor-intensive and the cost of developing such a system. On the other hand, machine learning models for natural language processing become extremely difficult when the number of dimensions in the data set is high. Goodfellow et al., [7] present that machine learnings are insufficient to learn complicated function in high dimensional space which impose a high computational cost. Accordingly, machine learning in natural language processing is suffering from the problem which is the so-called “curse of

dimensionality.'' When the number of dimensions in the data is high, it hurts the generalization performance of the system. This problem can also result in model overfitting to the training data set. To overcome such a problem, deep neural network, which uses distributed word vector representation instead of sparse feature representation is proposed. Compared to classical machine learning model, deep neural network model that uses distributed representation to the task of NER has successively achieved a good performance [8, 9, 10, 11, 12, 13].

Most of the recent work has been reached with NER systems based on artificial neural networks. Kuksa et al. [14] stated that the recent neural NER architectures have been achieved better performance with minimal feature engineering and maximal performance NER. Different neural architectures have been proposed, mostly based on deep neural network over characters, sub-words, or word embeddings. For instance, the effectiveness of deep neural network in NER widely demonstrated for technologically advanced languages like English [14, 15], German [16, 17], Czech [11], Spanish, Italian [18], Chinese [19], and Japanese [13] and other as well.

Despite a lot of work has been conducted on NER for non-Ethiopian languages, only limited works are found in Ethiopian languages in Afaan Oromo [20, 21] and Amharic language [22, 23, 24]. Most of the researchers in these studies, utilized linear statistical models which rely on handcrafted feature engineering and task specific resource. Such approaches, however, have failed to address data sparsity, generalization and overfitting of model for new example.

According to Guo et al. [25], data sparsity in natural language is mainly caused by two factors: namely the lack of labeled training data and the Zipf distribution of words. These factors happen because of, the high-dimensional and sparse lexical feature representation, which completely ignores the semantic similarity between features, particularly word features. To address this problem, an effective way is to use deep neural network that uses a generalized representation of words by taking advantage of the numerous unlabeled training data. Deep neural network based NER used both label and unlabeled text which decreases the labor-intensive and the cost.

1.2 Motivation

According to Sundheim and Grishman [2], NER is a task of automatically identifying entities of certain types from a text document: - for instance, identifying all person and an organization name from a new text [26]. It mostly utilized before complex natural language processing tasks such as relation extraction, knowledge base generation, and question answering [27]. Furthermore, it also plays a vital role in machine translation, information retrieval, text summarization, text clustering, and recommender systems. Oudah and Shaalan [28] stated that named entity in information retrieval can be used for the process of recognizing named entity in a user's query and within the documents to extract only relevant ones based on identified named entities. Mohan and Gupta [29] stated that NER can be used to automatically generate summaries of resumes by extracting only chief entities like name, education background, skill, and so on. Likewise, they presented that NER can be used in developing algorithms for recommender systems that automatically filter relevant content we might be interested in and accordingly guide us to discover related and unvisited relevant content based on our previous behavior. According to Cao et al. [30], named entity types drastically improves clustering quality as compared to the purely keyword-based VSM, for both hard and fuzzy clustering on testing data. Thus, NER is also useful in transforming the unstructured representation of text into a structured knowledge representation of databases.

According to Sanjana and Rupali [31], NER is challenged by various complexities that are inherent in any natural language. Similarly, Afaan Oromo NER has many challenges due to the structure of the language. These challenges include ambiguity, spelling variations, and nested named entities, and some other. For instance, in Afaan Oromo the word **Abdii Boruu** can be a name of a person or a name of organization based on the context of sequences.

All of the above mentioned the importance of NER application for various downstream tasks in NLP and its challenges motivated us to conduct this research. Afaan Oromo NER will be served as a downstream application for several Afaan Oromo NLP research. With this view in mind, the author will aim at developing NER for Afaan Oromo using deep neural network mainly by combining BiLSTM with CRF which uses neural word embeddings and character level representation as an input features.

1.3 Statement of the Problem

Previously, a number of studies has applied rule-based or supervised machine learning approaches to NER. These approaches depend on hand design feature engineering or domain-specific resources [32, 33, 34, 35]. Ghiasvand [3] stated that the rule-based method to NER lacks portability and robustness. The rule based method also requires a significant amount of human time and effort. To handle these problems, a machine learning model is proposed, which contains Hidden Markov Models [32], Conditional Random Field [34], and other as well. To this end, in supervised machine learning, annotation of the data can be a difficult, time-consuming and expensive process which makes it hard to apply supervised models in such manner. Despite the fact that classical machine learning models are able to achieve high accuracy, they might be failing to generalize for identifying new types of entities. Fortunately, deep neural network based NER is proposed that uses both label and unlabeled text which is expected to decrease the labor-intensive and the cost of developing such a system. Over time, a considerable number of studies have applied deep neural network approaches to NER such as [9, 11, 12, 15]. Deep neural network approaches for NER is able to tackle a problem of hand design feature in machine learning methods, which makes a feature extraction is no longer a bottleneck in machine learning tasks.

A few researchers have applied rule-based and machine learning for Afaan Oromo NER. We present their work as follows with their limitations.

Mandefro Legesse [20] employed machine learning technique named as Conditional Random Field for Afaan Oromo NER. To increase the performance of Afaan Oromo NER, Abdi Sani [21] proposed a hybrid approach which contains rule based and machine learning. The rule-based part includes parsing, filtering, grammar rules, whitelist gazetteers, blacklist, and gazetteers and machine learning techniques named as CRF. They improved the performance of the system by achieving 5.92% of F1, 8.3% of precision, and 3.8% of recall improvement of the baseline work. However, both of these systems highly relied on human designed features and linguistic resources. Generally, the shortcomings of their works are computational ineffectiveness, data sparsity problem, and minimal performance of the system because of the manual feature designed. Accordingly, it is advantageous to enhance Afaan Oromo NER by: (1) applying deep neural network with word and character embeddings,

which is expected to improve the effectiveness and efficiency of Afaan Oromo NER; (2) applying neural word embeddings to eliminate the need for linguistic resource and hand designed features; (3) combining BiLSTM with CRF to encode the context and decode the tag jointly. Thus, the main objective of this research is to conduct NER for Afaan Oromo using deep neural network with word and character embeddings, which do not need manually designed features such as POS taggers and external resources such as gazetteers, which is expected to have better results.

1.4 Objectives

General objective

The main objective of this research is to develop Afaan Oromo named entity recognition system using neural word embeddings.

Specific objectives

To achieve the goal of the general objective, the author will use the following specific objectives:

- To collect and prepare data for a corpus development;
- To conduct a literature review on named entity recognition;
- To generate features from Afaan Oromo texts using neural network;
- To develop a prototype;
- To evaluate the effectiveness of developed model and compare its performance with previous work baseline.

1.5 Methods

Data Collection and Corpus Preparation

The author will collect the raw text data set from different sources like new article, web and magazines. We will also collect the data from the Oromo wiki by downloading a dumped file. For the experimental case, we will collect and prepare both labeled and unlabeled corpora.

Literature Review

All-embracing literature review will be conducted for this study. Specifically, we will review the literature on brief overview and introduction about the Afaan Oromo structure like sentence structure, word segmentation. In addition to this, the author will conduct the literature review in the area of information extraction, particularly on NER and its application, approaches, feature, context encoder and decoder model.

Prototype development

We will develop a prototype to check our model for Afaan Oromo named entity recognition.

Evaluation

We will develop a model for Afaan Oromo NER using training data set and with different parameter. After that, the model will be evaluated by test data set using standard measurement metrics such as F-score, recall, and precision. Finally, the model will be compared with the baseline work.

1.6 Scope and Limitations

The scope of this thesis is to develop Afaan Oromo named entity recognition. We will employ the combination of BiLSTM as context encoder with CRF as tag decoder at the top of BiLSTM that uses distributed representation as an input feature. Its scope will be limited to the detection and classification of named entity such as the name of a person, organization, location, and miscellaneous.

1.7 Application of Results

Named entity recognition task plays a great role in information extraction in natural language processing application [1]. This task can be applied to different natural language processing applications such as information extraction, question answering, text summarization, text clustering, and information retrieval. The significance of this research is summarized as follows: first, it will be significant for us to gain a deep knowledge of named entity recognition and natural languages processing; second, after the researcher successfully conducted this study, it will be significant for another researcher who will develop natural languages application for Afaan Oromo; third, it also helps society and organization to extract structured

information from an unstructured one; finally, the researchers will hope that the produced artifact in the form of a model will be helpful for another researcher who will conduct further research on named entity recognition in Afaan Oromo. Ultimately, Afaan Oromo named entity recognition plays an important role in the development of Afaan Oromo natural language processing application.

1.8 Organization of the Rest of the Thesis

The remainder of this thesis document consists of five chapters and organized as follows. Chapter Two presents a brief overview of the literature review on Afaan Oromo. Moreover, it provides information extraction; mainly, NER and its approaches, features, applications and model used to encode and decode. Chapter Three presents the related work on NER for Ethiopian and non-Ethiopian languages. First, we will present NER for non-Ethiopian languages such as English, Russian, German Czech and some other. Second, we will present NER for Ethiopian languages, Afaan Oromo and Amharic. Lastly, the chapter provides the summary of related work. Chapter Four focuses on the design of Afaan Oromo NER. It presents particularly proposed approaches and the overall architecture of the model with its basic component and sub-component. The chapter also presents the discussion of the component it's their interaction within the system. Chapter Five explains experimentation and the result. This chapter is divided into five sections: experimental settings such as corpus preparation, performance measure, hyper-parameter settings; development tools and programming language; result like effect of different features; comparison with the previous work and discussion. Chapter Six presents conclusions from experimental observations, contribution and future works to show further areas of improvement for Afaan Oromo NER systems.

Chapter Two: Literature Review

2.1 Introduction

In this chapter, we provide a high-level overview of the most significant theoretical knowledge about the Oromo language and information extraction specifically named entity recognition. First, we provide a literature reviews on the Oromo language like sentence structure, part of speech particularly on nouns (maqaa), word segmentation and punctuation mark. Secondly, information extraction specifically, named entity recognition and its applications, approaches, feature, context encoder and tag decoder model will be reviewed. Finally, the summary of the chapter will be presented.

2.2 The Oromo Language

Afaan Oromo (literally Oromo mouth) or the Oromo language, belongs to the Cushitic branch of Afro-asiatic language phylum [36]. According to Gene Gragg [37], Afaan Oromo is probably the third-most widely spoken Afro-asiatic language in the world, next to Arabic and Hausa. Afaan Oromo has been spoken not only in Ethiopian but also spoken in neighboring countries like Kenya, Uganda, Tanzania, Djibouti, and Somalia.

Currently, Oromia regional state which is found in Ethiopian, employed Afaan Oromo as an official language. With respect to the writing system, Qubee becomes the official writing system for the Afaan Oromo since 1991. Qubee has become a name universally recognized in Ethiopia as the Afaan Oromo writing system based on the Latin alphabet. Afaan Oromo is now taught as an examinable subject in schools and employed as a medium of instruction in both primary and junior secondary schools in Oromia. It also offered as a minor and major subject of study at both undergraduate and postgraduate levels at Addis Ababa University, Wollega University, Jimma University, Harmaya University, and other Universities in Ethiopian.

Afaan Oromo has a very rich morphological structure like other African and Ethiopian languages [38]. In Afaan Oromo, most of the grammatical information is conveyed through affixes and other structures. The grammar of Afan Oromo exhibits gender, number, cases, and tenses like any other language. However, the grammatical presentation of Afaan Oromo is different from other languages and exhibits its own structure. Unlike English, Afan Oromo

gender,number,tense and other cases are described using affix. Therefore the grammatical rule is mostly dependent on the affixation rule of the language. For example:

Inni ar'a dhuf**e.** (He come today)

Isheen kaleessa dhuf**te.** (She came yesterday)

In the above two sentences the gender and tense of the sentences are described through suffix which is attached to the verbs dhuf-

2.2.1 Sentence Structure in Afaan Oromo

Unlike the English, sentence structure in the Afaan Oromo utilizes Subject-Object-Verb (SOV). Subject-Object-Verb (SOV) is a sentence structure where the subject, verb, and object come together respectively. For instance, Afaan Oromo sentence "**Nadhiin bilisa bahe**". In sequence of sentence, "**Nadhii**" is a subject, "**bilisa**" is an object and "**bahe**" is a verb. Hence, it has a subject-object-verb structure. The translation of the sentence in English is "**Nadhii has got freedom**" which has subject-verb-object structure.

2.2.2 Word Categories of Afaan Oromo

Based on the context, Afaan Oromo has five word categories such as noun, verb, and adverb, adjective and pre- and post-position [39].

Noun (Maqaa)

Afaan Oromo syntactic noun class are words that used to name or identify any of a class of things, people, places, or ideas and other as well. In Afaan Oromo, nouns are typically found at the beginning of sentence. For instance, all bolded words are a nouns in the subsequent sentences.

Caaltuun baratuudha. (**Caltu** is a student).

Geetaachoon gara Adaamaa deeme. (**Getachow** went to **Adama**).

The syntactic class noun can further be categorized into proper noun (maqaa dhuunfaa/matayyaa), common noun (maqaa gamtaa), collective noun (maqaa dimshaashaa), material noun (maqaa meeshaa), and abstract noun (maqaa killayyaa). However, our intuition in this research focuses on the proper nouns, which identify the specific name of a person, organization, location, and others as well. For example, **Gammaada, Caaltuu, Ayyaantuu**

(Gemeda, Caltu, Ayantu) are the names of the person**, Adaamaa, Assallaa, Bulee Horaa (Adama, Asella, Bule Hora)** are names of location and **Adaamaa Univarsiitii, Mana murtii (Adama University, Supreme Court)** are names of organizations. In written Afaan Oromo, proper nouns are always capitalized.

Verb (Xumura)

Verbs are a word that refer to actions and processes. In Afaan Oromo, verb occurs at the end of a sentence. For instance:

Caaltuun bartuu **cimtuudha.** (Caltu is a **clever** student).

Caaltuun gara Bulee Horaa **deemte**. (Caltu **went** to Bule Hora)

Adjective

Adjectives is a word class that modify a nouns or a pronoun by describing, identifying, or quantifying words. In Afaan Oromo, adjective follows a noun or a pronoun; their normal position is close to the noun they modify. For instance:

Rabbirraan farda **diimaa** bite (Rabira bought a **red** horse)

Pre- and Post-position

The term Pre- and Post-position refer to words, which will have meaning only when they are attached or used together with other words such as nouns, verbs, pronouns and adjectives.

For Instance:

Oromiyaa**tti** (in Oromia)

Oromiyaa**dhaaf (for** Oromia**)**

2.2.3 Afaan Oromo Word Segmentation

Afaan Oromo uses a white space character to separate words from each other words. For instance, "**Nadhiin Adaamaa deeme**". In this sentence, the word "**Nadhii**"," **Adaamaa**" and "**Deeme**" are separated from each other by a white space character. Thus, the action of taking an input sentence and adding valid word boundaries named as word segmentation which is operated using the white space characters.

2.2.4 Punctuation Marks in Afaan Oromo

Punctuation marks practiced in both Afaan Oromo and English are the same and used for the same purpose with the exclusion of the apostrophe. In Afaan Oromo, apostrophe mark (‘) represent a glitch (Hudhaa) sound, on the other hand in English, it illustrate possession. It plays a vital role in Afaan Oromo reading and writing system. In both language capitalization rules for detecting proper nouns with sentences enhance the process of named entity recognition. In Afaan Oromo, punctuation marks play a vital role in NER.

2.2 Information Extraction

As discussed in Chapter One, information extraction is the process of extracting information from unstructured text and turning it into structured text data. According to Daniel and Jurafsky [1] extracting information from unstructured text follows some steps: named entity recognition, relation detection and classification, event detection and classification, temporal expression detection and temporal analysis, and template filling. However, the main focus of this research is on named entity recognition, which is the first step in most information extraction task. We briefly discussed named entity recognition as follows:

2.2.1 Named Entity Recognition

Predominant to information extraction application is the task of named entity recognition, which involves the identification of proper names in texts (NER), and their classification into a set of predefined categories of interest (NEC) [40]. Unlike the pre-processing tools, which deal with syntactic analysis, NERC is about automatically deriving semantics from the textual content.

Grishman et al., [2] formally defined the task of NER in MUC6 as the task of identifying the names of all the people, organizations, and locations in a text as well as time, currency, and percentage expression. Previous MUCs focused on IE tasks; MUC-6 was the first comprising the NER task, which contained three subtasks:

- Entity names (ENAMEX): organizations, persons, locations;
- Temporal expressions (TIMEX): dates, times;
- Number expressions (NUMEX): monetary values, percentages.

Generally, NER is the task of identifying the mentions (or names) of entities in the text and assign semantic categories to them.

2.3 Applications of Named Entity Recognition

NER is an enabling technology for many applications. It acts as an important pre-processing step for a variety of downstream applications. This section briefly summaries application of NER as follows:

2.3.1 Relation Detection and Classification

Daniel and James [1] stated that relation detection and classification are the second steps for information extraction. The task of relation detection and classification is to find and classify semantic relations among entities discovered in a given text. Since, the relation is found between entities and concepts, recognizing named entities and or concepts is the first step for relation detection and classification [41, 42].

2.3.2 Temporal Expression Detection and Analysis

According to Daniel and James [1], the problem of figuring out when the events in a text happened and how they related to each other in time raises the twin problem of temporal expression detection and analysis. Extracting temporal expression detection and analysis requires the ability to recognize named entities that form integral parts of the temporal expression [43, 44].

2.3.3 Semantic Annotation

Textual annotation about the content of the document to formal identification of the concept and their relationships. For instance, a semantic annotation might relate the term **Adaamaa** to an ontology identifying it as an instance of the abstract Concept **City**, and linking it to instance **Ethiopia** of the abstract concept **country**, thus avoiding any ambiguity as to which **Adaamaa** text refers to. Semantic annotation is typically performed with Information Extraction techniques, among which NER is employed to recognize concepts to annotate.

2.3.4 Knowledgebase Generation and Population

Knowledge bases building involves extracting concepts and entities from the text and learning semantic relations between them. Therefore, knowledge bases building requires support from Named Entity Recognition.

2.3.5 Question and Answering

Question answering is the process of automatically finding the answer to a question expressed in natural language. According to Vikas and Steven [45], NER is a core component in many questions answering systems, in which it is used to recognize named entities in both the question and answer text. Molla et al. [46] presented that the fact-based question is named entities so incorporating named entity system improves the speed and accuracy of getting the correct answer.

2.3.6 Machine Translation

With integrations of Named Entity (NE) with neural machine translation has proposed by Arata et al. [47]. They demonstrate their model based on the encoder-decoder architecture. The exact identification of named entities (NEs) is an important problem for machine translation research and the development of commercial systems. Joining named entity recognition with machine translation has a great potential for improving the output quality [47, 48].

2.3.7 Information Retrieval

Oudah and Shaalan [28] stated that named entity in information retrieval can be used for the process of recognizing named entity in user's query and recognizing named entities with the documents to extract only relevant ones based on identified named entities.

2.3.8 Text Summarization

Automatic summarizers generate a summarized representation of the input text while maintaining relevant and important information content. Named entities are essential information of the text and enhance the performance of identification of text segments which are further included in summarized data. According Graeme et al. [49] and Nobata et al. [50], constructing text summarization uses the information about named entities to choose a better representative of text. Hence, Named Entity often hold the key representative information in the document, we can think that sentence contain named entities are vital for automatic text summarization.

2.3.9 Text Clustering

Tang et al. [30] presented text clustering with NER has the following advantages: named entity can be treated as special terms in which their meanings and relations are precisely pre-defined

in an ontology and knowledge base of discourse when they can make clustering results more semantically accurate for certain user needs; in some certain domains, such as news published in media and educational materials, named entities suggest meaningful representations for generated clusters because they capture salient points in document contents. Finally, they, concluded that NE types drastically improves clustering quality as compared to purely keyword-based VSM, for both hard and fuzzy clustering on testing data.

2.4 Named Entity Recognition Approaches

The primary aim of named entity recognition is to detect and classify names into predefined semantic categories. To achieve this aim one has to apply the NER approaches. This approaches are broadly classified into three main streams: Rule-based NER, linear statistical machine learning based approaches, (3) Deep-learning (non-linear statistical machine learning) based approaches, which automatically discover representations needed for the classification and/or detection from raw input in an end-to-end manner. Although, quite often the two techniques are mixed (see [21]). We discusses these approaches in detail as follows:

2.4.1 Rule-Based NER Approaches

Maynard et al., [51] stated that rule-based method for NER, which is used in ANNIE, GATE is information extraction system, typically, comprises a combination of gazetteer lists and hand-coded pattern matching rules. These rules determine whether candidate entities from the gazetteers are valid, or to extend the set of candidate based on contextual information. The gazetteers list act as an initial point from which to establish, reject, or refine the final entity to be extracted.

2.4.2 Machine Learning Based NER Approaches

Mitchell [52] presented that machine learning addresses the question of how to make machines to learn and automatically improve with experience. The linear statistical machine learning field builds and develops algorithms, which are used to automate tasks that are time-consuming or tedious for humans. We discussed in detail the linear statistical machine learning techniques as follows:

a. Supervised Learning

In supervised learning, the machine use manually labeled training data for extracting named entities. This method needs very accurate labeled data. The canonical example of supervised

machine learning model based NER includes Hidden Markov Models (HMMs) [32], Conditional Random Fields (CRFs) [34], Support Vector Machines (SVMs) [25], decision trees, and others as well. These types of methods work by categorizing a word or a phrase in a sentence whether it is a Named Entity or not.

Generally, the major limitation of supervised learning methods is dependence on a large amount of training data and hand design feature, which has to be created by the manual document annotation process. Therefore, the annotation process is usually difficult that can retire substantial investment in terms of both cost and personnel.

b. Unsupervised Learning

In contrast to supervised learning, unsupervised learning focused on algorithms for learning features from unlabeled data. Adam [53] investigated demystifying unsupervised feature learning, which the feature is learned from unlabeled data that is to train the data using only unlabeled data. In our case, we used semi-supervised learning, to learn the generalized representation of a word that is an effective way of handling data sparsity caused by high-dimensional lexical features in the NER. We obtained feature word representations from large unlabeled data via an unsupervised learning algorithm.

c. Semi-Supervised Learning

A supervised method mainly relied on linguistic resource and manually designed features. It substantial requires labor to prepare an annotated corpus. Annotating data can be difficult, time-consuming and expensive process which makes it hard to apply supervised model in such case. Using a supervised learning method, it is impossible to obtain sufficient labeled data for all name entity and it is also limited by the problem of data sparsity. In these case, semi-supervised learning can assist to make use of both annotated and easy-to-obtain unannotated dataset. Sogaard [54] stated that the idea behind semi-supervised learning is that we can exploit the marginal distribution of unlabeled data, typically available in larger volumes than labeled data, to learn better models than with labeled data alone. Semi-supervised learning methods are very effective in using large-scale unlabeled data and have successfully improved performances of the existing system of supervised systems on a deferent natural language processing task such as NER [19, 54, 55]**,** dependency parsing [56, 57] and so on.

2.4.3 Deep Learning Based NER Approaches

Goldberg [40] explains deep learning as a re-branded name for neural networks. It is a family of learning methods that were traditionally motivated by the way computation works in the human brain, and which can be characterized as learning of parameterized differentiable mathematical functions. They presented that all of the machine learnings can be characterized as learning to make a prediction based on past observations, however, deep learning approaches work by learning to not only predict but also to correctly represent the data, so that it is suitable for prediction. In addition to this, a deep learning approaches work by feeding the data onto a network that produces successive transformations of the input data until a final transformation predicts the output. In deep learning, embedding layers are a major component to learn the context word and represent as continuous vectors in a relatively low dimensional space.

In recent years, most of the recent work has advanced with NER models based on deep learning, which become dominant and yields a better results. Compared to linear statistical machine learning, deep neural network learning is helpfully generate hidden features automatically. The fundamental benefit of deep learning is the ability of representational learning and the semantic composition empowered by both the vector representation. For some time, linear statistical machine learning approaches targeting natural language process have been based on a shallow model trained on very high dimensional and sparse features. In contrast to non-deep learning, the main advantages offered by deep learning techniques is the requirement for little or no handcrafted features for natural languages task, specifically, for resource-scarce languages as many resources are not readily available. In recent years, neural networks based with representational learning like word embeddings have been produced and yields superior results on various NLP tasks. For instance, NER [23, 58, 59], dependence parsing [56]. Thus, this trend is produced by distributed representation of words specifically, by employing neural word embeddings [25, 56, 60, 61].

Several studies suggest that deep neural network learning based techniques empowered by word embedding has been employed in NER and yields good performance [8, 17, 62].

2.5 Named Entity Recognition Features

Broadly speaking, supervised NER models depend on a feature to represent the input sequence of words. Thus, choosing the right feature set has the highest importance for marking the presence and categories of NER. In machine learning, the features can be divided into binary, categorical, ordinal, real-valued, and so on. Most machine learning based NER models are widely dependence on features like word-level feature (e.g., case. Morphology, and POS tag) [21, 35], list lookup features (e.g., gazetteer) [20, 21], document and corpus features (e.g., local syntax and multiple occurrence) and External features (e.g., distributed representation). For further feature designs information for NER refer that works Nadeau and Skein [64]. In this study, we used distributed representation of word as a feature input. Hence, we explained the distributed representation in section 2.5.1.

2.5.1 Distributed Representation

Mostly, rule-based and statistical language processing algorithms regard words as atomic symbols [40]. This translates to a very sparse vector representation of the size of the vocabulary and with a single one at the index location of the current word. However, this type of learning often suffers from the notorious curse of dimensionality while learning joint probability functions of language models.

Consider the following example for more detail of "one-hot "representation:

Text1= "**Nadhiin gara Adaamaa deeme**"

When Text1 is converted to feature vector, it looks like as shown in Figure 2.2:

	Adaamaa	Nadhiin	deeme	gara
0	0	1	0	0
1	0	0	0	1
2	1	0	0	0
3	0	0	1	0

Figure 2.1: One-Hot Vector Word Encoding

The version of the this sentence in English is

Text1= "**Nadhi went to Adama**" and when it is converted to feature, it looks like as following

	Adama	Nadhi	to	went
0	0	1	0	0
1	0	0	0	1
2	0	0	1	0
3	1	0	0	0

This, so-called "one-hot" or "one-on" representations have a problem that does not capture any type of similarity between two words. As we have seen in the above example, if the size of the vocabulary is |Y| then word D can be represented as a vector of size |Y| in the index of word D is only one and remain are set to zero.

For example, consider the following two sentences for named entity tagging problem:

Text1= Nadhiin aapilii nyaachaa jiraa. **Nadhi is eating an apple**

Text2 = Nadhiin aapilii fayaadamaa jiraa. **Nadhi is using an apple**

If we consider the above Text1 and Text2, **aapilii** form the sentence, it gives different meanings when it is used with different (close by) adjacent words, **nyaachaa** (eating), and **fayaadamaa** (using). How do you make a machine recognize that "Apple" in "Apple is a tasty fruit" is a fruit that can be eaten not a company? The answer to such type of question lies in creating a representation of words that capture their meanings, semantic relationships, and the different types of contexts they are used in. This types of problems are addressed by distributed representations of word. These led many researchers to learn distributed representations of word existing in low-dimensional space. According to Mikolov et al. [61], word embedding is a feature learning methods that automatically generate word vector from the vocabulary. Over time, a considerable number of studies have applied distributed representation as input feature for deep learning based NER: neural word embeddings, character level representation and hybrid of character and word embeddings. Thus, we discuss distributed representation as follow:

Neural Word Embeddings Representation

According to Goldberg [40], embeddings layer is a main component of the neural network that maps discrete symbols to continuous vectors in a relatively low dimensional space and dense vector. Guo et al. [25] stated that word embeddings are conventionally defined as dense, continuous, and low-dimensional vector representations of words. Through the context-predicting model (example, neural network language models) or spectral methods (for example, canonical correlation analysis) large scale unlabeled text can be learned by utilizing word embeddings. In word embeddings, embedding is the vector representation of each core feature of the word that can be hopefully capture useful syntactic and semantic properties. Consequently, word embedding can be beneficial for NER which is the most simple and general way is to be fed as features to alleviate to some extent the discreteness and data-sparsity problem and to enhance the existing supervised machine learning process. In the task of NER, semantic information of the sequence is important that is captured by distributed representation. The basic idea of these features is simple. If we know that a word in Afaan Oromo **mummicha ministeraa (prime minister)** is usually followed by a person name, then the word**mootummaa** which is semantically similar to **mummicha ministeeraa"** should be also be followed by a person name. Of course, this idea can be extended in many ways. Thus, word embedding is introduced for handling such types of problems. Mikolov et al. [61] investigated the powerful idea of representing each word utilizing its neighbors. They proposed two architectures for efficient neural network language models. We consider these two a context-predicting model, in particular, the Skip-gram model and Continuous Bag-of-Words model for learning word embedding since it is much more efficient as well as memory-saving than other approaches.

Skip-Gram Model

Mikolovet al. [61] induced the Skip-gram model that predicts the likelihood of context words occurring given a center word. The training aims of this model to efficiently learn high-quality context vector representation of words from large amounts of unstructured text data set. This model assumes that context words are generated based on the central target word. For instance, consider the given sentences **“Nadhiin”,” ilma”, “isaa”, “jaalata”** the translation of the text sequence into English is **“Nadhi”,” loves”,” his”, “son”.** If the given central target word **jaalata**, the skip-gram model is concerned with the conditional probability for generating the

context words, **Nadhiin, ilma, isaa** that are within a distance of no more than two words, which is

P (“Nadhiin”,” ilma”, “isaa” | “jaalata”)

For a given target, the skip-gram predicts the context of words that are independent of each other. In such circumstances, the expression above can be rewritten as:

P (“Nadhii’’ | “jaalata’’) · P (“ilma" | “jaalata’’) · P (“isaa" | “jaalata’’)).

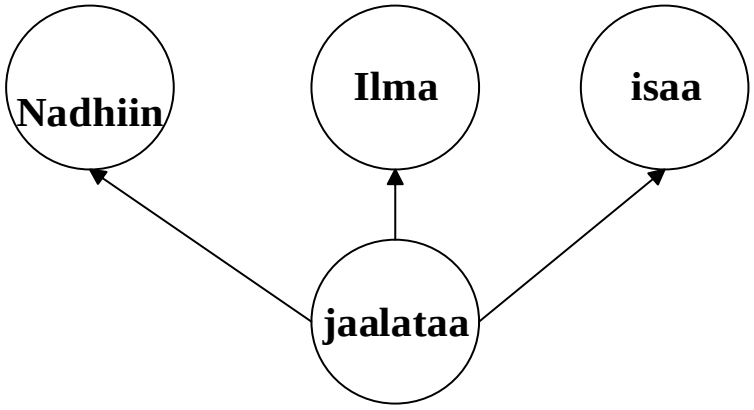


Figure 2.2: Skip-Gram model to generate context

The skip-gram model cares about the conditional probability of generating context words for a given central target word as depicted in Figure 2.3. **Jaalata** is our target word and **Nadhii**, **ilma, isaa** belongs to the context of **jaalataa**. The notion here is that with enough examples of contextual similarity, the network can learn the correct associations between words.

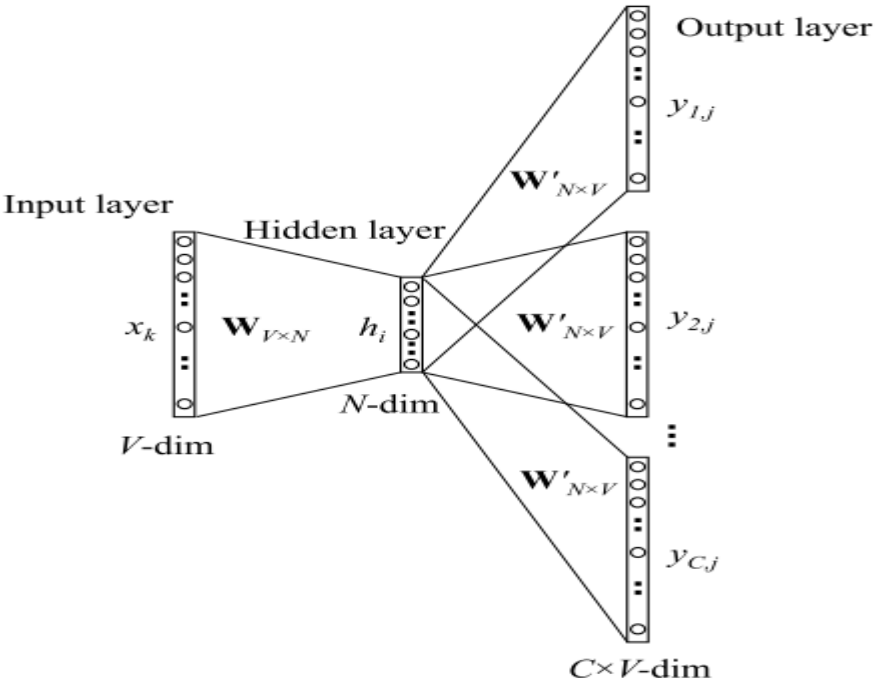


Figure 2.3: The skip-gram model

Figure 2.4 depicts the skip-gram model introduced by Mikolov et al. [61, 65]. As seen in Figure 2.3, $x_{k \text{ i is}}$ input vector in terms of one-hot encoding with a *V*-dim a size of vocabulary. The input layer are fully-connected network to the hidden layer through a *VxN* weight matrix $\boldsymbol{W}$. The hidden layer which represented as h_i fully-connected network to the output layer which represented as y_{1j}, y_{2j}, y_{cj}, where C is context window through *NxV* weight matrix $\boldsymbol{W'}$

The skip-gram model parameters are the target and context words vector for each words. Suppose that a given sequence of word w_1, w_2, w_n, the Skip-gram maximize the average log probability is given by the following formula:

$$\frac{\mathbf{1}}{\boldsymbol{N}}\sum_{\boldsymbol{n=1}}^{\boldsymbol{n}} \sum_{\boldsymbol{-s<j<s, j\neq 0}} \log \boldsymbol{p}\,(\boldsymbol{Wn+j|Wn}) - - - - - - - - - (1)$$

Where s is the size of the training context; we are going to learn the model parameters by maximizing the likelihood function, which is also known as maximum likelihood estimation. The interior summation goes from −s to s to calculate the log probability of correctly predicting the word given the word in the middle. The exterior summation goes throughout all words in the training data set.

Continuous Bag-of-Words Model

According to Mikolov et al. [61], the continuous bag-of-words model uses a feedforward neural network language model, with no non-linear hidden layer and a shared projection layer for all words. The model utilizes both words from the left and right (previous w tokens and next w tokens). The bag-of-words name comes from the fact that the position of the words in the window does not matter in the long run to determine the embedding. Figure 2.5 depicted the CBOW neural network architecture.

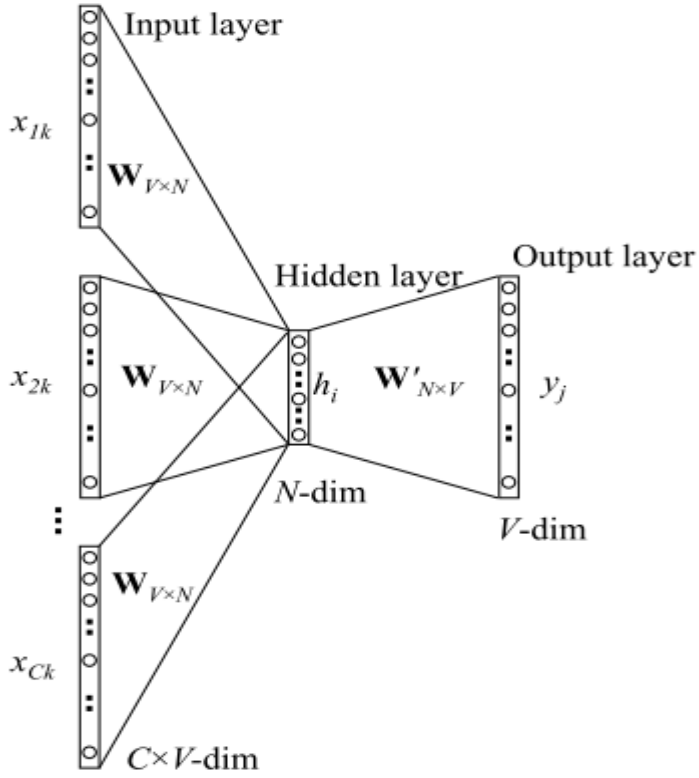


Figure 2.4: The CBOW Model

Figure 2.5 illustrates the CBOW model that introduced in Mikolov et al [61, 65]. In Figure 2.5, C represent a size of window and *V* represent size of vocabulary. The input is one-hot encoded context words (X_1, X_2, XC) with *VxN* weight matrix. The input vectors are fully-connected network to the hidden layer through a $V \times N$ weight matrix **W**. The hidden layer is fully-connected network to the output layer which represented as y_j through NxV weight matrix **W'**

Character Level Representation

Considerable number of studies have applied character level representation of word with word embeddings as an input feature for context encoder model in deep neural network based NER [13, 66]. This representation has been found helpful for utilizing explicit morphologic information such as prefix and suffix. Furthermore, character-level representation is able to controls out-of-vocabulary. Therefore, deep learning based NER that use character-based as input feature representation capable to predict the representations for unseen words during training and share information of morpheme-level regularity. Ma and Hovy [67] extract character-level representation of word using CNN. Finally, the character representation vector is concatenated with the word embeddings before feeding into a neural network.

2.6 Context Encoder Models

In deep neural network based NER, context encoder is the second stage that learn the context of the sequence from the input representation. In this section, we explore a context encoder model that related to our study in the subsequent section.

2.6.1 Recurrent Neural Networks (RNNs)

Moreno and Jugal [68] described that traditional neural networks have a major limitation in considering sequential relation of inputs and outputs. The sequential relation of input and output are unconnected to each other. To address the problem of traditional neural networks, recurrent neural networks have been proposed. Mikolov et al., [69] investigate recurrent neural networks which are powerful in modeling sequential data. They deal with sequential data prediction problem when constructing language models. Recurrent neural networks are being used for a broad variety of tasks. For instance, named entity recognition, recurrent neural network can be used to predict the probability of a word being named entity or not. According to Mikolov et al [69], recurrent neural networks are called recurrent because they operate similar task for every element of a sequence, with the output being reliant on the preceding computation. Additionally, it contains memory about what has been calculated so far and used it on current output computation.

Ashine and Shervin, [70] states the pros and cons of a typical RNN architecture are summarized up in the Table 2.1:

Table 2.1: Advantages and Drawbacks of RRN Architecture

Advantages	Drawbacks
• Possibility of processing input of any length • Model size not increasing with the size of the input • Computation takes into account historical information • Weights are shared across time	• Computation being slow • The difficulty of accessing information from a long time ago • Cannot consider any future input for the current state

2.6.2 Long Short-Term Memory (LSTM)

According to Hochreiter and Schmidhuber [71], in theory, RNNs can make use of information in arbitrarily long sequences, but in practice, they fail due to the gradient vanishing or exploding problems. They stated that the Long Short-Term Memory is a neural architecture that is designed for storing and accessing information better than standard recurrent neural networks. As an elite kind of RNN, Long Short-Term Memory (LSTM) neural network has been established to overcome both vanishing and exploding gradient problem. LSTM can be powerful in modeling sequential data. Therefore, it can learn long-term dependencies. Long Short-Term Memory consisting of three gates, forget and output gates as shows in Figure 2.4 Young et al [72].

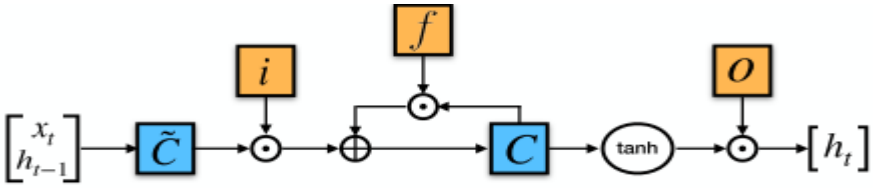


Figure 2.5: Long Short-Term Memory

It calculates the hidden state by taking a combination of these three gates (input, forget, and out get) as per the mathematical equations depicts in equation 2 up to 7.

$$x = \begin{bmatrix} \mathbf{h_{t-1}} \\ \mathbf{x_t} \end{bmatrix} \quad (2)$$

$$f_{t=} \sigma\left(w_{f.x+b_f}\right) \quad (3)$$

$$i_{t=} \sigma\left(w_{i.x+b_i}\right) \quad (4)$$

$$o_t = \sigma(w_{o.x} + b_o) \quad (5)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh(w_{c.X} + b_c) \quad (6)$$

$$h_t = o_t \odot \tanh(c_t) \quad (7)$$

Where σ denotes the sigmoid function; $\odot$ is the element-wise multiplication operator; **w** terms denote weight metrics; ***b*** is biased vectors; and ***i, f, o*** denote input, forget, and output gate respectively and ***c*** are cell activation vectors. The term ***h*** stands for the hidden layer nodes.

2.6.3 Bidirectional LSTM

The main shortcoming of an LSTM merely account for context from the past, that is the hidden state, h-t, takes only past information as input feature. To alleviate the limitation of LSTM, BiLSTM is proposed by Schuster et al. [73]. The author stated that the BiLSTM network is a modification of the LSTM, which is designed to capture information of sequential data and have access to both past and future contexts. BiLSTM contains two LSTM states, a forward and backward LSTM that encode the context of sequence data in two opposite direction. The forward LSTM reads a stream of input data and computes the forward hidden states, while the backward LSTM reads the sequence in reverse order and creates the backward hidden states. The idea behind BiLSTM is creating two separate hidden states for each sequence so that the information of sequences from both directions is memorized. The final output of BiLSTM is concatenation of the two hidden states. This advantage of memorizing information for long periods in both directions can make a great improvement in linguistic computation. The concatenation of output of BiLSTM forward hidden layer ($\vec{h}_t$) and backword hidden layer ($\overleftarrow{h}_t$), to yield an output ***y*** computed as:

$$\vec{h}_t = \sigma(W_{x\vec{h}_t} Xt + W_{\vec{h}_t} \vec{h}_{t-1} + b_{\vec{h}_t} \quad (8)$$

$$\overleftarrow{h}_t = \sigma(W_{x\overleftarrow{h}_t} Xt + W_{\overleftarrow{h}_t \overleftarrow{h}_t} \overleftarrow{h}_{t-1} + b_{\overleftarrow{h}_t} \quad (9)$$

$$y_t = W_{\vec{h}y} \vec{h}t + W_{\overleftarrow{h}_y} \overleftarrow{h}_t + b_y \quad (10)$$

Where $Xt \in R$N-dimensional input vector at time t, W is is the weights matrices, b is bias vectors, and $\vec{h} \in R^N$, $\overleftarrow{h} \in R^N$ are the output of hidden layer of forward and Backward LSTM respectively.

In the case of NER, it is advantageous to have encoded context history from the past as well as the future, or context of sequence from the left and right. This can be alleviated with BiLSTM that process context information in two LSTMs that can process context information in forward and backward manner in opposite direction.

In this paper, we experiments the main architecture of BiLSTM to capture the context of the sequence and automatically generate a feature from Afaan Oromo text to eliminate the need for most feature engineering. We investigated neural word embedding that used as input for BiLSTM for modeling Afaan Oromo NER.

2.7 Tag Decoder Model

In deep neural network based NER, the tag decoder is the final stage that takes context-dependent representation from the encoder as input and yield a sequence of tags corresponding to the input sequence. Thus, we explore a tag decoder model that related to our study in the subsequent section.

2.7.1 Conditional Random Fields (CRF)

According to Lafferty et al. [5], Conditional Random Fields is undirected graphical model that used to perform the mutual constraint between tags. CRF models takes advantages of the correlation between neighborhood tag into account and decode them jointly. For deep neural network based NER task, the CRF utilized as tag decoder. For instance, in deep neural network based NER, the CRF used on the top of Bidirectional LSTM layer [12, 13, 74] and on the top of a CNN layer [18, 66]. Likewise, we also use CRF as tag decoder model for modeling Afaan Oromo NER. Despite the CRF models take benefit of the correlation between labels, it heavily relies on hand crafted features and task-specific resources. Subsequently, Bidirectional LSTM network utilizes both left (past) and right (future) features automatically, we propose a hybrid of BiLSTM and CRF network to form a BiLSTM-CRF model based on the work of Huang et al, [74].

For an input sentence **S** = (s1, s2, · · ·, sn), we use **N** to represent the output features of Bi-LSTM. The output dimensionality of Bidirectional LSTM **N** is the size of n × k, where k is the number of distinct tags, and Ni, j corresponds to the score of the j^{th} tag of the i^{th} token in the input sequence. For a sequence of predictions **y= (**y1, y2, y3, … yn), we define the score function as the following:

$$\mathbf{s(S, y)} = \sum_{\mathbf{i=0}}^{\mathbf{n}} \mathbf{A_{y,y+1}} + \sum_{\mathbf{i=1}}^{\mathbf{n}} \mathbf{N_{i,yi}} \quad (11)$$

Where A is the transition matrix of CRF such that $A_{i, j}$ represents the score of the transition from the tag i to tag j. y0 and yn are the commencing and ending tags of a sequence. In lieu of capturing hand crafted features, we use context vector output of BiLSTM as an input to feature to CRF model that decodes it into the desired name entity. Similarly, the conditional probability of sequence ***y*** given input ***S*** turns out to be the following:

$$\boldsymbol{p(y|S)} = \frac{\mathbf{e}^{\mathbf{s(S,y)}}}{\sum \boldsymbol{\hat{y}} \in \mathbf{Yx}\ \mathbf{e}^{\mathbf{s(S,\hat{y})}}} \quad (12)$$

Where Y is the vocabulary of all possible label sequences of given input ***x*** throughout the training, we maximize log-likelihood of the correctly labeled sequence as:

$$\boldsymbol{log\big(p(y|S)\big) = s\,(S, y) - log\big(\big(\sum_{\hat{y} \in Y} e^{s(S,\hat{y})}\big)\big)} \quad (13)$$

$$\boldsymbol{= s\,(S, y) - \underset{\hat{y} \in Yx}{logadd}\, s(S, \hat{y}\,)} \quad (14)$$

While decoding, we predict an output sequence that achieves the highest scores and predicts the best tag path as follows:

$$\boldsymbol{y^* = \underset{\hat{y} \in Yx}{argmax}\, s(S, \hat{y}\,)} \quad \mathbf{(15)}$$

Chapter Three: Related Work

3.1 Introduction

Researchers on NER in the previous years can be divided into two directions. The first is to advance existing models (e.g. rule-based, machine learning models, hybrid of rule and machine learning) that use hand design feature for NER. Another direction, which gains popularity in recent years is the use of deep neural network learning that uses automatically discovered features like neural word embeddings and character level representation for NER model. This work also falls in this direction. We include pre-trained word embeddings and character level representation, which is expected to improve the effectiveness and efficiency of NER systems. In this chapter, we present NER for non-Ethiopian and Ethiopian languages.

3.2 Named Entity Recognition for non-Ethiopian languages

Numerous studies have attempted to conduct NER for no-Ethiopian languages like for English, Russian, Vietnamese, Dutch, German, Spanish, Turkish, Czech, English and other as well. The solution involves both supervised and semi-supervised learning. Recent studies have been conducted using artificial neural networks.

Jason and Nichols [15] employed artificial neural networks specifically, BiLSTM with CNNs architecture to create a NER model and eliminate the need for feature engineering. They applied CNN to capture the character level representation that was concatenated with the word embeddings of the word and feed into the BiLSTM for context vector learning and predicting the name entity. They employed different feature like word embeddings, capitalization and lexicon. Their neural network model was inspired by the work of collebert et al. [14]. Previously, character level representation was applied to Spanish and Portuguese NER by Santos and Guimaraes [10]. Experimental results showed that BiLSTM-CNNs, which obtained 91.62% F2 score on CoNll-2003 and 86.52% F1 scores on OntoNotes 5.0 dataset, surpassing system that heavy relied on feature engineering.

Strakova and Hajic [11] also proposed neural network based NER, mainly for the Czech language. They conclude that a recurrent neural network-based recognizer with word embeddings and character-level word embeddings achieved excellent results for Czech NER. Their proposed model used lemmas, POS tags and word embeddings as feature. Their model

achieves better results by using combination of character-level and word embeddings, prefixes, and suffixes. The highest F-measure achieved was 89.92%.

Suzan Usual et al., [75] presented new results of 93.59% and 79.59% F-Score for Turkish and Czech named entity recognition based on the model of Lample et al. [17]. For the experiments, they used features like character-based word representation and word vectors. Yamada et al., [60] used neural network, word-embeddings, character-level word embeddings for NER and they presented comparable results. Their method works by jointly mapping words and entities into the same continuous vector space. The classical skip-gram model is enhanced by the KB graph model that learns the relatedness of entities using the link structure of the knowledge base and by presenting context feature that objectives to align vectors so that related words and entities occur close to one another in the vector space.

Wang et al., [76] proposed way to use Bi-LSTM-RNN with word embedding for tagging problem. In their work, they employed automatically generated word features and a recurrent layer to the process of predicting sequence tagging problems. This system avoids involving human made feature design, instead, it utilizes word embeddings learned automatically from the unlabeled text. They achieved 89.64 % F-score for NER. Similar system was proposed by Lample et al. [17] for NER using CRF and Bi-LSTM. They demonstrated neural architectures for NER that did not use language-specific resources or features beyond a small amount of supervised training data and unlabeled corpora. They employed two sources of feature information about the words; character and neural word embedding which learned from the supervised and unsupervised corpora respectively. They used English, Dutch, German, and Spanish corpus to evaluate their developed model. The results achieved were 90.94%, 81.74%, 78.76%, and 85.75% of F-scores of English, Dutch, German, and Spanish respectively.

Pham and Phuong [77] proposed end-to-end deep neural network architectures for Vietnamese NER. They proposed an end-to-end deep learning model which achieves a good performance on a standard NER data set for Vietnamese language. Their best model is a combination of Bi-LSTM, CNN, and CRF models, which achieves F1 score of 88.59% by using pre-trained word embeddings as input. They examined three approaches to generate word embeddings: a skip-gram model, a CNN model, and Bi-LSTM model.

Arkhipov et al., [12] studied the application of a hybrid Bi-LSTM-CRF model to the task of Russian NER. According to their results demonstration, the basic Bi-LSTM model is not adequately surpass the existing works of NER. Then, they add the CRF layer to the top of Bi-LSTM network and significantly increases the performance of the model. To improve performance further and achieve good result for the Russian NER, they finally add external word embeddings to their model. The overall architecture of their NER is depicted in Figure 3.1. Here, x1 and x2 are representations of the word in a sequence. It feeds into character and word level embedding blocks. Then character and word-level representations are concatenated into c1 and c2. Bi-LSTM performs conditioning of the concatenated representations in forward and backward contexts. Lastly, CRF layers give output tag predictions yi.

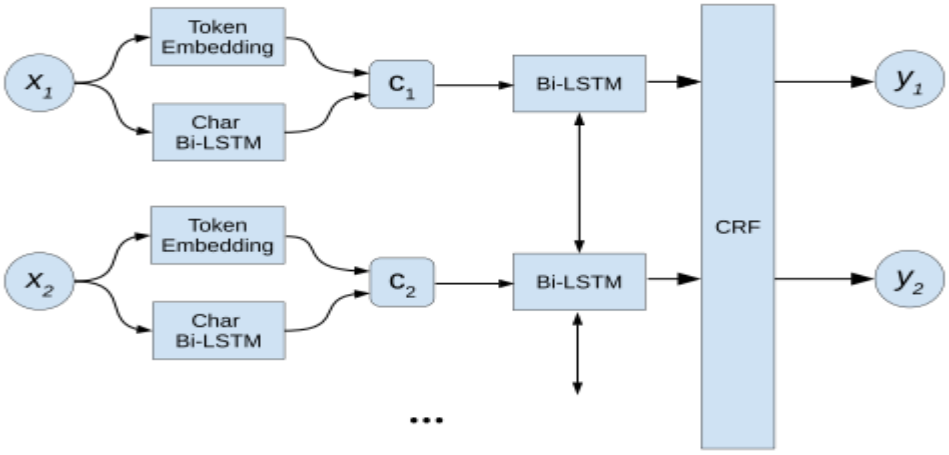


Figure 3.1: The Architecture of Bi-LSTM Neural Network for Solving Russian NER Task

They typically found that the extension of the Bi-LSTM model with CRF radically improved the quality of NER predictions. They said that input tokens with external word embeddings minimize training time and allowed to achieve good result for the Russian NER task. The performance evolution of their models was accomplished through three Russian datasets: Gareev's, FactRuEval 2016, and persons-100 and their result for Bi-LSTM+CRF+Lenta word embedding of F1 82.10%, 87.17% and 99.26 respectively for these three datasets. Because of the use of pre-trained word embeddings, the proposed model is not robust for the out-of-vocabulary words.

Hakan and Ozgur [55] proposed semi-supervised learning-based NER in morphologically rich languages based on distributed word representations and neural networks. The main feature

they utilized for NER model is a neural word embeddings learned in unsupervised approaches and they seed a word vector features to neural network for the development of a NER system. To obtain the continuous space vector representations of words, they adopted the Mikoto et al. [61]. They evaluated their system for Turkish and Czech languages, which is highly inflectional. They achieved better result for Turkish and Czech. Contrasting, the previous existing systems developed for these languages, their system does not make use of any language-dependent features.

Ping et al., [66] investigated the influence of character feature learning on NER tasks and proposed two models of learning local semantic and position-related features in word characters by concatenation and stacking. They employed deep neural network of the task of NER and solves the shortcoming of the traditional model that heavily depends on complex feature engineering. They developed a CNN+BiLSTM concatenation model to learn the local semantic and a CNN+ BiLSTM stack model to learn the position features of word sequences. A character features are concatenated together with word embedding to form the input vector. They evaluated their system by using the CoNLL-2033 English dataset. Experimental results showed that the CNN+BiLSTM concatenation and CNN+LSTM stack models, which obtained 91.58% and 91.52% F1 scores on the CoNLL-2003 dataset, respectively, possessed a strong learning capability and outperformed the existing models.

3.3 Named Entity Recognition for Ethiopian languages

A considerable amount of work has been published on NER for technologically advanced languages like for English, Chinese, Russian and other as well by employing the recent deep neural network. In Ethiopian languages, however, most of the existing works employed classical machine learning such as Conditional Random Field (CRF) and rule-based approaches for Afaan Oromo [20, 21] and Conditional Random Field, Support Vector (SV), and J48 for Amharic [22, 24, 35]. Some of those researches which are closely related to this work is discussed as follows:

Named Entity Recognition for Afaan Oromo

Afaan Oromo NER has many challenges. Among these challenges ambiguities in proper nouns, lack of labeled data, scarcity of linguistic resources and being morphologically rich,

spelling variations and nested named entities are common ones. These challenges need to be addressed while developing Afaan Oromo NER.

As related work shows, there are two research conducted for Afaan Oromo NER. The first NER for Afaan Oromo was conducted by Mandefro Legesse [20]. They employed a machine learning algorithm named as Conditional Random Field as classifiers underlying for Afaan Oromo NER architecture. A corpus of size 23,000 words has been used for training. An average performance of recall, precision and F1-measure 77.41%, 75.80%, and 76.60% achieved respectively.

The second research was conducted by Abdi Sani [21]. They employed the hybrid of machine learning and rule based approach for Afaan Oromo NER. The rule based components include feature like gazetteer, blacklist gazetteers, exact matching, and grammar rules. The classifier used as an underlying for the system architecture is a Conditional Random Field from machine learning component. The same corpus was used with the first researcher. An average performance of recall, precision and F1-measure 81.21%, 84.12%, and 82.52% was achieved respectively.

Over all, the rule-based lack portability and robustness. The supervised machine learning extensively relies on hand design feature and need a manual annotation process that needs human labor and time. In addition to this, there remain some major unsolved problems and challenges which the most widely concerned is the data sparsity problem.

Named Entity Recognition for Amharic

Moges Ahmed [35] employed machine learning techniques named Conditional Random Field (CRF) for Amharic NER. They used the feature sets like prefix, tokens, suffix and speech tag of tokens, and word and tag context features. They conducted a different combination of these features to determine the best performing feature sets. They got an average performance of precision recall and F-measure of 72%, 75%, and 73.47% achieved respectively.

Similarly, Besu Fekad also [22] used machine learning techniques named Conditional Random Field for Amharic NER. For the experiments, they used the following features like named entity tag of a word, word pairs, word shape, prefix, and suffix. The highest F-measure achieved was 80.66%.

Mikiyas Tadele [24] used hybrid approaches for Amharic NER. The classifiers they used as an underlying Amharic NER architecture were Decision Tree (J48) and Support Vector machine. They achieved an average performance of 96.1% highest of F-score for decision tree and 85. 9% for SVM.

Dagimawi Demissie [23] proposed Amharic NER using neural word embeddings as a feature. They achieved the maximal performance of Amharic NER as compared to the previous research. The author used the different classifiers with word vector features, such as SVM, J48, random tree, IBk (Instance-based learning with parameter k), attribute selected, and OneR (one rule). They obtained the highest F-score 95.5% using the SVM classifier. The authors stated that using word embedding and deep neural network classifiers is achieved the maximal performance of NER on different groups of classifiers.

3.4 Summary

As related work shows, the neural network with a Conditional Random Field based NER achieved the highest result without using manual features design. Likewise, incorporating word embedding and character-level representation with neural network architecture achieved good results as compared to the system that used hand design feature. For instance, the combination of BiLSTM network with CRF with pre-trained word and character level representation achieved a good result for English, German, Dutch, Spanish, and Russian and for other languages as well. In addition to BiLSTM-CRF, the use of the BiLSTM-CNN-CRF model also successfully achieved a good results for Vietnams, English, Chines, Italian, and some other languages. A lot of the researches have been done in this area used the same approach with some modifications to improve the accuracy of their model.

Despite a lot of works have been conducted on NER for non-Ethiopian, only limited work was found in Ethiopian languages in Afaan Oromo [20, 21] and Amharic [22, 24, 35]. However, these researchers utilized linear statistical machine learning models which rely on hand design feature engineering and linguistic task specific resource. On the other hand, recently, many researchers proposed deep neural network based NER to deal with the drawback of linear statistical machine learning. Neural network is able to extract feature automatically rather than creating manually designed features. The hand design feature engineering is also task-dependent and empirically requires analysis for each new NLP task.

The manually designed feature also suffers from data sparsity problem. To address this problem, an effective way is to learn a more generalized representation of words by taking advantage of the numerous unlabeled training data. Therefore, our attempt is expected to improve the existing study by using deep neural network. The deep neural network uses distributed representations of a word as an input feature that solves the curse of dimensionality while learning joint probability functions of the language mode. For Afaan Oromo NER, however, no studies have investigated the impact of deep neural network that uses automatically generated character level representation and pre-trained word embeddings as input features. Therefore, we attempt to reduce the handcrafted feature engineering by using distributed representation techniques like neural word embeddings and character level representation.

Chapter Four: Design of Afaan Oromo NER

4.1 Introduction

In this Chapter, we present the work of the overall design of Afaan Oromo NER. Further, we explain system architecture and its subcomponents like preprocessing that is used to make data ready for annotated named entity and neural word embeddings, encoder and decoder model.

4.2 System Architecture

We propose a deep neural network based approach for Afaan Oromo NER, which is a combination of the BiLSTM-CRF algorithms that benefits from pre-trained neural word embeddings and character level representation. The proposed approach for Afaan Oromo NER captures the word features from unlabeled text dataset automatically rather than hand design features through unsupervised learning. From existing deep neural networks, we use BiLSTM as context encoder of the sequence of words. On the top of BiLSTM, we also added CRF as jointly decode the best tag sequence for the labeled sequence of an input text. This model uses a pre-trained word embeddings and the labeled named entity as an input feature. The proposed NER model contains pre-processing, encoder, decoder and named entity recognizer. The components of proposed system are depicted in Figure 4.1 and the explanation of subcomponents are described in the subsequent subsections.

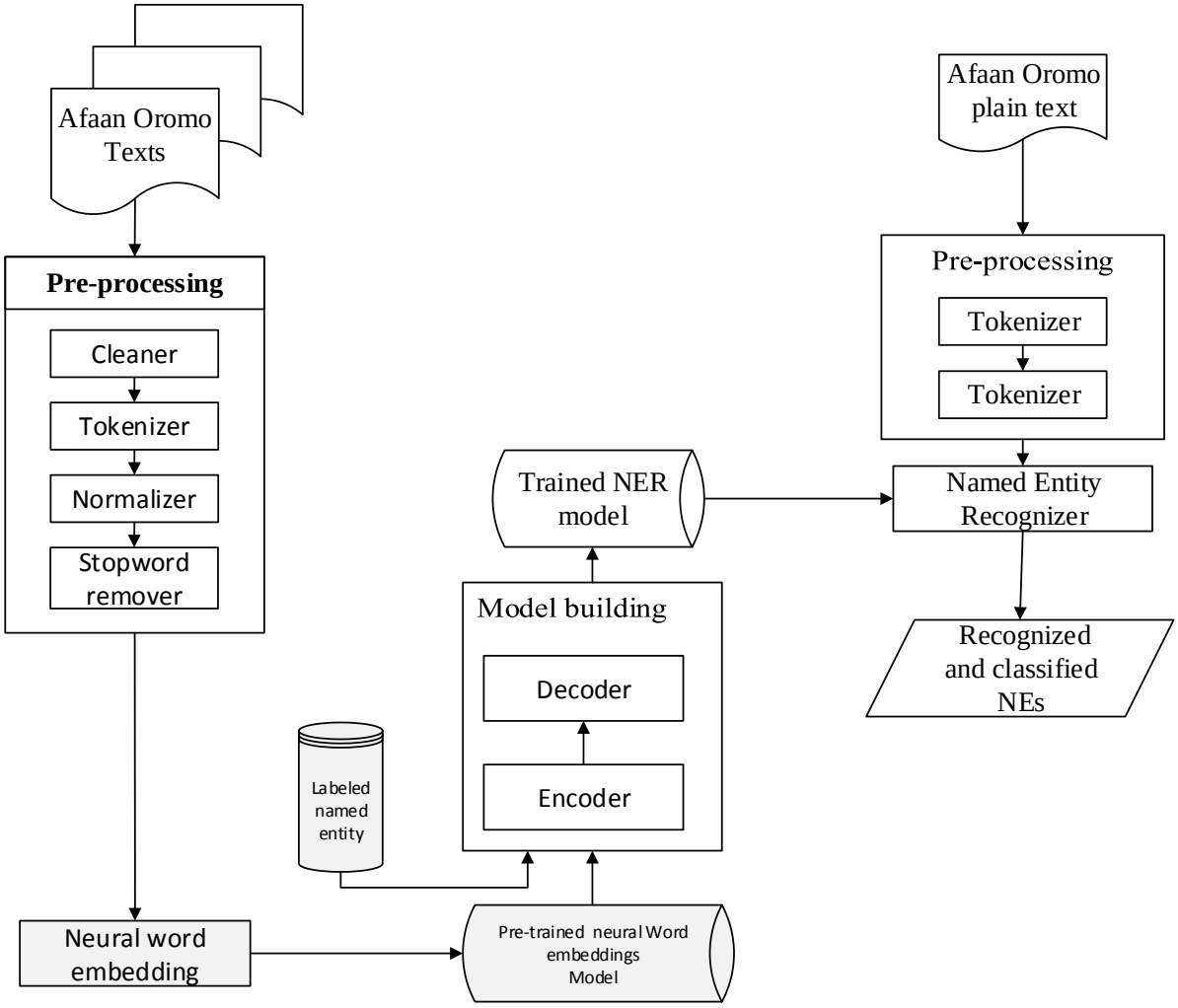


Figure 4.1: Architectural Design of Afaan Oromo NER

4.2.1 Preprocessing

Preprocessing involves preparing the dataset into a format that is desirable for further work. We preprocess Afaan Oromo texts to prepare labeled and unlabeled corpus for tagged named entities and word embeddings respectively that is desirable for training and evaluation of the model. Afaan Oromo texts, which are obtained from websites and document passed through the sub-stage of preprocessing that is data staging process namely cleaner, tokenizer, normalizer, and stopwords remover.

Cleaner

The text data, which is gathered from different websites, must be cleaned because the gathered data have different problems such as noisy, incomplete, inconsistent, dirty, and unformatted data for both named entities and neural word embeddings. In principle, we must

clean the data before training the word embedding, and preparing the tagged named entity. Cleaning involves preparing the corpus into a format that is desirable for tagging name entity and word embeddings. The garbage characters that are a part of the HTML syntax and delimiters like < and > are useless for our task. The garbage characters and delimiters don't provide value for our tagged name entity and word embeddings. So, we get rid of the garbage character and delimiters. Algorithm 4.1 shows the cleaning procedure for the given document.

Algorithm 4.1: Cleaning Procedure

```
Begin
Input: Afaan Oromo texts
For each Afaan Oromo texts in the corpus
Check for the presence of HTML characters OR a numeral
    If opening AND closing HTML characters present
        Remove
    End if
 End for
Output list of cleaned text
Stop
```

Tokenizer

To train models for both NER and neural word embeddings, require the input text data in a list of sentences, with each sentence being a list of words. Thus, we do tokenization to break the stream of character into raw term or token by detecting word boundaries of the written text. Afaan Oromo has its own punctuation marks that demarcate words and sentences. The tokenization part of this work determines words using the Afaan Oromo punctuation mark and white spaces. A word that has an apostrophe **(Hudhaa)** in Afaan Oromo is tokenized as they are. For example, Israa'el (Israel), har'a (today). Hyphenated words such as **mata-duree** are also tokenized as they are that means as a single word. In general, we differentiate the difference between words and tokens. We entail to the output of a tokenizer as a token and the meaning-bearing units as words. A token may be compiled of two words, several tokens can be regarded as one word, and sometimes different tokens denote the same underlying word.

Algorithm 4.2: Tokenizer algorithm

```
Begin
Input: cleaned text
detect punctuation symbol()
Split the cleaned text in to words()
if the word is Hyphenated
      add the word  to list()
Create array of a sentence()
Output
list of each sentence
list of words
Stop
```

Normalizer

While we were generating a neural word embeddings model for our study, it was important to lowercase the text. The reason to lower case the text data in our dataset is to effectively represent the learned features with the same weight for the same words. For instance, **(wallagga vs Wallagga)** is found in different variations of input capitalization that gives as different output in neural network representation learning weight for having the same name. Before generating word embeddings feature, which is a fed to the neural network, all the text has to be converted into lowercase. While lowercasing is generally helpful, we only apply for the neural word embeddings generation. We also used normalization to identify and replace a short from of Afaan Oromo texts to their corresponding long forms. Afaan Oromo texts which refer the same meaningful entity can be written in abbreviation, for example, Mootummaa Naannoo Oromiyaa, MNO, Kongirasii Federaalistii Oromoo, KFO etc.

Algorithm 4.3: Normalizer algorithm

```
Begin
Input: text and list of Afaan Oromo abbreviations (A)
Output: normalized word or phrase
   detect case of the word in A
   if the word is uppercase()
       lower case  the word ()
    end if
   For every word/phrase(C) in A
     if A == C
         replace A with C
     End if
   End for
   Stop
```

Stopword Remover

As stop words have little or no discriminating value for word representation purposes, they should be removed. For instance, in Afaan Oromo, words like “eega”, “eegasii” and “kun” provide very little value to word representation, and in many cases instead of a description, they add noise. Stopword remover removes stop words from segmented sentences by finding the match with list of stopwords stored in the NLTK stopwords of Afaan Oromo. Algorithm 4.1 shows how to remove the stopwords from the given document.

Algorithm 4.4: Stopwords Remover Procedure

```
Begin
Input: Afaan Oromo texts
For each sentence in each paragraph
     If a word in each sentence is stopwords then
       Remove the word from the sentence
      End if
 End for
Output: list of sentences without stopwords
Stop
```

4.2.2 Neural Word Embedding

Before preparing vector representation for sequence of each words, data preprocessing tasks such as cleaning, tokenization, normalization, and stopword removal have been performed. After that, we used word2vec model techniques namely the skip-gram with a negative subsampling model to generate word vector from preprocessed Afaan Oromo texts. To automatically generate a word embedding model, we first learn the embedding for Afaan Oromo word on a large unlabeled corpus in an unsupervised learning. Finally, we obtained Afaan Oromo neural word embeddings model, which used as pre-trained word embeddings as input features during final model building for Afaan Oromo NER. Because of Afaan Oromo is resource-scarce language and have no enough dataset for NER, we use pre-trained word embedding model to learn a generalized representation of words. Therefore, the main purpose of employing pre-trained embeddings for Afaan Oromo NER is to provide vector representation for a word that did not appear in the supervised training set. Thus, learning generalized representation of words is an effective way of handling data sparsity caused by high-dimensional lexical features in the NER. By using this neural word embeddings, we attempted to reduce the hand design feature engineering and data sparsity problem. To this end, the pre-trained word embedding model overpasses the shortages of the scarcity of labeled data to some extent. Finally, the pre-trained word embeddings are seeded to the encoder. To model the neural word embedding, we use an unlabeled corpus size of 50,284kb and 302,691 sizes of a sentence of Afaan Oromo.

Algorithm 4.5: Neural Word Embeddings Procedure

```
Begin
Input: document
Output: word embedding model
Read text document  into a list of strings
Build the dictionary and replace a rare word with unk token
Create a function to generate a training of a batch for the
skip-gram model
Build and train a skip-gram model
Begin training
Stop
```

4.2.3 Encoder

The encoder takes labeled named entity and word vector as input, then trains and extracts the local context representative features for each sequence of words. Since, we used BiLSTM algorithm as encoder, the input is processed in terms of two distinct hidden states to accomplish both the past and the future context information for the sequence of tokens. The two hidden states is concatenated and given to the decoder for further processing and finding the best tag sequence. Overall, encoding the context vector representative for the sequence of each words from the past as well as from the future is advantageous for named entity recognition. For instance, if we provide a document and need to retrieve named entities such as **Israa'el** (**Israel**), **Oromiyaa** (**Oromia**), and **Abdii Boruu** (**Abdi Boru**), as well as to categorize them into a predefined set of semantics such as location, organization, person, or other. The NER task is context dependent, as **Oromia** can be a location (the state) or a name of person, **Abdi Boru** can be a name of person or organization and **Israel** can be the name of a country or a person. This can be handled with a BiLSTM algorithm which processes information in the forward and backward direction for a given sequence of each tokens.

Before training a model for Afaan Oromo NER, the encoder component generate a character level representation from supervised dataset (annotated Afaan Oromo named entities**).** The encoder automatically generates character level representation during training instead of hand-engineering of prefix and suffix information about words. Thus, we obtain the final word embedding representation by concatenating forward and backward LSTM state to an embedding from lookup table. To this end, we assume the final representation of the BiLSTM contains two state: forward LSTM and backward LSTM that can accurately represent the suffix and prefix of a word respectively. Thus, we attempt to handle the drawback of word embedding that means Out-of-Vocabulary words and morphological information of words regularities using character embedding generated by BiLSTM before training of the model of NER.

4.2.4 Decoder

The decoder component takes the output of the encoder as input, then processes and finds the best label sequence which has the highest prediction score for the sequence of each tokens. This component is used to decode the text data and effectively model tagged named entity

features. Furthermore, the decoder component predict the best tags and verifies the validity of tagged named entity. For instance, in BIO encoding ‘O I-label’ tag sequence is invalid as the first label of one NER should start with B- not I- that means valid tag sequence can be ‘O B-label. Since we used CRF algorithm as decoder, the adjacent dependence between successive labels is learned and jointly decode the best chain of labels for a given sentence.

4.2.5 Named Entity Recognizer

The trained NER model supports named entity recognizer by supplying the necessary information to recognize and classify named entity from the testing data. By taking into account the tokenized Afaan Oromo plain text from user and the trained NER model generated from encoder and decoder, named entity recognizer predicts the class label of each sequence of tokens in sentence. To do this, named entity recognizer performance two main tasks: named entity detection and classification. Named entity detection is the first task in named entity recognition that finds the named entity from the tokenized Afaan Oromo plain text. Named entity classification is the second task in named entity recognition that classifies the detected named entity into predefined semantic classes. We use a CRF to predict a label from discrete set of classes for a sentences. The named entity recognizer select candidate named entity based on the calculated probability by invoking the trained model.

Chapter Five: Experimentation and Result

5.1 Introduction

In this chapter, we present the experimental aspects of our study. First, we explain the experimental procedures like experimental setting, corpus preparation, performance measure, and hyper-parameter settings**.** Secondly, we explain the development tools and programming language. Finally, we discuss the result of our experimental scenarios.

5.2 Experimental Settings

In this study, an experiment for the proposed model are performed on DELL Laptop with Intel core i5-60030CPU and 4 GB RAM running on Windows 8 Operating with Keras and Tensorflow backend support. To train BiLSTM-CRF network model with distributed representation as an input feature for Afaan Oromo NER, it takes half of the daytime. Likewise, training neural word embeddings also takes more time.

5.2.1 Data collection and Corpus Preparation

Currently, the development of NER more of using neural word embeddings because it requires little or no handcrafted features compared to non-deep learning. However, the deep neural networks approach relies on the tagged and untagged dataset. We apply different techniques to collect and prepare datasets for both annotated and unannotated for name entity and word embeddings respectively. Thus, the data for the tagged name types and word embeddings are collected from an unstructured source like website and magazines. The data collected from Kallacha Oromia is in the form of pdf format, a pdf miner tool is used for extracting the text. The Python libraries such as Requests and Beatifulsoup were used to extract text data from websites such as VOA, Afaan Oromo Fana Broadcasting, BBC Afaan Oromo, OBN, and OMN. We also downloaded a dumped file of omwiki-2016001 from Oromo wiki and then we extracted the text data. The collected corpora named **Orcorpus** scraped from the different website and article. To prepare annotated dataset, we followed BIO tagging scheme by following CoNLL's 2002 format and entity types is a person, organization, location, and miscellaneous. The tagging scheme is used to mark the word segmentation result manually for the tagged named entity types. Each word is assigned one of the subsequent tags, as seen in Table 5.1. We typically split our dataset into three sets: training, validation and test to model

our NER, which is good at predicting new text that learning algorithm did not see during model construction. We use the two holdout sets: validation data set to choose the selection of learning algorithm and guide the selection of the best values of hyper-parameters; and test data set to evaluate the performance of Afaan Oromo NER model. Our holdout data set is separated from the data used for training the NER model.

The topics of the texts for preparing both annotated name types and neural word embeddings are from all domain that means from politics, religion, sport and academic. The statistic for name types and word embeddings are depicted in Table 5.2 and Table 5.3 respectively.

Table 5.1: BIO Tags for NER

TAG	DESCRIPTION
O	Not part of a NE
B-PER **I-PER**	starting word of a person name continuation of a person name
B-LOC **I-LOC**	starting word of a location name continuation of a location name
B-ORG **I-ORG**	starting word of an organization name continuation of an organization name
B-MISC **I-MISC**	starting word of another kind of named entity continuation of another kind of named entity

Table 5. 2: Statistics of Afaan Oromo Named Entity

Sno	Tag name	Training set	Validation set	Test set
1	Person	2879	681	933
2	Organization	2075	485	626
3	Location	1875	473	618
4	Miscellaneous	1709	269	340
		Total NEs =8,538	Total NEs =1,908	Total NEs=2,517
		Total = 12,963		

As shown in Table 5.2, we prepared totally 12,963 named entities from these 8,538, 1,908 and 2517 are used for training, validation and test set respectively. Our corpus contains four named entities which are classified as person, organization, location and miscellaneous. These groups of named entities are manually annotated and evaluated by the domain expertise.

Table 5.3: Statistics of Orcorpus for Word Embedding

Dataset	Data size	Sentence len	Raw words	Effectively trained
Orcorpus	50,284KB	302,691	2,492,547,500	2,047,147,048

As shown in Table 5.3, the first column shows the name of our corpus which is named as Orcorpus. The second column describes the size of our data set, third column depicts the sentence length in our corpus. The fourth column shows the raw words in our Orcorpus and the fifth column shows effectively trained word vector.

5.2.2 Performance measure

For the sake of NER task, the performance of the model can be valued by calculating correct entities identified by the system and total named entities present in the dataset. In this study, we evaluate the performance of the model using test data set. The performance of the model is evaluated by standard measurement metrics such as F-score, recall, and precision.

The recall (R) is the number of correctly identified (and classified) named entities, divided by the number of correct (gold) named entities:

$$R = \frac{|gold \cap retrieved|}{|gold|} \quad (16)$$

Precision (P) is the number of correctly identified (and classified, if required by the task) named entities, averaged with the number of named entities retrieved by the system. All tokens of the named entity must be identified and correctly classified as marked in gold data (that is, in data annotated by human annotators) can be calculated as.

$$P = \frac{|gold \cap retrieved|}{|retrieved|} \quad (17)$$

F-measure, (or sometimes also F-score, with or without hyphen) is the harmonic mean of precision and recall, computed as:

$$F = \frac{2(P * R)}{P + R} \quad (18)$$

Accuracy is the ratio of correctly identified (and classified) named entities to the total observations, Computed as:

$$acc = \frac{\dot{gold}}{total\ tagged\ dataset} \quad (19)$$

5.2.3 Hyper-parameter settings

We use a validation dataset on BiLSTM-CRF to optimize hyper-parameters to get better results. We implement with different values of batch size, number of units in each layer, optimizers, dropout rate and activation functions. We train our models using the training data and optimizing hyper-parameters on validation data. To train our parameter, we mainly based on the back-propagation algorithm which updates parameters on every training example, throughout the time. For parameter optimization and the loss function minimization, we employ Stochastic Gradient Descent (SGD) and Adam algorithm with a learning rate of 0.001 and a gradient clipping of 5.0. Numerous approaches have been proposed to improve the performance of SGD, such as Adadelta [78], or Adam [79]. Although we observe faster convergence using these methods, none of them perform as well as SGD and Adam with gradient clipping. To generalize our model and avoid overfitting, we add a dropout layer with dropout probability of 0.5 just after the embedding layer. The hyper-parameter used to train our model is summarized in Table 5.4.

Table 5.4: Hyper-Parameter Setting for the Proposed Model

Hyper-parameter description	Short name	Range	Final Value
Word embedding dimension	- dim_word	[50,250]	200
Character embedding dimension	- dim_char	[10,30]	30
Number of iteration	- nepochs	[50-100]	100
Character LSTM hidden vector size	- char-lstm-dim	[10,30]	30
Word LSTM hidden vector size	- word-lstm-dim	[50,250]	200
Dropout probability	- Dropout rate	[0.2,0.7]	0.5
Optimization algorithm	- lr_method	-	SGD,Adam
Learning rate	- lr	[0.001,0.1]	0.001
Learning rate decay	- lr_decay	[0.001,1.0]	1.0
A set of number of observations of training data	- Batch size	[30-100]	100
Gradient clipping	- Clip	[1-5]	5

The first column shows the hyper-parameter description to optimize, whereas the second column describe the short name of the hyper-parameter used in the experiment. The third column depicts the interval and fourth column shows the parameter values to obtain the highest recognition score on the validation dataset.

Table 5.5: Parameter Setting of Neural Word Embeddings

Hyper-parameter description	Short name	Values
A set of number of observations of training data	batch_size	128
Dimension of the embedding vector	embedding_size	128
How many words to consider left and right	skip_window	1
How many times to reuse an input to generate a label	num_skips	2
Random set of words to evaluate similarity on	valide_size	16
Only pick dev samples in the head of the distribution	valid_windows	100
Number of negative examples to sample	num_sample	64
Number of iteration	num_steps	5000000
Optimization algorithm	optimizer	SGD

As Table 5.5 shows, the first column shows the hyper-parameter description to optimize our model for word embeddings, whereas the second column describe the short name of the hyper-parameter used in our experiment of neural word embeddings, third column depicts the parameter values.

5.3 Development tools and programming language

In this study, we used many development tools such as anaconda framework and Deep learning frameworks namely, TensorFlow.

Anaconda

We prioritized the Anaconda framework because it is a free and open-source distribution of the Python and other programming languages for scientific and computing that aims to simplify package management and development. This framework equips to work with many open-source packages and libraries. Using the conda command, we can install Python library and many other packages. In addition to this, we easily manage multiple data environments that can be maintained and run separately without interference from each other.

Python 3.7 programming language installed with anaconda is used to experiment Afaan Oromo NER. Python libraries such as Requests and Beautiful Soup was used to analyze the structure of the website, extract texts, and combine to a single text file. We chose to work with Python because of the rich community and library infrastructure. As shown in Table 5.6, the listed python libraries infrastructures have been used for our implementation:

Table 5. 6: Python Library Used for the Study

Library package	Version	Description
Numpy	1.16.2	Used to process multi- dimensional arrays and matrices
Matplotlib	3.0.3	used to create static, animated and interactive visualization
Nltk	3.4	Used to analysis and process the text data. for example, tokenization ,and stopwords removal
Spyder	3.7	Used to edit, analysis and debugging the code for NER
Juputer notebook	5.7.8	Used to edit, analysis and debugging the code
Word2vec	-	Used to efficiently learning for distributed word representations
PCA and t-SNE	-	for dimension reduction and word embedding visualization

Tensorflow

TensorFlow 1.13.1 installed using conda is used to conduct this research. This framework is designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. We prioritized TensorFlow framework because of a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors; it contains programming support of deep neural networks and machine learning techniques; a high scalable feature of computation with various data sets and it also includes a unique feature of optimization of same memory and the data used. We used these frameworks to train and run a deep neural network to build a model for NER and neural word embeddings.

5.4 Result

5.4.1 Effect of Word embeddings

Automatically generated vector representations of the word, can be used as a feature for NER. As the result obtained from the experiment shows that word embeddings features can significantly improve Afaan Oromo NER, as seen in Table 5.7. These results also show that, employing large unlabeled Afaan Oromo text to generate features and train a model for NER with a small amount of labeled significantly improved Afaan Oromo NER**.** As result shows in Table 5.7, BiLSTM-CRF model that uses pre-trained word embeddings as a feature yields high performance as compared to BiLSTM-CRF model with character embeddings.

5.4.2 Effect of Character embedding

We also incorporated character-based word representation learned from BiLSTM in our model to handle the drawback of word embeddings and captures some information of morpheme-label regularity. We also examine the effect of character embedding for Afaan Oromo NER by removing word embedding. As seen in Table 5.7, BiLSTM-CRF model with character embeddings yields the lower performance as compared to BiLSTM-CRF with word embeddings**.**

5.4.3 Effect of character and word Embedding

Instead of only considering word-level or character level representations of words as to the basic input, we incorporated character and word vector representation in our model. The combination of BiLSTM and CRF with pre-trained word and character embeddings yields the highest performance with respect to precision, recall and F-score as compared to BiLSTM-CRF with character embedding or pre-trained word embeddings, as shown in Table 5.7. However, enhancements in terms of performance are not adequate.

5.4.4 Effect of dropout

As observed from the Table 5.7, the early experimentations showed that character-level embeddings did not adequately outperformance our overall performance while employed in combination with pre-trained neural word embeddings. To improve the generalization and avoid overfitting of the model, we employed a dropout rate after the embedding layer just before feed to our encoder model. The BiLSTM-CRF model which use pre-trained word and character embeddings as input feature with dropout rate surpasses the BiLSTM-CRF model

that use only char or word embedding or char and word embedding without dropout rate, as seen in Table 5.7.

Table 5.7: Result of Afaan Oromo NER Using Different Feature Configurations

Model	Input representation	Performance			
		Recall	Precision	F1-Measure	Accuracy
BiLSTM	Char+ Pre-trained+dropout	82	84	82.93	96.40
BiLSTM-CRF	Pre-trained	84	85	84.62	96.77
BiLSTM-CRF	Char	83	84	83.37	96.44
BiLSTM-CRF	Char+Pre-trained	87	88.19	87.79	97.49
BiLSTM-CRF	Char+Pre-trained +dropout	93	94	93.26	98.87

In Table 5.7, “Pre-trained” means a model that incorporates pre-trained word embeddings, whereas “Char” means a model that contains character level representation of words, “dropout” states a model that comprises a regularization the dropout rate.

As shown in Table 5.7, the results achieved from the BiLSTM with character level representation and word embeddings and dropout rate are much lower compared to BiLSTM-CRF with different input feature configuration. BiLSTM model predict correct tag only by considering the context of word. Accordingly, the main errors of this model are the misunderstanding with non-named entities and named entities. Adding CRF as tag decoder on the top of BiLSTM into the system improves the results however, enhancements in terms of performance are not adequate. As experimental result shows in Table 5.7, the BiLSTM-CRF model which uses pre-trained word and character embeddings as input feature with dropout rate surpasses the BLSTM-CRF model that use only char or word embedding or char and word embedding without dropout rate.

5.5 Comparison with the previous work

We compared the result of our work with previous works done by Mandefro and Abdi. As shown in Table 5.8, we found that our precision, recall and F-score suppress the performances of the previous works. Using pre-trained word embeddings (dense representation of words), and character label representation gave the best performance for our system than the sparse representation of words (one-hot).

Table 5. 8: Comparison of Afaan Oromo NER with the Previous Work

Work	Model	**P**	**R**	**F**	**Acc**
Mandefro [20]	CRF	77.84	75.59	76.70	-
Abdi Sani [21]	CRF, Rule	84.12	81.21	82.52	-
Our work	**BiLSTM-CRF**	**94**	**93**	**93.26**	**98.87**

5.6 Discussion

To conduct experiment Afaan Oromo NER, we conducted three main experimental.

The first experiment is collecting and preparing a dataset for Afaan Oromo NER. We collected Afaan Oromo texts from a new domain and article by using the Python library. After we had collected the dataset, we manually prepared annotated a corpus size of 768KB and unannotated corpus size of 50,284KB.

The second experiment is automatically generating neural word embeddings from a large an unannotated corpus size of 50,284KB using unsupervised learning. The neural word embeddings for Afaan Oromo texts is prepared by using word2vec model. Specifically, we used Skip-gram model with negative sampling. This developed feature has an ability to capture the semantic and syntactic features of the word relation. During experimentation, neural word embeddings did not capture the semantic and syntactic word relation for a limited amount of training data set. However, after we increased our untagged dataset into a million of corpus size, word embedding captured the syntactic and semantic word relation for Afaan Oromo words. To such a degree, we concluded that the word2vec model able to capture the semantic and syntactic of word relation for huge amount text data size (see the screenshot taken while developing a model from Appendix C). The generated word embeddings is used as a pre-trained for Afaan Oromo NER.

The third experiment is conducting an experiment of Afaan Oromo NER by: (1) using BiLSTM model that use pre-trained word and character level representation as input feature with dropout rate is not sufficiently surpasses all subsequent models; (2) adding CRF as tag decoder on the top of BiLSTM that used only word embeddings as feature is also not significantly improved the performance of our models; because word embeddings features possess it to own limitation; for example embedding for different words is independent; an

out-of-vocabulary word cannot be handled; (3) Combining BiLSTM with CRF and using only character embeddings as future is also not adequately surpasses overall performance of subsequent models; (4) concatenating character embedding with pre-train word embeddings features and seed to the BiLSTM-CRF model also did not adequately improve the performance of our model; (5) employing a dropout rate after the embedding layer just before feed to BiLSTM-CRF model significantly improved our performance of Afaan Oromo NER. In this possible manner, using BiLST-CRF model with pre-trained word and character level representation and adding dropout rate after the embedding significantly improved Afaan Oromo NER with achieved F-score 93.26. Thus, as the demonstration shows, deep neural network-based NER system that uses a dense vector representation typically achieves excellent results in Afaan Oromo NER without manually designed features like gazetteers, word shape, POS tag, position, token, prefixes and suffix as unlike the work of by Mandafro Legesse and Abdi Sani.

The result shows the combination of BiLSTM with CRF and using neural word embeddings with character embeddings is a practicable solution for resource scarce languages like Afaan Oromo. The results also showed that automatically learned vector representation of a word that means word and character embeddings features can substitute manually designed features for Afaan Oromo NER. Furthermore, these features have merely delivered better performance while radically reducing the effort in manual feature design. In conclusion, our neural network (BiLSTM) with traditional classifier (CRF) surpasses the current system of Afaan Oromo NER.

Chapter Six: Conclusion and Future Works

6.1 Conclusion

Named entity recognition is a crucial task which is often implemented while processing natural languages. It is used to extract the proper nouns from text and classify them into predefined semantic categories such as person, location, organization, date and time. In this study, deep neural network-based named entity recognition model is proposed for Afaan Oromo. The proposed model uses distributed representation of words as an input feature. It is the combination of bidirectional LSTM as context encoder and CRF as tag decoder which identifies and classifies named entities from Afaan Oromo texts. Due to the use of context encoder, the proposed model has the ability to handle the word-order and morphologic word regularity. Experimental result shows that applying the combination of Bi-LSTM as context encoder with CRF as tag decoder model achieved F-score value 93.26 on the Orcorpus. The proposed model achieves better results while comparing with other existing system developed for Afaan Oromo named entity recognition.

6.2 Contribution of This Work

In this work, we primarily point out the problem of the previous works of Afaan Oromo NER models: they are just as supervised machine learning which means “memorizing “names seen in train data. To better learn “what is the name “, we analyze in the capability of a word embedings and character-level representation of words in the task of recognizing name tokens from non-name tokens. We demonstrate that using pre-trained word and character level representation of the word as an input feature provides a simple and powerful model for capturing these differed, identifying named entity tokes and boost the performance of Afaan Oromo NER. Moreover, the amalgamating learning algorithm like BiLSTM-CRF with concatenation of pre-trained word embeddings with character level representation and with regularization techniques (dropout) significantly improved the performance of an Afaan Oromo NER.

Below, we summaries the contribution of our works:

- This study has been one of the first attempts to thoroughly examine the combination of Bi-LSTM and CRF models that use the concatenation of pre-

trained word embedding and character embedding with regularization techniques (dropout) for Afaan Oromo NER.

- We prepared annotated dataset for Afaan Oromo NER

6.3 Future works

As our work demonstrated that automatically generated neural word embedding and character level representation of have able to use as an input feature for Afaan Oromo NER. The experimental result demonstration that the high performance of the Afaan NER model can be built from the BiLSTM-CRF that uses character level representation and pre-trained word embedding as an input feature with regularization (dropout rate). To make Afaan Oromo NER a full-fledged system, a further research needed to be conducted. Based on our observations, we recommend the following future research areas:

- We applied word2vec model for distributed representation of words, further studies need to be carried out in order to validate the effectiveness of distributed representation of words using different neural embedding model;
- We used BiLSTM to generate character level representation, further studies need to be carried out to generate character level representation by using other deep neural network;
- We utilized BiLSTM-CRF to model Afaan Oromo NER, future studies should explore the effects of other deep neural network with different input features for the improvement of performance of the system
- Our training and the two holdout datasets are small, further works need to enhance this dataset for better performance of the system.

References

[1] D. Martin and J. Jurafsky, Speech and Language Processing: An introduction to natural language processing,computational linguistics, and speech recognition, June 25, 2007.

[2] R. Sundheim and G. Beth, "Message Understanding Conference-6:A Brief History," in *In Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, Copenhagen,Denmark, 1996.

[3] G. Omid, "Unsupervised Biomedical Named Entity and Recognition," UWM Digital Commons,Teses and Dissertations,The University of Wisconsin-Milwaukee, August 2017.

[4] Changki Lee , Yi-Gyu Hwang, Myung-Gil Jang, "Fine-grained named entity recognition," in *and relation extraction for question answering In Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Develop*, 2007.

[5] John Lafferty,Andrew McCallum,Fernando C.N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," June 2001.

[6] C. N and S.-T. J, "An introduction to support vector machines and other kernel-based learning methods," in *Cambridge University Press*, US, 2000.

[7] Ian Goodfellow,Yoshua Bengio,Aaron Courville, Deep Learning, 2018.

[8] Brian Y Tsui , Shamim Mollah , Dylan Skola , Michelle Dow , and Chun-Nan Hsu , "Creating a scalable deep learning based Named Entity Recognition Model for biomedical textual data by repurposingBioSample free-text annotations," *Data and text mining*, Sep. 12, 2018.

[9] Andrej Zukov-Gregori , Yoram Bachrach, and Sam Coope, "Named Entity Recognition With Parallel Recurrent Neural Networks," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 69–74*, Melbourne, Australia, July 15 - 20, 2018.

[10] S. C´ıcero dos and G. Victor, "Boosting Named Entity Recognition with Neural Character Embeddings," in *Association for Computational Linguistics*, Beijing, China, July 26-31, 2015.

[11] J. Hajin and J. Straková, "Neural Network Based Named Entity Recognition," Doctoral thesis,Institute of Formal and Applied Linguistics, Department of Computer Science,Charles University, 2017.

[12] T. Le, M. Y. Arkhipov and M. S. Burtsev, "Application of a Hybrid Bi-LSTM-CRF model to the task of Russian Named Entity Recognition," *Communications in Computer and Information Science,* vol. 789, pp. 91-103, 2018.

[13] Shotaro Misawa, Motoki Taniguchi, Yasuhide Miura and Tomoko Ohkuma, "Character-based Bidirectional LSTM-CRF with words and characters for Japanese Named Entity Recognition," in *Proceedings ofthe First Workshop on Subword and Character Level Models in NLP,pages 97–102,*, Copenhagen, Denmark, September 2017.

[14] R. Kuksa, J. Collobert, L. Weston, M. Bottou, K. Karlen and P. Kavukcuoglu, "Natural language processing (almost) from scratch.," *Journal of Machine Learning Research,,* 2011.

[15] E. N. Jason P.C. Chiu, "Named Entity Recognition with Bidirectional LSTM-CNNs," arXiv:1511.08308v5 [cs.CL], 19 Jul 2016.

[16] Gregor Wiedemann, Raghav Jindal, Chris Biemann, "microNER: A Micro-Service for German Named Entity Recognition," *arXiv:1811.02902v1 [cs.CL],* 7 Nov 2018.

[17] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami and C. Dyer, "Neural architectures for named entity recognition.," in *In Proceedings of NAACL-HLT 2016*, 2016.

[18] Pierpaolo Basile, Giovanni Semeraro, Pierluigi Cassotti, "Bi-directional LSTM-CNNs-CRF for Italian Sequence Labeling," in *CLiC-it 2017*, Rome, December 2017.

[19] X. Jingjing, H. Hangfeng, S. Xu, R. Xuancheng and L. Sujian, "Cross-Domain and Semi-Supervised Named Entity Recognition in Chinese Social Media: A Unified Model," *IEEE,* 2017.

[20] Mandefro Legesse , "Named Entity Recognition for Afan Oromo," Unpublished Masters Thesis,Department of Computer Science,Addis Ababa University, Adis Ababa, 2010.

[21] Abdi Sani Genemo , "AFAAN OROMO NAMED ENTITY RECOGNITION USING HYBRID APPROACH," Unpublished Masters Thesis,Department of Computer Science,Addis Ababa University, Addis Ababa, 2015.

[22] Besufekad Alemu, "A named entity recognition for amharic," Unpublished Master Thesis,Departemnt of Computer Science ,Addis Ababa University,Addis Ababa, AAU, 2013.

[23] S. L. Dagimawi Demissie, "Amharic Named Entity Recognition Using Neural Word Embedding as a Feature," Unpublished Masters Thesis,School of Electrical and Computer Engineering,Addis Ababa University, Addis Ababa, 2017.

[24] Mikiyas Tadele, "Amharic Named Entity Recognition Using a Hybrid approach," Unpublished Masters Thesis,Department of Information Science ,Addis Ababa University, Addis Ababa, 2014.

[25] J. Guo, W. Che, H. Wang and T. Liu, "Revisiting Embedding Features for Simple Semi-supervised Learning," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Harbin Institute of Technology, Beijing,China, October 25-29, 2014.

[26] S. Tjong Kim, E. F. and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *In Proceedings of HLT-NAACL*, 2003.

[27] Y. Ren and F. Liu, "Japanese named entity recognition for question answering system," in *In 2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, 2011.

[28] M. Oudah and K. F. Shaalan, "A pipeline arabic named entity recognition using a hybrid approach," 2012.

[29] G. Mohan, "Towards Data Science," A Review of Named Entity Recognition (NER) Using Automatic Summarization of Resumes, 2018-07-09T20:58:31.125Z. [Online]. Available: https://towardsdatascience.com/a-review-of-named-entity-recognition-ner-using-automatic-summarization-of-resumes-5248a75de175. [Accessed 27 10 2018].

[30] C. Tru H, T. Thao M and C. Cuong K, Text Clustering with Named Entities: A Model,Experimentation and Realization, Verlag Berlin Heidelberg: Springer, 2012.

[31] K. Sanjana and R. Wagh, "Named Entity Recognition Approaches and Challenges," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. vol. 6, no. 2, February 2017.

[32] D. Bikel, R. S. S Miller and R. Weischedel, "Recent Nymble: a High-Performance Learning Name-finder," in *Proceedings of the fifth conference on Applied natural language processing*, 1997.

[33] K. Z, F. O, M. A, M. R, S. A and G. J, "Combining data-driven systems for improving named entity recognition," *Data & Knowledge Engineering,*, 2007.

[34] J. Lafferty, A. McCallum and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[35] Moges Ahmed , "NAMED ENTITY RECOGNITION FOR AMHARIC LANGUAGE," Unpublished Masters Thesis,Department of Computer Science,Addis Ababa University, Addis Ababa, November, 2010.

[36] Bernd Heine and Derek Nurse (eds.), African Languages: An Introduction, Cambridge, United Kingdom: Cambridge University Press, 2000.

[37] G. Gene, "Oromo Dictionary," *Committee on Northeast African Studies,*, vol. Monograph No. 12, 1982.

[38] Tullu Guya, CaasLuga Afaan Oromoo Jildii-1, Finfnnee: Gumii Qormaata Afaan Oromootiin, Komishinii Aadaaf Turizimii Oromiyaa, 2003, pp. 105-220.

[39] G. Rabbirraa, SEERLUGA AFAAN OROMOO, Addis Ababa: Kuraz Intenational Publishing Eterprise, 2015.

[40] G. Yoav, Neural Network Methods in Natural Language Processing, U. o. T. Graeme Hirst, Ed., Toronto: Morgan and Claypool, 2017.

[41] F. Cl´audia, S. Diana, M. Cristina, G. Hugo and O. alo, "Relation detection between named entities: report of a shared task," in *Proceedings of the NAACL HLT Workshop on Semantic Evaluations: Recent Achievements and Future Directions,Association for Computational Linguistics*, Boulder, Colorado, June 2009.

[42] L. Ni, W. William and C. Cohen, "Relational Retrieval Using a Combination of Path-Constrained Random Walks," *Machine learning,*, p. 53–67, 2010.

[43] B. Matteo and D. Leon, "Recognising and Interpreting Named Temporal Expressions," in *Proceedings ofRecent Advances in Natural Language Processing*, Hissar, Bulgaria, September 2013.

[44] Z. FADI and M. JAD, "Arabic Temporal Entity Extraction using Morphological Analysis," *IJCLA*, vol. VOL. 3, p. 121–136, JAN-JUN 2012.

[45] V. Yadav and S. Bethard, "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models," in *Proceedings of the 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA;, August 20-26, 2018.

[46] D. Molla, M. van Zaanen and D. Smith, "Named Entity Recognition for Question Answering," in *Proceedings of the 2006 Australasian Language Technology Workshop (ALTW2006)*, Sydney.

[47] U. Arata, T. Akihiro, N. Takashi, T. Hiroya and O. Manabu, "Neural Machine Translation Incorporating Named Entity," in *Proceedings ofthe 27th International Conference on Computational Linguistics*, Santa Fe, New Mexico, USA, August 20-26, 2018.

[48] Bogda Babych, Anthony Hartley, "Improving Machine Translation Quality with Automatic Named Entity Recognition," in *In Proceedings of the 7th International EAMT workshop on MT and other Language Technology Tools Improving MT through other Language Technology Tools*, UK, 2013.

[49] H. Graeme, H. Eduard and J. Mark, Multi-source, Multilingual Information Extraction and Summarization,Theory and Applications of Natural Language Pro- cessing, P. Thierry, S. Horacio, P. Jakub and Y. Roman, Eds., New York Dordrecht London: Springer Berlin Heidelberg, 2013, p. 229–252.

[50] N. Chikashi, S. Satoshi, I. Hitoshi and G. Ralph, "Summarization System Integrated with Named Entity Tagging and IE pattern Discovery," 2002.

[51] M. Diana, B. Kalina and A. Isabelle, Natural Language Processing for the Semantic Web, Morgan and Claypool, 2017.

[52] T. M. Mitchell, Machine learning, New York, NY, USA: McGraw Hill, 1997.

[53] C. Adam, "DEMYSTIFYING UNSUPERVISED FEATURE LEARNING," Doctor Thesis,Department of Computer Science,Stanford University, September 2012.

[54] S. Anders, Semi-Supervised Learning and Domain Adaptation in Natural Language Processing, U. o. T. Graeme Hirst, Ed., Toronto: Morgan and Claypool, 2013.

[55] D. Hakan and O. Arzucan, "SEMI-SUPERVISED LEARNING BASED NAMEDENTITY RECOGNITION FOR MORPHOLOGICALLY RICH LANGUAGES," Master thesis,Graduate Program in Computer Engineering,Bo˘gazi¸ci University, 11/06/2014.

[56] Terry Koo, Xavier Carreras , Michael Collins, "Simple Semi-supervised Dependency Parsing," in *Proceedings of ACL-08: HLT pages 595–603, Association for Computational Linguistics*, Columbus, Ohio, USA, June 2008.

[57] L. Zhenghua, Z. Min and C. Wenliang, "Ambiguity-aware Ensemble Training for Semi-supervised," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 457–467*, Baltimore, Maryland, USA, , June 23-25 2014.

[58] K. Mahnoosh, D. V. Lance, S. Laurianne, Z. Guido and N. and Anthony, "The Benefits of Word Embeddings Features for Active Learning in Clinical Information Extraction," in *In Proceedings of Australasian Language Technology Association Workshop*, 2016.

[59] D. Ngan and N. Kim Anh, "Attentive Neural Network for Named Entity Recognition in Vietnamese," *FPT Technology Research Institute*, 31 Oct 2018.

[60] Y. Ikuya, S. Hiroyuki, T. Hideaki and T. Yoshiyasu, "Joint learning of the embedding of words and entities for named entity disambiguation," 2016. [Online]. Available: http://arxiv.org/abs/1601.01343. [Accessed 27 10 2018].

[61] M. Tomas, C. Kai, C. Greg and D. Jeffrey, "Efficient Estimation of Word Representations in Vector Space," *arXiv:1301.3781v3 [cs.CL]*, 7 Sep 2013.

[62] Z. Aston, L. Zachary C, L. Mu and S. Alexander J, Dive into Deep Learning, Mar 15, 2019.

[63] Richa Sharma, Sudha Morwal, Basant Agarwal, Ramesh Chandra,Mohammad S. Khan, "A deep neural network-based model for named entity recognition for Hindi language," *Neural Computing and Applications*, 4 April 2020.

[64] D. Nadeau and S. Sekine, "A survey of named entity recognition and classification," *International Journal of Linguistics and Language Resources*, vol. 30, no. 1, pp. 3-26, 20.

[65] Tomas Mikolov,Ilya Sutskever,Kai Chen,Greg Corrado,Jeffrey Dean, "Distributed Representations ofWords and Phrases and their Compositionality," *arXiv:1310.4546v1 [cs.CL]*, 16 Oct 2013.

[66] Z. Ping, T. Qingping, Z. Haoyu, M. Xiankai, Z. Zhuo and X. Jianjun, "Character Feature Learning for Named Entity Recognition," p. 1811, JULY 2018.

[67] M. Xuezhe and H. Eduard, "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Berlin, Germany, August 7-12, 2016.

[68] L. Marc Moreno and K. Jugal, "Deep Learning applied to NLP," *arXiv:1703.03091v1 [cs.CL]*, 9 Mar 2017.

[69] Tomas Mikolov, Martin Karafiat, Lukas Burget Jan, Honza and Sanjeev Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH 2010*, Brno University of Technology, Czech Republic;Department of Electrical and Computer Engineering, Johns Hopkins University, USA, 2010.

[70] A. Afshine and A. Shervine, "CS 230 - Recurrent Neural Networks Cheatsheet," stanford university , 2019. [Online]. Available: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks. [Accessed 04 04 2019].

[71] H. Sepp and S. Jurgen, "LONG SHORT-TERM MEMORY," in *Neural Computation*, Munchen, Germany, 1997.

[72] T. Young, D. Hazarika, S. Poria and E. Cambria, "Recent Trends in Deep Learning Based Natural Language Processing," *EEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55-75, 2018.

[73] M. Schuster and K. K. Paliwal, "Bidirectional Recurrent Neural Networks," *TRANSACTIONS ON SIGNAL PROCESSING*, vol. 45, pp. 2673-2681, 11, NOVEMBER 1997.

[74] Zhiheng,Huang; Wei, Xu; Kai, Yu;, "Bidirectional LSTM-CRF Models for Sequence Tagging," *arXiv:1508.01991v1 [cs.CL] 9* , Aug 2015.

[75] O. Gungor, E. Yıldız, S. Uskudarlı and T. Gungor, "Morphological Embeddings for Named Entity Recognition in Morphologically Rich Languages," in *arXiv:1706.00506v1 [cs.CL]*, Jun 2017.

[76] P. Wang, Y. Qian, F. K. Soong, L. He and H. Zhao, "A Unified Tagging Solution: Bidirectional LSTM Recurrent Neural Network with Word Embedding," 2015.

[77] P. Thai-Hoang and L.-H. Phuong, "End-to-end Recurrent Neural Network Models for Vietnamese Named Entity Recognition: Word-level vs. Character-level," *arXiv:1705.04044v3 [cs.CL]*, 21 Jul 2017.

[78] M. D. Zeiler, " Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[79] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[80] H. M. a. S, "Design Science in Information Systems Research," *Management Information Systems Quarterly*, pp. 75-105, 2004.

[81] J. Straková, M. Straka and J. Hajič, "Neural Networks for Featureless Named Entity Recognition in Czech," in *Springer International Publishing*, Brno , Czech Republic, 2016.

[82] T. H. Cao, V. M. Ngo, D. T. Hong and T. T. and Quan, "Semantic Document Clustering on Named Entity Features," Faculty of Computer Science and Engineering, Ho Chi Minh, Vietnam.

[83] S. Sarawagi, "Information Extraction," *Foundations and Trends in Databases*, vol. Vol. 1(3), pp. 261-377, 2007.

[84] K. Michal and K. Miloslav, "Named Entity Recognition," Doctoral Thesis,Department of Computer Science and Engineering,University of West Bohemia, October 9, 2015.

[85] A. Das, D. Ganguly and U. Garain, "Named Entity Recognition with Word Embeddings and Wikipedia Categories for a Low-Resource Language," *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 16, no. 3, pp. 1-19, 2017.

[86] M. Bulcha, "Language, Ethnic Identity, and Nationalism in Ethiopia, The Oromo Commentary," pp. 5-18, 1993.

[87] Y. Zhang and a. J. Yang, "Chinese NER Using Lattice LSTM Yue," *arXiv:1805.02023v4 [cs.CL]*, 5 Jul 2018.

Appendix A

Sample Annotated Dataset

Ispeen B-LOC
ragaan O
dhiheenya O
kana O
Dhaabbata B-ORG
Fayyaa I-ORG
Addunyaarraa I-ORG
argame O
mirkaneessetti O
, O
jijjiiramni O
haala O
qilleensa O
dhibee O
fidaa O
jira O
. O
Akka O
magaalaa O
Siiviillaatti B-LOC
yeroo O
27ffaa O
kan O
qophaa'e O
dorgommii O
qaxxaamuura O
biyyaa O
kiilomeetira O
10.8tti O
atileenni O
Ugaandaa B-LOC
beekamaan O
Moosees B-PER
Kipikiroo I-PER
fi O
Taarikuu B-PER
Baqqalaa I-PER
tokkoffaa O
fi O
lamaffaa O
ta'nii O
mo'ataniiru O
. O

Appendix B

Sample word Vector feature

jijjirraan 0.10534852 0.35259408 0.40470353 -0.14484258 -0.65202725 -0.05509495 0.69524723 0.37600115 0.19823563 -0.16790134 0.21679589 -0.19473898 0.3839752 0.31709513 0.3337952 -0.016725358 -0.007182841 -0.01458523 0.0055932417 -0.028646847 0.59075236 -0.20871055 -0.32403818 0.09262035 -0.24605173 -0.22886424 0.1643355 0.18038048 -0.21234202 -0.31912768 0.20924999 0.11826073 0.01229804 0.062013224 -0.05817279 0.1792012 -0.28106096 -0.09584165 -0.009604835 0.070910506 -0.13036613 0.50816005 -0.0026158763 0.0055251624 0.2821714 0.10677329 0.10787608 0.27301148 -0.064949006 0.30669808 -0.046958953 0.20743291 -0.065330505 -0.19744833 -0.099958465 -0.7036997 0.20003131 0.43741122 0.11746727 0.10775665 0.09972246 -0.034934677 -0.075541176 0.12585603 -0.1290511 0.32942033 0.114333756 0.038972113 -0.054285627 -0.41352183 -0.2085138 0.27872476 -0.17734113 0.38891092 -0.015413372 -0.30633065 -0.07911143 -0.30322996 0.19566138 0.41906115 -0.0052055973 -0.46497265 0.15236029 -0.017354585 0.38735133 -0.018112637 -0.3294063 0.099531464 -0.038807623 0.02604827 0.034926392 -0.37569556 0.39262435 -0.2695635 0.13896002 -0.058951933 -0.18782169 -0.49178833 -0.13786896 -0.061503127

sochoonee 0.18483345 -0.11839012 0.17807509 -0.17338607 -0.1292448 -0.029900154 -0.45949483 -0.07338905 -0.313344 0.030627536 -0.023672493 -0.14917776 0.15303762 0.34510604 -0.15205695 0.33657998 -0.011326771 -0.06341887 0.13533586 0.12740387 0.22856225 -0.15653537 -0.10973493 -0.21651797 0.07091151 -0.19018477 -0.23566604 -0.035987098 0.16867873 -0.27956972 -0.0515709 -0.016401595 -0.026859766 -0.20449227 0.23958398 -0.23077707 -0.052396096 0.008172692 0.14304912 0.24082404 0.013598302 0.20478305 0.14808813 -0.12760086 0.13905497 -0.031762224 -0.015034203 -0.22035852 0.02070667 0.34459808 -0.28733182 0.07477937 0.26405495 0.13420966 0.32390216 0.15233317 -0.08714172 0.11912725 -0.32602862 0.06645035 0.035310775 0.037506346 -0.30248457 -0.21460955 -0.026344107 -0.012087038 0.075106315 0.20003253 0.09428645 0.11749025 0.31167775 0.43049568 -0.28875813 0.14862868 0.062417373 -0.034796305 0.31387797 -0.016971312 0.054624274 -0.03447201 -0.29911184 0.11343152 0.25469807 -0.0208199 0.11705259 -0.035183854 -0.09778151 0.51538604 -0.23288058 0.25593498 -0.33168584 -0.09331212 0.15270142 -0.18062478 0.22676189 -0.13216725 -0.3138586 -0.15763426 0.18991108 0.0554862

himatamoonni -0.24025156 0.74732995 0.13588808 0.0020900315 -0.30475065 0.102808736 0.24873976 0.21790202 -0.48689473 0.28500554 0.078521214 -0.065063976 0.2637413 0.058969095 0.22927177 0.5631822 0.23897325 -0.40856642 0.06948647 -0.19036475 0.42615995 -0.24045339 -0.092664585 0.25626457 -0.19338036 -0.29152477 -0.08035196 0.23030056 -0.26303458 -0.14691085 0.0072854455 0.257369 0.052480202 -0.13729966 -0.3404116 -0.044890277 0.12748007 0.27136928 0.10901408 0.23851073 -0.20212087 -0.040779077 0.14343509 -0.16475052 0.19259216 -0.09631659 -0.18175359 -0.1692441 0.21512893 0.28516096 0.30839273 0.16892767 -0.26644945 -0.15463538 -

0.035478644 -0.093010664 0.360656 -0.09505492 0.164456 -0.28850484 -0.11559568 -0.27541718 -0.08861778 -0.13449633 0.24357341 0.11085477 -0.042393774 0.52305704 -0.11089825 0.004491279 0.13121575 0.526053 -0.22025447 0.32441044 -0.3131116 -0.5332483 0.038963232 0.25990814 0.17464663 -0.10325752 -0.27346215 -0.05450863 0.03469467 -0.115731254 0.038011715 0.3754384 0.620045 -0.27254456 0.22783156 0.5178073 -0.44879645 0.27621737 -0.038083117 0.31400362 0.29573157 -0.18958338 0.3137314 -0.057966612 0.29896346 0.11602533

naggaa 0.05788046 0.6347339 0.18230917 0.07619059 -0.06561219 0.3674522 0.12643169 0.016221343 -0.23313634 0.465726 0.39072216 -0.16816346 0.31318152 0.1012763 0.4916209 0.4558045 0.4258486 -0.3159671 0.10622902 -0.023425326 0.4073254 -0.080258414 -0.29764608 0.23084597 0.17271471 0.15515634 -0.15828551 0.44006386 -0.20075178 -0.10020153 -0.14395314 -0.08958849 -0.20205182 0.2626795 -0.23064445 -0.094444625 -0.46110514 0.2247008 0.19480515 -0.17300296 -0.06536004 0.25790945 -0.11893505 0.012725147 -0.3432201 -0.08770425 0.080826044 0.17097977 -0.09675631 0.8112419 -0.3140453 0.035962533 -0.06891765 -0.5667101 0.01522785 -0.45075837 0.48623782 -0.12951298 0.33895695 -0.20971566 -0.046138987 -0.10708222 -0.0026168444 0.30082873 0.0069240746 0.32877305 -0.07111629 0.5081262 -0.034258585 -0.038285926 0.22206159 0.26134497 -0.4135844 0.05809597 -0.39422822 -0.2547192 0.3756849 -0.049092833 0.54253715 0.0729346 -0.025235524 -0.5394849 0.34396845 0.026134904 0.22542548 0.22200847 0.26899827 -0.2071131 0.10849741 0.5248451 -0.14411564 -0.123945154 0.32514817 0.28475797 0.41100904 -0.13204853 -0.28034762 -0.12169208 -0.044426154 0.04981947

aksiyoonaa 0.092987694 -0.060906533 -0.24779524 0.1068987 -0.26892444 -0.3077877 0.8811925 0.04963309 -0.30711982 -0.18207225 0.42648846 -0.12406242 0.2859442 -0.18273431 0.20602714 0.17708834 0.450994 -0.08478343 -0.100482546 -0.2687244 -0.031203328 -0.20911677 0.2731181 0.019692363 0.5818455 0.45748505 0.17386256 -0.08303197 -0.2807434 0.36556306 -0.023793489 0.29260018 0.20076251 -0.21503866 0.010218958 -0.037654802 -0.26368406 0.21400706 0.17876987 -0.68706536 -0.017888464 0.26834828 -0.22239521 0.12825704 -0.047581952 0.20449828 -0.29257482 0.1941031 0.08417383 0.47045925 0.22649588 -0.16976294 0.07718785 0.17902672 0.052194446 0.4627059 -0.31891277 -0.2203302 0.30250588 -0.19427013 0.5142994 0.07120062 -0.40610677 -0.23192045 -0.19068506 0.37869585 0.1351362 0.3107676 -0.35023418 0.12364403 0.28447548 0.79121727 0.312073 0.14220436 -0.020727456 -0.26017368 0.9492723 -0.29467186 0.52314174 0.24518271 0.032199703 -0.12162194 0.42915753 0.4976891 0.0019914722 -0.29057318 -0.1439795 -0.16996329 -0.33779636 0.24843957 0.31866035 -0.17728476 -0.007767014 -0.29313248 -0.030591518 -0.54646665 0.37939447 0.05681834 0.2124738 0.4332863

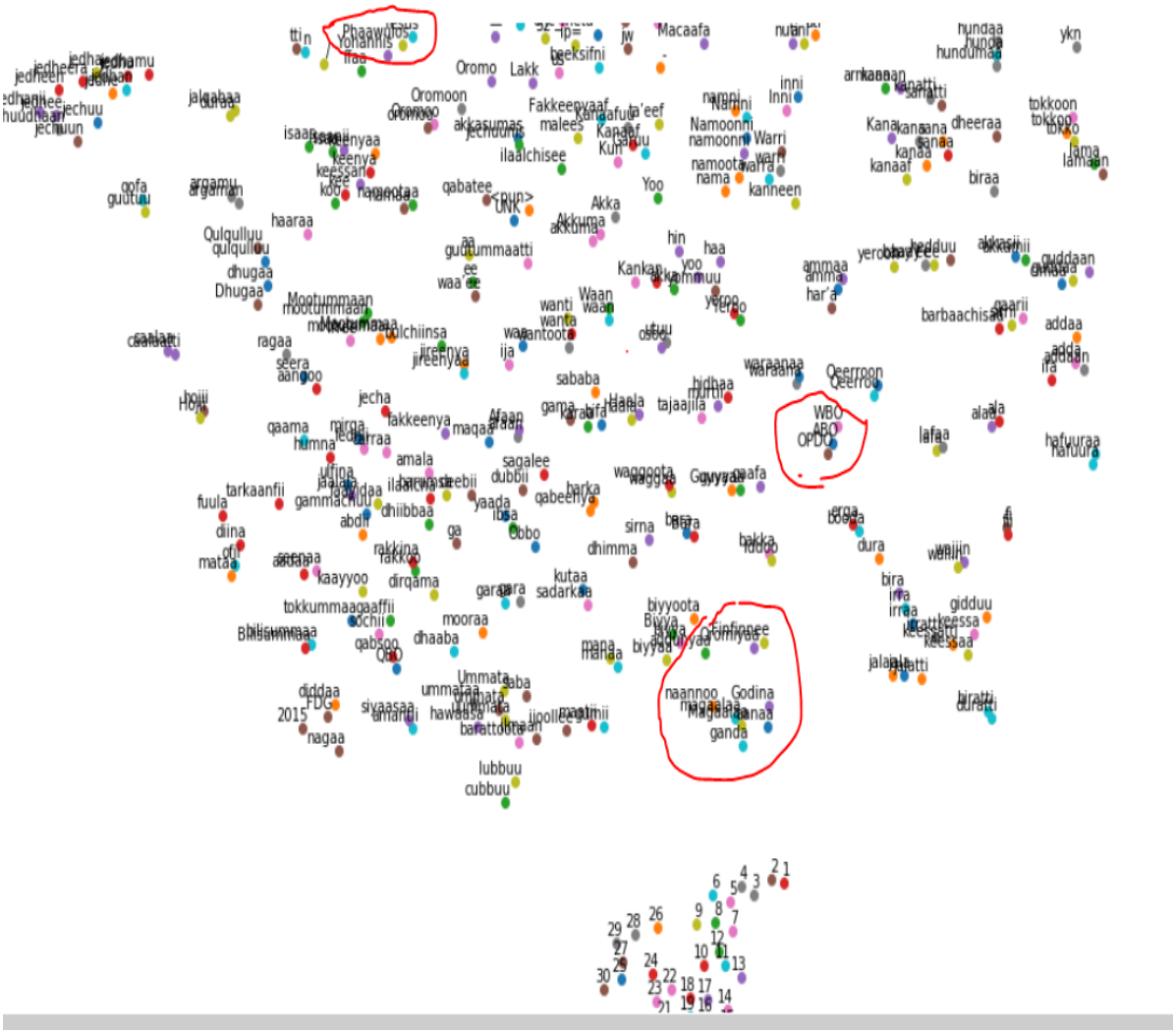
Appendix C

Sample screenshot of word relation of word embeddings

```
In [76]: w1 = "naqamtee"
         model.wv.most_similar (positive=w1,topn=10)

Out[76]: [('amboo', 0.6937726736068726),
          ('adaamaa', 0.6281418800354004),
          ('finfinnee', 0.5881106853485107),
          ('jimmaa', 0.5566478967666626),
          ('bishooftuu', 0.5328055620193481),
          ('giincii', 0.5197537541389465),
          ('wallaggaa', 0.5184266567230225),
          ('ambootti', 0.5174486637115479),
          ('mattuu', 0.512199878692627),
          ('jimmaatti', 0.5058525204658508)]
```

```
In [82]: w1 = "kadiir"
         model.wv.most_similar (positive=w1,topn=10)

Out[82]: [('umar', 0.5235335230827332),
          ('hassan', 0.4704226851463318),
          ('muusaa', 0.4558213949203491),
          ('jamaal', 0.4483434557914734),
          ('zuwaay', 0.4473126530647278),
          ('tasfaayee', 0.43459102511405945),
          ('yeshiwaas', 0.43025559186935425),
          ('kadir', 0.4292832911014557),
          ('ismaael', 0.4239320755004883),
          ('kunnenis', 0.4219245910644531)]
```

```
In [108]: w1 = "opdo"
          model.wv.most_similar (positive=w1,topn=5)

Out[108]: [('abo', 0.7079765200614929),
           ('tplf', 0.6628265380859375),
           ('d.u.o', 0.5630043745040894),
           ('eprdf', 0.5471445918083191),
           ('wayyaanee', 0.5450842380523682)]
```



Appendix D

Evaluation of Afaan Oromo NER

```
input> Tabor Wogii gara Adaamaa deeme
Tabor Wogii gara Adaamaa deeme
B-PER I-PER O    B-LOC   O

input> Baqqalaa Garbaan miseensa KFOtti
Baqqalaa Garbaan miseensa KFOtti
B-PER    I-PER   O        B-ORG

input>
```

Declaration

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

Declared by:

Name: Mekonini Kasu Taye

Signature: _____

Date: _____

Confirmed by advisor:

Name: Yaregal Assabie (PhD)

Signature: _____

Date: _____