



**JIMMA UNIVERSITY**

**JIMMA INSTITUTE OF TECHNOLOGY**

**SCHOOL OF GRADUATE STUDIES**

**FACULTY OF ELECTRICAL AND COMPUTER  
ENGINEERING**

**Named Entity Recognition for Gamo Language with  
Deep Neural Network**

By

Amanuel Zemach

This thesis is submitted to School of Graduate Studies of Jimma University in partial  
fulfilment of the requirements for the degree of

Master of Science

in

Computer Engineering

September 2020

Jimma, Ethiopia

**JIMMA UNIVERSITY**  
**JIMMA INSTITUTE OF TECHNOLOGY**  
**SCHOOL OF GRADUATE STUDIES**  
**FACULTY OF ELECTRICAL AND COMPUTER**  
**ENGINEERING**

**Named Entity Recognition for Gamo Language with  
Deep Neural Network**

By  
Amanuel Zemach

Advisor: Dr. Kinde Anlay

Co-Advisor: Mr. Fetulhak Abdurahman

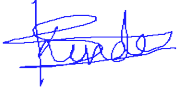
Submission Date: July, 2020

# Declaration



I, declare that this thesis entitled named entity recognition for Gamo Language with deep neural network is my original work and it has not been presented for any degree in this institution or any other universities, and all sources and references used for this thesis work have been appropriately duly acknowledged.

**Student Name:** AMANUEL ZEMACH **Signature:**  **Date:** 21/07/20

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

	Signature	Date
<b>Main Advisor:</b> <u>DR.KINDE ANLAY</u>		<u>23/07/2020</u>
<b>Co-Advisor:</b> <u>MR. FETULHAK ABDURHMAN</u>	_____	<u>21/09/ 2020</u>

This Thesis has been Approved by

Board of Examiners	Signature	Date
<b>External Examiner:</b> <u>MR. TEWODROS ABEBE</u>		<u>14/09/ 2020</u>
<b>Internal Examiner:</b> <u>MR. KRIS CALPOTURA</u>		<u>9/09/ 2020</u>
<b>Chair Person:</b> <u>MR. FETULHAK ABDURAHMAN</u>	_____	<u>21/09/ 2020</u>

# Acknowledgements

First and foremost, praises and thanks to Almighty God, for His blessings throughout my research work to complete the research successfully. Then I would like to express gratefulness to my advisor Dr. Kinde Anlay for his encouragement, guidance, understanding, and motivation throughout this thesis work. He has shown me how researches are produced and how to tackle a problem by investing his invaluable time. Also, I would like to thank my coadvisor Mr. Fetulhak Abdurrahman for providing me good guidance and feedback. The last but not the least, I take the opportunity to express my very sincere thanks to my teachers, colleagues, family, and friends. I will not be able to mention the name list here, but I had it in my memory.

# Contents

<b>Declaration</b>	<b>i</b>
<b>Acknowledgements</b>	<b>ii</b>
<b>Contents</b>	<b>iii</b>
<b>List of Figures</b>	<b>vi</b>
<b>List of Tables</b>	<b>vii</b>
<b>List of Abbreviations</b>	<b>viii</b>
<b>Abstract</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Background of Study . . . . .	1
1.2 Research Motivation . . . . .	2
1.3 Statement of the Problem . . . . .	3
1.4 Objectives of the Research . . . . .	4
1.4.1 General Objective . . . . .	4
1.4.2 Specific Objectives . . . . .	4
1.5 Research Methodology . . . . .	4
1.5.1 Review of Literature . . . . .	4
1.5.2 Corpus Collection and Dataset Development . . . . .	5
1.5.3 Tool Selection . . . . .	5
1.5.4 Prototype Development . . . . .	5
1.5.5 Conducting Experiments . . . . .	5
1.6 Scope and Limitation of the Study . . . . .	5
1.7 Contributions . . . . .	6
1.8 Thesis Organization . . . . .	6
<b>2 Theoretical Background</b>	<b>7</b>
2.1 Named Entity Recognition . . . . .	7
2.2 Applications of Named Entity Recognition . . . . .	8
2.2.1 Semantic Annotation . . . . .	8
2.2.2 Question and Answering . . . . .	8
2.2.3 Opinion Mining . . . . .	9

---

2.2.4	Machine Translation	9
2.2.5	Information Retrieval	9
2.2.6	Text Summarization	9
2.2.7	Text Clustering	10
2.3	Overview of Gamo Language	10
2.3.1	Written Language	10
2.3.2	Grammatical Arrangement	12
2.3.3	Gamo Language Nouns	13
2.3.4	Challenges of Gamo Language Named Entity Recognition	14
2.4	Named Entity Recognition Features	14
2.4.1	Word-level Features	14
2.4.2	List Lookup Features	15
2.4.3	Document and Corpus Features	16
2.5	Distributed Representation of Words	17
2.5.1	Skip-gram Model	18
2.5.2	Continuous Bag of Words Model	18
2.5.3	FastText	19
2.6	Semantic and Syntactic Information in Neural Word Vectors	20
2.7	Approaches for Named Entity Recognition	20
2.7.1	Traditional Approaches to NER	20
2.7.1.1	Rule Based Approaches	20
2.7.1.2	Machine Learning Approaches	21
2.7.2	Deep Learning	22
2.7.2.1	Recurrent Neural Networks	22
2.7.2.2	Convolutional Neural Network	25
<b>3</b>	<b>Literature Review</b>	<b>27</b>
3.1	Named Entity Recognition on Local Languages	27
3.2	Named Entity Recognition on Foreign Languages	30
<b>4</b>	<b>Named Entity Recognition System Design and Architecture</b>	<b>33</b>
4.1	Corpus Collection and Dataset Development	34
4.1.1	Data Collection	34
4.1.2	Data Preparation	34
4.1.3	Distributed Representations for Input	35
4.1.3.1	Word-level Representation	36
4.1.3.2	Character-level Representation	36
4.1.4	Tagging	37
4.2	Context Encoder	38
4.3	Tag Decoder	39
4.4	Evaluation Techniques	40
<b>5</b>	<b>Experimental Result Discussion and Evaluation</b>	<b>42</b>
5.1	Experimental Scenarios	42
5.2	Result Discussion and Evaluation	43

---

<b>6 Conclusion and Recommendation</b>	<b>51</b>
6.1 Conclusion . . . . .	51
6.2 Recommendation . . . . .	53
<b>Bibliography</b>	<b>54</b>

# List of Figures

2.1	Skip gram model . . . . .	18
2.2	CBOW model . . . . .	19
2.3	Structure of RNN . . . . .	23
2.4	Structure of a unit in LSTM . . . . .	23
4.1	GLNER System model . . . . .	33
4.2	The convolution neural network for extracting character-level representations of words. . . . .	37
5.1	Visualization of word embedding . . . . .	43
5.2	Word embedding with skip . . . . .	44
5.3	The precision, recall, and F1-Score of each categories using word embedding features. . . . .	48
5.4	The precision, recall, and F1-Score of each categories using word and Character embedding. . . . .	48
5.5	The precision, recall, and F1-Score of two GLNER models . . . . .	49
5.6	The precision, recall, and F1-Score of ANER and GLNER model. . . . .	50



# List of Tables

3.1	Researches on Ethiopian languages NER with machine learning . . . . .	30
3.2	Researches on Ethiopian languages NER with deep learning . . . . .	30
5.1	Corpus of Gamo language . . . . .	43
5.2	Effect of embedding algorithm on the first model . . . . .	44
5.3	Effect of embedding algorithm on the second model . . . . .	44
5.4	Labelled dataset of Gamo language . . . . .	45
5.5	Selection of encoding method . . . . .	46
5.6	Effect of number of hidden layers . . . . .	46
5.7	Optimization algorithm selection . . . . .	46
5.8	Mini-batch SGD . . . . .	47
5.9	Effect of Learning Rate . . . . .	47

# List of Abbreviations

<b>AMU</b>	Arbaminch Universty
<b>ANER</b>	Amharic Named
<b>BiLSTM</b>	Bidirectional Long Short Term Memory
<b>CBOW</b>	Continous Bag Of Words
<b>CNN</b>	Convolutional Neural Network
<b>CoNL</b>	Conference on Natural Language Learning
<b>GL</b>	Gamo Language
<b>GLNER</b>	Gamo Language Named Entity Recognition
<b>BiGRU</b>	Bidirectional Gated Recurrent Unit
<b>ILCA</b>	Initiative for Living Community Action
<b>LSTM</b>	Long Short Term Memory
<b>ML</b>	Machine Learning
<b>MUC</b>	Message Understanding Conference
<b>NE</b>	Named Entity
<b>NER</b>	Named Entity Recognition
<b>NLP</b>	Natural Language Processing
<b>POS</b>	Part Of Speech
<b>RNN</b>	Recurrent Neural Network
<b>SGD</b>	Stochastic Gradient Decent
<b>SSL</b>	Semi Supervised Learning
<b>SVM</b>	Support Vector Machine
<b>t-SNE</b>	t-Distributed Stochastic Neighbor Embedding
<b>TREC-8</b>	Text REtrieval Conference
<b>WS</b>	Window Size

# Abstract

A named entity (NE) is a word or a phrase that clearly identifies one item from a set of other items that have similar features. The term named entity refers to organization, person, and location names in a text. Named entity recognition (NER) is the process of locating and classifying named entities in text into predefined entity categories.

The proposed approach to Gamo language entity recognition involves four major layered processes. These are corpus collection and dataset development, distributed representation, context encoder, and tag decoder layer, all the layered processes are interconnected and arranged in a way that an output of a current process is an input to the next process. For distributed representation character and word level embedding used, context encoding BiLSTM was used, and decoding CRF was used.

NER for GL with deep learning based on features that extracted from word-level embedding alone achieved performance of accuracy 93.94%, precision 73.96%, recall 68.62% and F1-Score 71.19%, and word and character level embedding together achieved performance accuracy 94.59%, precision 76.15%, recall 72.49%, F1-Score 74.28%. So the second model improves precision 2.19%, recall 3.87% and F1-Score 3.09 respectively.

NER for GL with deep learning based on features that extracted with word-level embedding in conjunction with character level embedding outperforms deep learning based on feature extracted with word-level embedding.

**Keyword:GLNER,Encoder, Decoder**

# Chapter 1

## Introduction

### 1.1 Background of Study

Named entity (NE) could be a word or phrase that identifies one item from a group of other items with similar features[1]. The term named entity refers to an organization, person, and location names in a text.

Named entity recognition (NER) is the process of identifying and classifying named entities into predefined entity categories. The task focuses on a small set of coarse entity types and one type per named entity, which is called the coarse-grained NER[3],[4], on the other hand, fine-grained NER tasks[4],[5] focuses on a much larger set of entity types.

NER provides a service that can be used as a pre-processing step for a variety of downstream applications, such as information retrieval, question answering, machine translation, etc. Let us take a look at the application of semantic search to show the importance of NER in supporting various applications. Semantic search refers to a collection of techniques that enable search engines to understand the concepts, meaning, and intent behind user queries[6]. About 71% of search queries contain at least one named entity[7]. Recognizing named entities in search queries would enable the engine to better understand user intents, hence providing better search results.

When incorporating named entities into the search, entity-based language models[6], consider both individual terms and term sequences that have been annotated as entities (both in documents and in queries)[8].

Several approaches have been used to develop NER system for different languages, from dictionary lookup to deep neural networks. Traditional approaches to NER are broadly classified into three main streams: rule-based, unsupervised learning, and feature-based supervised learning approaches [9],[10].

The proposed research is the baseline research for the Gamo language in NLP. Unlike other languages such as English, Spanish, German and Chinese, the Gamo language does not have organized and structured text data, which makes it a challenging task to extract an entity in the language. Identifying named entities from unstructured text data is a difficult task. Identifying and classifying named entities into semantically meaningful classes is the ultimate goal of a named entity recognition task.

In this study, we have developed a named entity recognition model for Gamo language using deep neural networks. We considered the person, location, organization, and miscellaneous to be the major entity classes in our research.

## 1.2 Research Motivation

Gamo language is spoken in different areas of Ethiopia with different dialects, and it has a rich morphology. Things are getting automated nowadays, but there is no research that has been done in this language. In the near future, an enormous amount of text data will be introduced because of the wide use of electronic devices, to cope up with there requires studies related to NLP tasks associated with the language.

Therefore, development of NLP applications for this language is needed to address the emerging and current NLP challenge. In addition to these, this study is motivated by the contribution of NER tasks to other NLP studies.

---

## 1.3 Statement of the Problem

Since the advent of a computing system, the data has been stored in a particular format on a disk. One of the formats is a plain text in which all aspects of things are documented, so the amount of plain text data is getting very large. These large data presented different challenges, such as ambiguity and unstructuredness.

Extracting relevant information from ambiguous and unstructured text is a challenging task to do, internet search engines face such a problem.

Many of the language engineering applications such as Machine Translation, Question-Answering systems, indexing for Information Retrieval, and Automatic Summarization require an efficient information extraction mechanism. There is no relevant information extraction mechanism in Gamo language text.

The proposed research will answer the research questions: what kind of existing approach should be used for named entity recognition task for Gamo language text?

1. What data representation best suit for Gamo language text?
2. What learning approaches best suit for Gamo language text?

---

## 1.4 Objectives of the Research

### 1.4.1 General Objective

The general objective of this research is developing named entity recognition model for Gamo language with deep neural networks.

### 1.4.2 Specific Objectives

The general objective of this research is achieved by accomplishing the specific objectives of the study listed below.

- Assess the structure of GL and the linguistic patterns that refer to NEs.
- Review related studies that are conducted so far in different languages.
- Propose and design Gamo NER model.
- Develop a prototype of the proposed Gamo NER model.
- Test the performance of the developed prototype with various parameters.

## 1.5 Research Methodology

Different methods have been used for the successful completion of this study, all of them are discussed below.

### 1.5.1 Review of Literature

In order to understand the problem in depth and come up with a possible solution. To do so related researches have been reviewed, this consists of professional journals, formal research reports, university-affiliated bulletins, reports, monographs, and NER systems developed for other languages.

In addition, discussions were held with Gamo language linguistic experts regarding the linguistic nature of the language like the grammatical structure and the properties of Gamo language named entities.

## 1.5.2 Corpus Collection and Dataset Development

A well-developed and standardized dataset for linguistic analysis is important, but the same as other Ethiopian languages, the Gamo language does not have a well-organized dataset for research that has led us to prepare a dataset.

To do so, the manually written text documents were collected and converted to digital text data by employing typewriters, and each word in the document was assigned with the corresponding tag, which is called *data annotation*.

## 1.5.3 Tool Selection

We used python programming language to develop the Gamo language Named Entity recognition(GLNER) model. Packages and libraries used are gensim, pandas, scipy, sklearn, NLTK, numPy, and pyTorch.

## 1.5.4 Prototype Development

A prototype was developed to test the proposed model, followed by an experiment to evaluate efficiency.

## 1.5.5 Conducting Experiments

Experiments were conducted on the prototype to measure performance. The performances obtained throughout the conducted experiments are given in four evaluation metrics: accuracy, precision, recall, and F1-score.

## 1.6 Scope and Limitation of the Study

The purpose of this research is to develop named entity recognition for Gamo language. The scope of our study is limited to detection and classification of named entities of person, organization location, and MISC only.



---

## 1.7 Contributions

The main contributions of this research are:

- We have developed GLNER model that uses character and word level embedded features.
- We have prepared large unlabeled Gamo language corpus that can be used for future researches.
- We have developed labeled datasets for NER task.

## 1.8 Thesis Organization

The document is organized as follows. Theoretical backgrounds in NER are explained in chapter two. Literature review discussed in chapter three. Our proposed approach is explained in chapter four. The experimental procedures followed and results are discussed in chapter five. Finally, chapter six presents conclusions from experimental observations and recommendation to show further areas of improvement on GLNER models.

# Chapter 2

## Theoretical Background

In this chapter, the basic theoretical foundations used for solving the problem of Gamo language named entity recognition and the area of its application are discussed. In addition to that overview of Gamo language is given.

### 2.1 Named Entity Recognition

A *named entity (NE)* is a word or phrase that identifies one item from a set of other items with similar features[1]. The term named entity refers to an organization, person, and location names in a text. Named entity recognition (NER) is the process of identifying and classifying named entities in text into predefined entity categories. It is to recognize names of people, organizations, locations, and time, currency, percentage expressions in the text[2]. The task focuses on a small set of coarse entity types and one type per named entity, this kind of NER tasks is called coarse-grained NER [3][4], on the other hand, fine-grained NER tasks [4],[5] focus on a much larger set of entity types where a mention may be assigned multiple types. NER is part of many NLP tasks such as information retrieval, machine translation, question answering, sentiment analysis, etc. There were several approaches have been used to develop NER system for different languages, from dictionary lookup to deep neural networks. Traditional approaches to NER are broadly classified into three main streams: rule-based, unsupervised learning and feature-based supervised learning approaches [9][10].

## 2.2 Applications of Named Entity Recognition

Most often named entity recognition (NER) has been used as a preprocessing step for other natural language processing tasks[1]. It plays a crucial role in many natural language applications such as semantic annotation, question answering, and opinion mining[11]. It also plays a vital role in natural language processing applications such as machine translation, information retrieval, text summarization, and text clustering.

### 2.2.1 Semantic Annotation

Semantic annotation is a specific metadata generation meant to enable new information access methods. The annotation is based on the concept that the named entities constitute an important part of the semantics of the documents referred to therein. Semantic Annotation is about assigning entities in the text links to their semantic descriptions, and this kind of metadata provides both class and instance information about entities. Semantic annotation is applicable for sort of text web pages, regular documents, text fields in databases, etc[67]. This typically implemented with information extraction techniques, one of which is Named Entity Recognition is used to identify concepts to annotate[11].

### 2.2.2 Question and Answering

Question answering systems provide concrete answers to queries, and NER techniques are often used for this type of system as a means of facilitating the choice of answers. Indeed, in TREC-8 about 80% of the queries used to evaluate this kind of system were of the type who, where and when, which use to be answered with named entities of type person, organization, location, and date[11].

### **2.2.3 Opinion Mining**

Opinion Mining is a field of research that analyzes people's opinions about entities such as products, services, organizations, individuals, issues, events, topics, and their attributes. Opinion mining systems use named entity recognition as one of the preprocessing techniques to provide users with more related opinions. After named entities are identified in the process related opinions can be fetched easily[11].

### **2.2.4 Machine Translation**

Machine translation is the use of computers to automate the process of converting content in one language to another language. In the process of translation, named entity recognition helps to disambiguate some words with similar surface form as of named entities[13].

### **2.2.5 Information Retrieval**

Information retrieval is the task of retrieving relevant documents for the user query. Information retrieval benefits from named entity recognition in two ways. The first one is in the process of recognizing named entities in the user's query. The second benefit is recognizing named entities within the documents to extract only relevant ones based on identified named entities[14].

### **2.2.6 Text Summarization**

Text summarization is a process of summarizing a piece of text to a shorter version, reducing the size of the initial text while at the same time preserving key informational elements and the meaning of content. Named entities give a clue about the topic of a text, they are among the most information-dense tokens of the text and largely define the domain of the text. Therefore, named entity recognition enhances the identification of important text segments in text summarizer[15].

## 2.2.7 Text Clustering

Text clustering is a grouping of textual documents in a way that, documents having similar content will be clustered in the same group. Named entity recognition can be used in text clustering for ranking resulted in clusters based on the ratio of entities associated with each cluster. It enhances the process of analyzing the nature of clusters and also improves the clustering approach using selected features[14].

## 2.3 Overview of Gamo Language

Gamo language is spoken in the south-western part of Ethiopia, in the Gamo Zone of the Southern Nations, Nationalities, and People's Region (SNNPR). The language is called gamottso by the speakers, who refer to themselves as Gamo. Over one million people in the whole SNNPR consider themselves to be Gamo (Office of the Population Census Commission, 2007). Gamo Language is being given as a language of instruction in the lowest grades in primary school and it is given as a course up to university[36].

Gamo language is an Omoto language of the Omotic family. The classification of Omotic is somewhat problematic, and there has been, and still is the discussion regarding the external and internal classification. Omotic is often classified as a branch of Afroasiatic/Afrasian, alongside Berber, Chadic, Cushitic, Egyptian and Semitic[35]. Theil argues that the arguments for placing Omotic under Afroasiatic are not convincing and that Omotic should be regarded as an independent language family. The classification is based on mass comparison and lexico-statistics, both of which are methodologically problematic.

### 2.3.1 Written Language

Gamo language is mainly written using the Latin script, but the Ethiopian script is also used. The school books in Gamo use the Latin script. The alphabet contains all the letters used in the English alphabet, including ten new letters, Nine of these are composed of two

letters. The tenth is the numeral < 7 > which symbolizes the glottal stop, but never in word initial position. In some cases, the value for the different letters are not the same as for English. < C > stands for the ejective affricate /č'/, < q > for the ejective stop /k'/, < x > symbolizes the ejective affricate /tʃ'/, and < j > is sometimes used to symbolize the semivowel /j/ and sometimes used to symbolize the sibilant /Z/. Sometimes the word /haZé/ 'big wasp' written as < hajje > and < hazhze >.

The second consonant of a consonant cluster is always geminated in written Gamo, but there seems to be variation as to whether a consonant following < y > is geminated or not. < y > symbolizes the consonant /j/, and the vowel /i/ when this occurs after another vowel. A few of the letters in the alphabet do not correspond to a phoneme in the language, these are < v >, < nh > and < ng >. There is no labiodental voiced consonant, like /v/ in Gamo. < nh > represents a nasalized vowel, a feature which is not phonemic in Gamo[38]. < ng > represents one of the allophones if a nasal in a consonant cluster. The letter < ny > represents ñ, a phoneme that is, at best, very rare in Gamo[39]. The letter < f > is used alongside < p >[36].

The difference between Gamo Language and English

Major differences in the letters and phonetic Systems of English and the Gamo Language , English has 26 alphabetical letters, which are

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

The Gamo Language has 35 alphabetical letters, which are

A B C CH D DH DZ E F G H I J K L M N NH NY O P PH Q R S SH T TH U V W X Y Z 7

Alphabetically, Gamo Language have an intersection of 26 letters. In the Gamo language, we have 9 more letters, of which 8 are hybrids of existing letters and 1 is a number representing a glottal sound.

The hybrid letters are Ch, Dh, Dz, Nh, Ny, Ph, Sh and Th. The glottal sound is 7, which is taken to represent the international phonetic symbol ʔ. The Gamo Language have the above major differences form wise. In the letter to sound representation, the Gamo language is phonemic, meaning that one letter represents only one sound. A letter cannot represent more than one sound unlike many cases in English. We can see these through some examples. Xerox - in this word, we have letter 'x' both initially and finally.

The initial 'x' represents /z/ while the final /ks/. Cancer - in this word we have letter 'c' initially and medially. The initial 'c' stands for /k/ while the medial one for /s/. Such differences do not exist in the writing and reading system of the Gamo language. The correspondence between letter and sound is one to one. Once, you know what sound a letter represents, then that letter represents only that same sound wherever you find it.

The other main difference between English and the Gamo language writing and reading system is that lengthening of sounds in words is represented by doubling of letters in the Gamo Language writing. When a consonant sound is lengthened, it is geminated and we double-write it in that position. When a vowel sound is lengthened, we say it is long and double-write it in that position.

Let's take some examples here:

words with short vowels

1. awa - sun
2. eqa - standup

words with long vowels

- aawa - father
- eeqa - Priest

Words with short consonants

1. aga - baptise
2. tama - fire

words with geminated consonants

- agga - leave it
- tamma - ten

## 2.3.2 Grammatical Arrangement

### *Word Order*

#### *Word Order in Noun Phrase*

- The word order in the noun phrase is: modifier + noun + postposition
- Relative clauses are formed by making use of short verb form.
- When relative clauses occur to the right of the noun they modify, they take morpheme -iss to nominalize which inflects for a person, number, and gender.

### ***Word Order in Main Clause***

- Gamo is a strict verb final language(**SOV**).
- The object occurs immediately before the verb(**SOV**).
- The order between direct object and indirect object is free when the verb is ditransitive.
- The direct object is always realized in absolutive case and the indirect one is realized in the oblique case.

### **2.3.3 Gamo Language Nouns**

Gamo language nouns are words used to name or identify any of a class of things, people, and places or ideas. Gamo language nouns can be classified into tangible (countable nouns), intangible (non-countable), collective nouns, common nouns and proper nouns. Tangible nouns are nouns which represent something that can be seen or touched, for instance *asa*, *wonbara*, and *ketha* (human, chair, and house) denote things that can be counted. On the other hand intangible nouns represent something that cannot be seen or touched, for instance nouns like *tuma*, and *na77aatetha* (dark, and childhood) represent things that cannot be counted. Collective nouns are used to denote class of things or concepts instead of particular entities. For instance *kafo*, *mole*, and *wude* (bird, fish, and herd) are names for class of things. Common nouns denotes particular group of things and concepts in real world. For instance *dorsa* and *deysha* (Sheep and Goat) refer particular group. The another class which is main focus of this research is proper nouns. Proper nouns are nouns that identify specific person, organization, location, and others. For instance *Kasto*, *Gacho*, and *bore* are names of a person, *Tophphiya*, *Arbamince*, *Bonke*, and *Chencha* are names of locations, and *Arbamince yuniversity*, and *Arbamince hosptale* are names of organizations.



### 2.3.4 Challenges of Gamo Language Named Entity Recognition

One of the main challenges of named entity recognition is that some proper nouns can belong to open class category. For instance names of a location or organization can be added to the classes. Another challenge is ambiguity. In Gamo language same name may refer to different entities, for instance, Bonke can be the name of a person or location. Additionally, challenges like nested named entities and ambiguity are exhibited in Gamo language texts.

## 2.4 Named Entity Recognition Features

Supervised learning approaches for named entity recognition is extremely sensitive to selection of features[20]. Features are characteristic or attributes of words designed for algorithmic consumption. Features can be represented as boolean, numeric or nominal values. Features used for named entity recognition can be classified into three categories which are Word-level features, List lookup features, and document and corpus features [16].

### 2.4.1 Word-level Features

Feature extraction and selection plays a crucial role in deep learning. In deep learning the features extracted automatically from the representation of the data. The main features for the NER task have been identified based on the different possible combinations of available word and tag context. The features including prefix and suffix for all words. The term prefix/suffix is a sequence of first/last few characters of a word, which may not be a linguistically correct(meaningful prefix/suffix). Besides, various gazetteer lists have been developed for use in the NER task. The different combination from the following set for inspecting the best feature set for NER task have been considered: Previous NE tag, POS tags, first word, digit information, and gazetteer lists[40]. The following are the details of the set of features that were applied to the NER task:

- **Context word feature:** Previous and next words of a specific word.
- **Word suffix:** Word suffix information is useful for the identification of NEs, this feature may be used. It's used in two different ways. The first is a fixed-length word suffix of the current and/or the surrounding word(s), the second and more convenient approach is to modify the feature as a binary value. The variable-length suffixes of a word can be matched to predefined lists of useful suffixes for different NEs classes. Various suffixes that may be particularly useful for the detection of persons (e.g.-babu,-da,-di, etc.) and location names (e.g.-land,-pur,-lia, etc.) have also been considered. Both types of suffixes have been used here.
- **Word prefix:** Prefix information of a word is also important, a fixed-length prefix of the current and/or the surrounding word(s) might be treated as features.
- **Part of Speech (POS) Information:** POS information of the words, such as the current and surrounding word(s) can be used as a feature, the better way is to use a coarse-grained POS tagger. For NER, the coarse-grained POS tag is used, with only the following POS tags: Compound noun, Common noun, Compound proper noun, Proper noun, Postpositions, Number quantifier, and so on.
- **Named Entity Information:** The NE tag of the previous word is considered as a feature.
- **First word:** If the current word is the first word of a sentence. Then the 'First-Word' feature is set to 1 otherwise, it is set to 0.
- **Digit features:** Several binary digit features are considered depending on the presence and number of digits in the token, the combination of digits and punctuation symbols, the combination of digits and symbols. These binary-valued features are important for the recognition of miscellaneous named entities.

## 2.4.2 List Lookup Features

**Gazetteer Lists:** Gazetteer lists are used as the binary-valued features of the CRF. If the current token is in a particular list then the corresponding feature is set to 1 for the current and/or the surrounding word(s)[31]; otherwise, set to 0. The following is the list of gazetteers:

- **Organization suffix word:** This list contains words that are helpful in identifying organization names. The 'OrganizationSuffix' feature is set to 1 for both current and previous words.
- **Person prefix word:** This is useful for the detection of the name of a person. The 'PersonPrefix' feature is set to 1 for the current and the next two words.
- **Middle name:** These words generally appear in the names of the person. The 'MiddleName' feature is set to 1 for the current, the previous, and the next words.
- **Surname:** These words usually appear as their parts at the end of a person's names, the feature 'SurName' is set to 1 for the current word.
- **Common location word:** This list contains the words that are part of location names and appear at the end, the feature 'CommonLocation' is set to 1 for the current word.
- **Frequent word:** The feature 'RareWord' is set to 1 for those words that are not on the list.
- **Person name:** This list contains the first name of person names, the feature 'Person-Name' is set to 1 for the current word.
- **Location name:** This feature is the first name of person names, the feature 'Person-Name' is set to 1 for the current word.
- **Organization name:** This list contains the organization names and the feature 'OrganizationName' is set to 1 for the current word.

### 2.4.3 Document and Corpus Features

The document features are defined in terms of both document content and document structure. Large collections of documents (corpora) are also an excellent source of features[40]. Features that extend beyond single word and multi-word expression and include meta-information on document and corpus statistics, document and corpus features are:

- Multiple occurrences and multiple casing
- Entity co-reference and alias
- Statistics for Multiword units

## 2.5 Distributed Representation of Words

Word representations are mathematical objects associated with each word, and the representations are often in vector form. The contents of this vector representation are the characteristics of each word. The value of each dimension is the value of the specific feature. Traditionally, one of the most obvious ways to represent words is through a hot vector representation. Every word( $w$ ) in the vocabulary( $V$ ) has a unique index. The words are then represented by a vector of size in which the index of the word is one and the rest is zero[43].

- Home =  $[0,0,1,0,0,0,\dots,0,0,0,0]$
- House =  $[1,0,0,0,0,0,\dots,0,0,0,0]$

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

- “Kasto slept on the bed”
- “Gacho slept on the couch”

If the above sentences are represented as one hot vector algorithm trained with the first sentence “Kasto” as person name might fail to tag the word “Gacho” in the second unseen sentence. The reason is that the representation of the two person names doesn’t capture the relation between the two names. On the other hand, distributed representations use lower dimensional real valued vectors to represent words. Each dimension represents latent feature about that specific word. The representation may look like, as given below:

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

This kind of representation is hoped to capture semantic and syntactic relation between words[1]. Similar words are expected to be mapped to near vectors in the vector space. For example, in the two sentences mentioned above, if they are represented by a distributed representation, even if the tag did not see the word "Gacho" in the training data, the vector will be very similar to the word "Kasto" which makes the decision easier.

### 2.5.1 Skip-gram Model

This model accepts the word  $W_i$  and predicts the words around the given word ( $W_i$ ) that are context words ( $W_{i-2}, W_{i-1}, W_{i+1}, W_{i+2}$ ). Context words don't have to be immediate words. Some words can be skipped inside a given window size to look forward and backward from the target word. The Skip-gram model has a hidden neural network. The input layer consists of one-hot encoded vector of the vocabulary[12]. The Skip Gram model is shown in Figure 2.1 below.

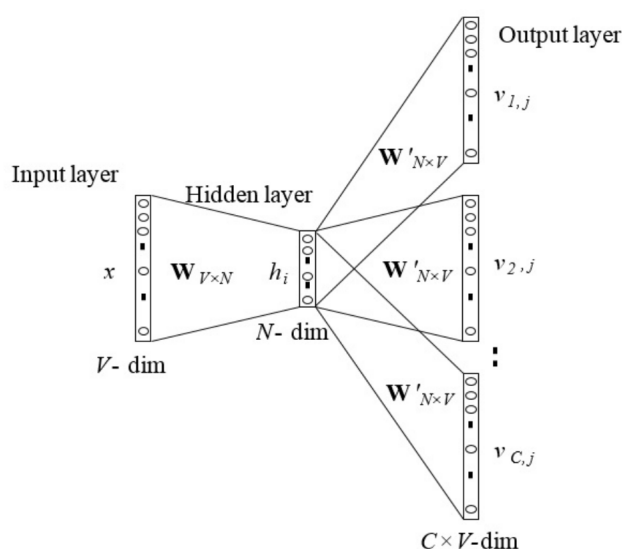


FIGURE 2.1: Skip gram model

### 2.5.2 Continuous Bag of Words Model

Continuous bag of words(CBOW) is reverse of skip gram model. Given the context ( $W_{i-2}, W_{i-1}, W_{i+1}, W_{i+2}$ ) the task is to predict the word. CBOW uses the average vector of the input context words to compute the output of the hidden layer and uses the output of the input layer hidden layer weight matrix and the average vector as the output[12]. The CBOW model is shown in Figure 2.2 below.

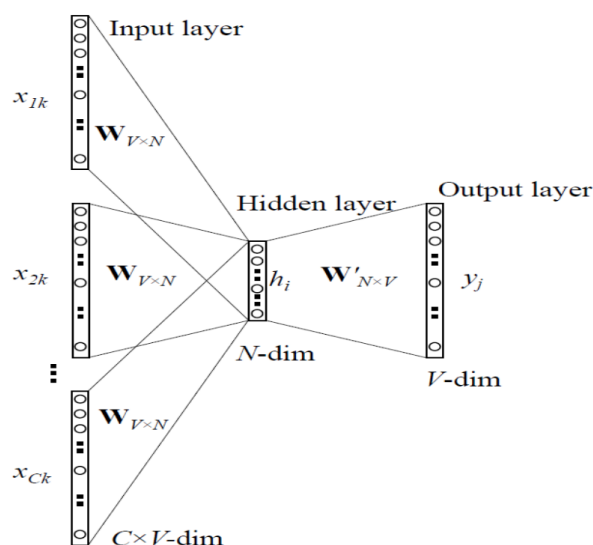


FIGURE 2.2: CBOW model

### 2.5.3 FastText

FastText is a library designed by Facebook’s AI Research Lab to learn word embeddings and text classification. It is another word embedding method which is an extension of the word2vec model. FastText represents every word as an n-gram of characters instead of directly learning vectors for words. So, for instance, take the word, “artificial” with  $n=3$ , the fastText representation of this word is  $\langle ar, art, rti, tif, ifi, fic, ici, ial, al \rangle$ , where the angular brackets indicate the beginning and end of the word. This helps to capture the meaning of shorter words and enables the embeddings to understand suffixes and prefixes. Once the word is represented using the character n-grams, the skip-gram model is applied to learn the embeddings. This model is considered to be a model word bag with a sliding window over a word because the internal structure of the word is not taken into account. As long as the characters are inside this window, the order of the n-grams does not matter.

FastText is working well with rare words. So even if a word hadn’t been seen during training, it could be broken down into n-grams to get its embeddings[69].

## 2.6 Semantic and Syntactic Information in Neural Word Vectors

Models discussed in the previous sections are capable of capturing many linear dependencies between words. The syntactic properties of words, such as inflective forms of a word, are most closely represented on a continuous vector space. Distributed vector representation of words shows the state of the art on the test set for the measurement of syntactic and semantic word similarities[21]. More complex similarity checks can be carried out by simple algebraic operations with vector representation of words.

## 2.7 Approaches for Named Entity Recognition

In this section, different approaches that have been used for the development of named entity recognition tasks are discussed, ranging from traditional machine learning to deep learning.

### 2.7.1 Traditional Approaches to NER

Traditional approaches to NER task are broadly classified into three main streams: rule-based, unsupervised learning, and feature-based supervised learning approaches[16].

#### 2.7.1.1 Rule Based Approaches

Rule-based approaches rely on handcrafted language rules prepared by language experts[17]. Generally, rules-based approaches consist of a set of patterns using grammatical, syntactic and orthographic features. Such systems can perform better for restricted domains. They can detect complex entities that may be difficult for learning models. However, the main disadvantages of this approach are its lack of portability, robustness and high maintenance costs due to a slight change in data[18].

### 2.7.1.2 Machine Learning Approaches

Machine learning approaches use a collection of training examples to automatically induce rules. These rules are sequence labeling algorithms for categorizing named entities [16]. Machine learning approaches are defined as statistical models for making predictions about named entities in a given text[17]. There are 3 major machine learning approaches, they are supervised learning, unsupervised learning, and semi-supervised learning.

1. ***Supervised learning:*** Nowadays, supervised learning approaches are the most dominant approaches to the tasks of the named entity classification. Popular supervised learning approaches include Hidden Markov, Maximum Entropy Models, Conditional Random Fields, Vector Support Machines, Decision Tree, etc[40]. We used a conditional random field to classify named entities into predefined categories.

***Conditional Random Field(CRF):*** can be considered as a logarithmic linear model suitable for sequence tagging when context information is taken into account. Suppose there are two random variables  $X$  and  $Y$ ,  $P(Y|X)$  is the conditional probability distribution of  $Y$  given  $X$ , if  $Y$  can form a Markov random field, then the conditional probability distribution  $P(Y|X)$  can be considered as a conditional random field, if  $X$  and  $Y$  have the similar structure, like  $X = (X_1, X_2, \dots, X_n)$ ,  $Y = (Y_1, Y_2, \dots, Y_n)$  are random variable sequences represented by linear chain, then  $P(Y|X)$  is the linear chain conditional random field (linear-CRF). In sequence tagging tasks, CRF can define features to capture the relationship between labels, and use these features to give scores to different predicted tag sequences.

2. ***Semi-supervised approaches:*** Semi-supervised learning methods use both labeled and unlabeled corpus for training. The impetus behind semi-supervised learning is to resolve the lack of labeled training data by using large unlabeled corpus[11]. The main method in Semi-Supervised Learning is called bootstrapping, which includes a small measure of control at the start of the learning process. The model is trained on the initial set of labeled data and then the prediction is made on a separate set of unlabeled data. The performance of the model is then improved by using the predictions of previously developed models[18]. Many languages do not have large annotated data for the training of supervised learning algorithms. An unsupervised learning approach is proposed to solve this problem.



3. ***Unsupervised learning approach:*** Unsupervised learning methods do not require any structural information on the data. Common unsupervised learning tasks include clustering, where the objective is to separate instances into groups; novelty detection, which identifies the few instances that are very different from the majority; and dimensional reduction, which seeks to represent each instance with a lower-dimensional feature vector[19].

## 2.7.2 Deep Learning

Deep learning, also known as deep structured learning or hierarchical learning, is a set of machine learning algorithms that try to learn a layered input model commonly known as neural networks. It enables computational models made up of multiple layers of processing to learn data representation at multiple abstraction levels. Deep learning algorithms are capable of discovering complex structures from large datasets using back propagation algorithms to update their internal parameters[1].

Researchers in this field are trying to create models that learn representations from large unlabelled data[22]. Various deep neural network architectures such as convolutional deep neural networks, deep belief networks, and recurrent neural networks have been applied to many areas and shown impressive results. These areas include natural language processing, computer vision, speech recognition, and bioinformatics[23]. There are many kinds of deep neural networks and deep neural network architectures. The neural networks used in our research are discussed below.

### 2.7.2.1 Recurrent Neural Networks

Traditional neural networks do not consider the relations and the order of input data. For example, if a NN is trying to classify what happened in a video, since the traditional NN is dealing with sample data individually, it can not use the former frames to help understand the later ones. The Recurrent Neural Network (RNN) was proposed to solve sequence problems by taking into account the continuity of individual data samples. There is a loop in the RNN units to keep the history information and forward the information from the previous hidden state to the next. The basic structure of the RNN is shown in Figure 2.3. There is a single unit of RNN on the left side of this figure, the loop of this unit can

forward the state to the next step at the current time. At each time step, the input  $x_t$  RNN generates the hidden state  $h_t$ . According to the right side of Figure 2.3, the loop operation of a single unit in the RNN may be regarded as a sequence of copies of the same unit in chronological order, and each copy shall pass the information to the copy in the next step[43]. There are several variations in RNN that depend on different needs.

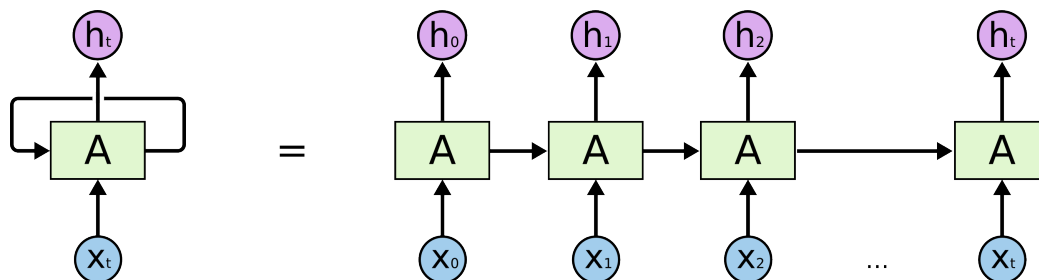


FIGURE 2.3: Structure of RNN

### **LSTM**

RNNs can keep the memory of each time step before the current one to help predict the result. The only thing needed to predict the next word of the text is to consider a few adjacent words, the word to be predicted has little distance from its related words. The RNN works in this case. But what if the relevant information is far from the word to be predicted? It is difficult for standard RNNs to link the information to be predicted and its relevant information as the gap between them increases. In order to solve the problem of "long-term dependence," Hochreiter and Schmidhuber[33] proposed the Long-Term Memory, which aims to remember information in the short term. Compared to standard RNNs, LSTM is better able to overcome the problem of vanishing gradients by using a more complex architecture unit as shown in Figure 2.4. There are three gates designed to control the way information passes through the cell. The three gates are described as follows:

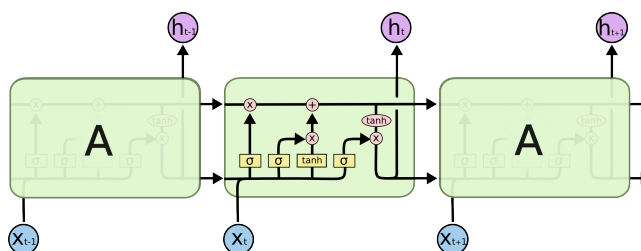


FIGURE 2.4: Structure of a unit in LSTM

1. **Forget gate:** Forget the gate controls how the information from the current input  $x_t$  and the output from the previous time step  $h_{t-1}$  flow in the current cell, after by an activation function  $\sigma$  has been calculated. Following equation 2.1 it outputs a vector  $f_t$  with all elements between 0 and 1. This vector points which information is allowed to pass.

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.1)$$

2. **Input gate:** The input gate decides how much new input information should be added to the cell state. First, a sigmoid function decides which information needs to be updated, and a tanh function generalizes the  $\hat{C}_t$  which means the contents available for update. Then, the old cell state  $C_{t-1}$  can be replaced by adding new information into the cell state and get  $C_t$ . This process is represented by equations 2.2, 2.3 and 2.4:

$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.2)$$

$$\hat{c}_t = \tanh (W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2.3)$$

$$c_t = f_t \times c_{t-1} + i_t \times \hat{c}_t \quad (2.4)$$

3. **Output gate:** What to output depends on the output gate. The output gate will filter the cell state  $C_t$  and output the hidden state  $h_t$  at current timestep. As equations 2.5 and 2.6 states, it uses a sigmoid layer to calculate a vector decides which part of the information in  $C_t$  is allowed to output ( $O_t$ ).  $O_t$  is multiplied by  $C_t$  to get the final result.

$$o_t = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.5)$$

$$h_t = o_t \cdot \tanh(c_t) \quad (2.6)$$

### ***Bi-directional LSTM***

LSTM only considers the influence of past information. Sometimes future features can also influence the prediction. Bi-directional LSTM (Bi-LSTM) [34] enables the use of both past features and future features for a certain time step.

For example, if a input sample sentence has 10 words (10 timesteps)  $x_1, x_2, \dots, x_{10}$ , there exists two separate LSTM cells, the Bi-LSTM network works as follows:

- For forward-direction LSTM cells, the order of input words is  $x_1, x_2, \dots, x_{10}$ , the output is a set of hidden states  $\vec{h}_1, \vec{h}_2, \dots, \vec{h}_{10}$ ;
- For backward-direction LSTM cells, the order of input words is  $x_{10}, x_9, \dots, x_1$ , and output the hidden states set  $\overleftarrow{h}_{10}, \overleftarrow{h}_9, \dots, \overleftarrow{h}_1$ ;
- Concatenate two sets of hidden state as  $[\vec{h}_1, \overleftarrow{h}_1], [\vec{h}_2, \overleftarrow{h}_2], \dots, [\vec{h}_{10}, \overleftarrow{h}_{10}]$
- For input  $x_t$  at each time step, get output states  $H_t = [\vec{h}_t, \overleftarrow{h}_t]$ , then the rest operations are as same as in LSTM network.

### 2.7.2.2 Convolutional Neural Network

Convolutional neural networks (CNN) are feed-forward neural networks which are usually used for image recognition and classification of objects. Deep learning models using convolutional neural networks (CNNs) have shown to achieve remarkable results for speech recognition tasks, especially in the context of semantic parsing [66]. In our case, CNN is used for modeling character-level information in the NER task[25]. CNNs use layers with convolving filters that are applied on top of word vectors obtained from the word embedding. All word representations are kept static after the word vectors have been learned and only the hyperparameters of downstream layers of the model are optimized. Considering a k-dimensional word vector corresponding to the  $i^{th}$  word in the sentence  $x_i \in R^k$ , a sentence of length n will be represented by an  $n \times k$  matrix as follows:

$$x_{1:n} = x_1 \oplus x_2 \oplus \dots \oplus x_n \quad (2.7)$$

Where  $\oplus$  serves as the concatenation operator. Hereby,  $x_{i:i+j}$  represents the concatenation of the embedded word vectors. Following the concatenation, a convolutional filter with a window size of h words is applied to produce a new feature in the convolutional layer. Let  $w \in R^{hk}$  be a convolution filter, b be a bias term, and  $\sigma$  being a non-linear function.

Then a feature  $c_i$  is generated from a window of words  $x_{i:i+h-1}$  as follows [66]:

$$c_i = \sigma(w \cdot x_{i:i+h-1} + b) \quad (2.8)$$

This filter technique is applied to all  $h$  possible window of words in the sentence, to produce a feature map  $c \in R^{n-h+1}$  :

$$c = [c_1, c_2, \dots, c_{n-h+1}] \quad (2.9)$$

Given the feature map, max-pooling is applied over the feature map which then takes the maximum feature value  $\hat{c} = \max(c)$ . By doing so, the max-pooling extracts the most important feature and deals with varying sentence lengths. Once the most important features from the convolutional layer have been extracted, these can serve as the input layer for a fully-connected layer, typically with some dropout and a softmax activation function  $S_j$ .

$$S_j = \frac{e^{x_j}}{\sum_{n=1}^N e^{x_n}}; \forall j \in 1, \dots, N \quad (2.10)$$

To account for word sequences using CNNs, each filter in a convolutional layer can be used as a time-step (a filter per word), this way word order is kept by sequentially feeding the convolutional filters to a downstream task.

# Chapter 3

## Literature Review

This chapter contains a review of literature both on local languages named entity recognition and foreign languages named entity recognition researches. All of them are briefly discussed below.

### 3.1 Named Entity Recognition on Local Languages

There is no related work in Gamo Language, but there are related works in other Ethiopian languages in Afan Oromo and Amharic.

The first research on named entity recognition was conducted in the Ethiopian language by Moges Ahmed in the Amharic language[40]. He presented the CRFs based on the entity recognition for the Amharic language. CRF is a statistical method that can be used to recognize the names of the Amharic input text. The research is aimed to test the applicability of the CRF based Named Entity Recognition System (NERS) for the Amharic language. On the research, they used 10,405 tokens were taken from WIC corpus and manually tagged text data. All the features, except prefixes and suffixes, were considered and achieved a performance of 74.61% of F-measure.

The second research in the Amharic language was conducted by Besufekad Alemu[41]. Like the research done by Moges[40], they used supervised machine learning called CRF. This research was carried out by making use of 13538 tokens taken from WIC corpus

and tagged manually. They conducted fifteen experiments and the roles of each feature in different experiments were tested, to identifying optimal features for ANER. They conducted fifteen experiments and the roles of each feature in different experiments were tested, to identifying optimal features for ANER. They achieved the highest performance in the work is 80.66%, with a window size of two on both sides, previous and next tag of a current word and prefix, and suffix with length 4. From the findings, prefix and suffix feature previous, and next NE tag of a token and window size of four are the observed optimal feature sets and widows size for recognizing Amharic named entities.

In contrast to the previous two works, the third research in Amharic was done ANER using a hybrid approach[42], using a previously prepared corpus[40][41]. The corpus is annotated with Person, Organization, Location, and Other named entity types and consists of a total of 12191 tokens. The rule-based component of the system was built by adopting the two rules used for building a rule-based NER for Arabic[45]. After selecting a baseline feature set, different experiments were conducted to explore the performance of their hybrid NER system in combination with different feature sets. They discovered that the hybrid NER model has not improved significantly compared to the pure ML NER model. The hybrid approach for ANER using the NE type has been predicted from the rule-based component they built as a feature that has not improved the performance of ANER. The POS feature used in their experiment has improved the performance of ANER. The nominal flag value feature used in combination with POS information demonstrated a significant improvement in performance. The two ML algorithms, namely SVM and J48, which they used in their experiments resulted in better performance compared to the CRF ML algorithms. In terms of accuracy, J48 outperformed SVM among the two ML algorithms[42].

The previous three researches were done with a machine learning approach and manually designed features, one of which was with a hybrid approach. The fourth one was with automatically extraction features with word embedding. In addition to machine learning algorithms, they implemented deep neural networks for ANER. They achieved 92.6%, 90.4%, 90.6% for BLSTM, LSTM, and MLP respectively. According to their conclusion from the two types of LSTM cells, bidirectional LSTM outperforms unidirectional LSTM and Among all classifiers used in their experiments SVM achieved the highest  $F_1$ -score[43].

The fifth research was done using deep learning with word embedding. In this work, the word embeddings were generated in two ways, through word2vec [14] and random vectors. In the random vector setting, all the words in the corpora are initialized with random values. These random vectors are fed to the bi-directional LSTM model to classify the words into the six categories. The word2vec model were trained using a softmax function or negative sampling [46]. Then a bi-directional LSTM model was used to classify the words based on the word vectors generated either by word2vec or randomly. They achieved 77.2% of precision, 63.4% of recall, and 69.7% of F1 - score[44].

The first research in Afan Oromo done was, based on a hybrid approach: machine learning and rule-based. But they didn't implement rule-based component. They implemented the machine learning component using the CRFs algorithm, with a corpus of more than 23,000 words out of which around 3600 are Nes was used. The parser checks the correctness and they achieved 77.41% Recall, 75.80% Precision and 76.60% F1-measure. They concluded that features play a vital role than the change in the training data size. This, on the other hand, justifies that we used the best feature combination for the development of the system[47].

The second study in afan Oromo also done using hybrid approach, they achieved an overall improvement of the AONER performance. It is capable of recognizing person, organization and location name, date time, currency, and address name. Based on experimental results, they conclude that the hybrid approach outperforms the pure Rule-based approach and the pureML-based approach. Their hybrid NER system for Afaan Oromo outperforms the Afaan Oromo NER in terms of f-measure when applied to AONERcorp dataset with f-measure of 83.22% for Person named entities, f-measure of 84.06% for Location named entities, f-measure of 80.83% for Organization named entities, f-measure of 86.06% for Date and Time named entities, f-measure of 83.67% for Percentage named entities and 80.58% for Address named entities[48].



TABLE 3.1: Researches on Ethiopian languages NER with machine learning

Research	Approach	Features used	F1-score(%)
[40]	CRF	Context features, POS tag, prefix, suffix	74.61
[41]	CRF	Context features, word pairs, word shape, prefix, suffix	80.66
[42]	SVM, J48	Context features, POS tag, prefix, suffix, nominal feature	85.9, 96.1
[47]	CRF	Position features, Normalized features, words shape, POS tag, prefix, suffix	76.6
[48]	CRF	Morphological, POS tag, word length, dot flag, capitalization, Gazetteers flags, nominal, NE tag	82.52

TABLE 3.2: Researches on Ethiopian languages NER with deep learning

Researche	Input representation	Context encoder	Tag decoder	F1-score (%)
[43]	Word-level embedding with Skip-gram	-	SVM, BiLSTM	95.5, 92.6
[44]	Word-level embedding with Skip-gram	BiLSTM	softmax	69.7

## 3.2 Named Entity Recognition on Foreign Languages

A lot of research has been conducted in this field, with different approaches from traditional approaches to deep learning. Rule-based NER systems rely on hand-crafted rules that can be drawn based on domain-specific gazetteers[49] and syntactic-lexical patterns[50]. Such traditional systems are mainly based on hand-crafted semantic and syntactic rules for the recognition of entities. Rule-based systems work very well if the lexicon is exhaustive. As a result of domain-specific rules and incomplete dictionaries, high precision and low recall are often observed from such systems and systems can not be transferred to other domains.

Clustering is a typical approach of unsupervised learning[51]. Clustering-based NER systems extract named entities from clustered groups based on context-likeness. In doing so, lexical resources, lexical patterns, and statistics computed on a large corpus can be used to infer named entities(NEs) in a corpus. Research with an unsupervised learning approach, for gazetteer building and named entity ambiguity resolution[52]. This model combines entity extraction and disambiguation based on simple heuristics.

Supervised learning based NER task, in which given annotated data samples, features are carefully designed to represent each training example. Machine learning algorithms are then applied to learn a model to recognize similar patterns from unseen data. Feature engineering is critical process to supervised NER tasks. Feature vector is a text abstraction where a word is represented by one or more Boolean, numeric or nominal values[51]. Word-level features [53], list lookup features [54], and document and corpus features [56] have been widely used in various supervised NER systems. Bikel et al. [56] proposed the first HMM-based named entity recognition system, called *IdentiFinder*, to identify and classify names, dates, time expressions, and numerical quantities. Zhou and Su [53] extended *IdentiFinder* by making use of mutual information. The difference is that Zhou’s model assumes mutual information independence while HMM assumes conditional probability independence. Besides, Szarvas et al. [57] developed a multilingual NER system by making use of the C4.5 decision tree and AdaBoostM1 learning algorithm. Major merit again is that it provides an opportunity to train several independent decision tree classifiers through different subsets of features then combine their decisions through a majority voting scheme. Given labeled samples, the principle of maximum entropy is applied to estimate a probability distribution function that assigns an entity type to any word in a given sentence based on its context. Borthwick et al. [58] proposed a “maximum entropy named entity” (MENE) by applying(making use of) the maximum entropy theory. MENE can make use of a various range of knowledge sources in making its tagging decisions. Other systems using maximum entropy can be found in [59]. [62] proposed an SVM-based system, which uses an uneven margins parameter leading to achieve better performance than original SVM on a few datasets. SVM does not consider “neighboring” words when predicting an entity label, but CRF takes context into account. McCallum and Li [61] proposed a feature induction method for CRFs in NER. Krishnan and Manning [60] proposed a two-stage approach based on two linked CRF classifiers, the second CRF uses the latent representations drawn from the output of the first CRF.

In recent years, deep-learning NER models have become dominant and have achieved state-of-the-art results[19]. There are three key strengths in applying deep learning techniques to NER. First, NER benefits from a non-linear transformation that produces non-linear mappings from input to output. Compared to linear models ( for instance,

log-linear HMM and linear-chain CRF), deep-learning models can learn complex and intricate features from data via non-linear activation functions. Second, deep learning saves significant effort in the design of NER features. Traditional feature-based approaches require a considerable amount of engineering skills and expertise in the field. Deep learning models, on the other hand, are effective in automatically learning useful data representations and underlying factors from raw data. Third, deep learning NER models can be trained in an end-to-end paradigm, by gradient descent. This property enables us to design relatively complex NER systems. One such research is [63] which builds a biomedical named entity recognition system based on deep learning. In this paper huge potential feature information represented as word vectors are generated by neural networks based on unlabeled biomedical text files, their system achieved F-score 71.01% on GENIA regular test corpus. From their experiments, they concluded that a deep learning approach can effectively perform better on biomedical NER. Similarly, [25] makes use of CRF and BLSTM, the model relies on two sources of information about words: character-based word representations learned from the supervised corpus and unsupervised word representations learned from unannotated corpora. They used English, Dutch, German, and Spanish corpus to evaluate their approach. The results achieved were 90.94%, 81.74%, 78.76%, 85.75% F-scores respectively on four languages. Another approach proposed in [65] uses neural network architecture that automatically detects Word and character-level features using a hybrid bidirectional LSTM and CNN architecture. They have used lookup tables to transform discrete features such as words and characters into continuous vector representations, which are then concatenated and fed into the BLSTM network. A convolutional neural network is used to induce character-level features. They achieved 91.65% F-score on CoNLL data. Most of the researches done in this area used the same approach with some modifications to improve the accuracy of their model.

Based on the literatures reviewed above, we have used character and word level representations for feature extraction, deep-learning for encoding feature, and CRF for decoding(prediction) to investigated for the GLNER system.

# Chapter 4

## Named Entity Recognition System Design and Architecture

The proposed approach to Gamo language entity recognition involves four major layered processes. These are corpus collection and dataset development, distributed representation, context encoder, and tag decoder layer. All layered processes are interconnected and arranged in such a way that the output of the current process is an input to the next process, as shown in Figure 4.1 below. Each step in the GLNER model described as follows.

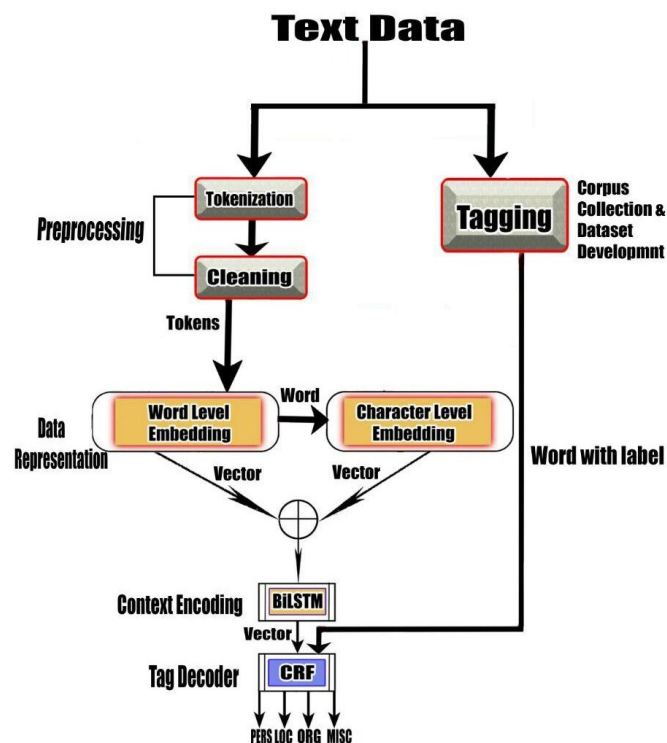


FIGURE 4.1: GLNER System model

## 4.1 Corpus Collection and Dataset Development

Corpus collection and dataset development is a pre-processing stage in which all text data are collected and prepared for the next step. Two forms of data have been prepared. The first is a collection of unlabelled text corpus for data representation, including data collection and pre-processing. The second is labeling dataset for training, validation, and testing the model, in which all the words labeled or tagged with the corresponding tag, tags are LOC, PERS, MISC, ORG, and O.

### 4.1.1 Data Collection

There are more than 72 different languages in Ethiopia, but there is no organized dataset developed by institutions for all languages. Some researchers have developed datasets for Amharic and Afan Oromo for their study[40][47].

Like most Ethiopian languages, Gamo language does not have organized dataset. So the first task we did was developing a dataset, to do so we collected about 2.6 MB of plain text from ILCA and Arbaminch Universty (AMU) softcopy, and Dehub radio and television organization, Arbaminch branch hardcopy. We found an average of 49 sentences with named entities from 7 modules and 300 sentences with named entities, which were collected from AMU and ILCA respectively. We found 4400 sentences with named entities that were collected from Dehub radio and television organization, 10 people involved in converting hardcopy to softcopy.

Totally, we extracted about 19567 sentences from which we found 5000 sentences with named entities.

### 4.1.2 Data Preparation

Data preparation is a preprocessing step for data representation. Data preparation in our model involves two processes which are tokenization and cleaning, both of them are described below:

### ***Tokenization***

Tokenization is a method of maintaining a list of words or tokens that we can use in our deep learning models. This involves converting the raw text to a list of words. Which would be done by splitting the document by white space, including " (space), new lines, and tabs. Tokens are provided as input to the next process called **Text cleaning**.

### ***Text Cleaning***

After tokenization, tokens are used as input and then unnecessary tokens are cleaned up from the corpus, such as punctuation like commas, apostrophes, quotes, question marks, section markers (such as I and II) and hyphenated descriptions such as armor-like were removed. Then the important tokens are provided as input for the next process called **data representation**. In this research, we implemented a **distributed representation**.

### **4.1.3 Distributed Representations for Input**

The tokens that are generated from the previous step forms a list of words, which are provided as an input for embedding called distributed representation. In distributed representation data represented in n-dimensional space and low dimensional real-valued dense vectors where each dimension represents a hidden feature.

The distributed representation automatically learns from the text representation and captures the semantic and syntactic properties of the word that are not explicitly present in the input of the NER. One of the benefits of using dense and low-dimensional vectors is that most neural network toolkits do not perform well with very high-dimensional, sparse vectors. The main advantage of dense representations is the power of generalization, where certain features may provide similar clues, it is worth providing a representation capable of capturing these similarities[64]. Distributed representation captures the correlation between the words, unlike to one hot representation[1]. **Word** and **character-level** distributed representations were used in our model.

### 4.1.3.1 Word-level Representation

Word embeddings are a type of word representation that enables words of similar meaning to be represented in a similar way. A distributed representation of the text is perhaps one of the key breakthroughs in the performance of deep learning methods to address the challenges of natural language processing[64].

CBOV and Skip-Gram model were used as part of the Word2Vec approach to word embedding learning. The Skip-gram model is an efficient method for learning high-quality distributed vector representation that captures syntactic and semantic word relationships[22].

We used the skip-gram model for word embedding learning over our corpus. Where each word is represented by a point in the embedding space, and these points are learned by moving around based on the words that surround the target word. The vector-space representation of words provides a projection where words with similar meanings are locally clustered within space.

Notably, larger window sizes appear to produce word embeddings with less syntactic information, considering the window size in a range[1-16][68], so we considered the window size in the range[3-21]. We've used a 100-dimensional vector space. Word level embedding generates a vector representation of each word, then concatenated with a character level embedding representation vector and provided as an input to the **context encoder**.

### 4.1.3.2 Character-level Representation

Character level representation is capable of extracting sub-word level information such as prefix and suffix. Another advantage of character-level representation is that it naturally deals with out-of-vocabulary. Thus, the character-based model can infer representations for unseen words and share information of morpheme-level regularities.

CNN is an efficient approach to extract morphological information (like the prefix or suffix of a word)from characters of words and encode it into neural representations[25]. We have used CNN-based architectures for extracting character-level representation.

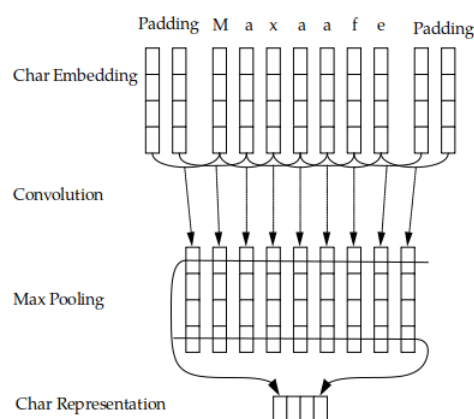


FIGURE 4.2: The convolution neural network for extracting character-level representations of words.

The character level embedding takes word sequence from the pre-trained word embedding on our corpus. As shown in figure 4.2, it takes word sequence and appends padding if the number character of the input word is less than a character window size. A conservative CNN configuration is used with 30 filters and a character window size(kernel size) of 5 with a rectified linear (relu) activation function. This is followed by a max-pooling layer that reduces the output of the convolutional layer by half, we use a maxpool to extract meaningful features out of our convolution layer.

Next, the 1D output from the CNN part of the model is flattened to one long 1D vector to represent the features extracted by the CNN. CNN outputs the extracted features in a 25-dimensional vector space. Then the character-level representation vector is concatenated with the word-level embedding before feeding into a **BiLSTM context encoder**.

#### 4.1.4 Tagging

Tagging/labeling is the process of assigning a tag to each word in a sentence. Thus, the second dataset we developed was labeled dataset for training, validation and testing proposed GLNER model. To do so, sentences with named entities were extracted from the collected text data, then each word in a sentence labeled with a tag. Tags are **LOC**, **PERS**, **MISC**, **ORG**, and **O**. For example, name of locations tagged/labeled with **LOC**, name of a person labeled with **PERS**, name of an organization labeled with **ORG**, another named entities labeled with **MISC**, all the special characters labeled with **O**. In doing so, the tagging scheme called BIO was used[1], it works as below.



***BIO tagging Scheme:***

- I - Word is inside a phrase of type TYPE
- B - If two phrases of the same type immediately follow each other, the first word of the second phrase will have tag B-TYPE
- O - Word is not part of a phrase

Some of GLNER sentence available in the dataset annotated in BIO scheme:

Arbamince yuunversite 6 sha7appe bollara oraththa tamaareta ekkides .

B-ORG I-ORG O O O O O O O

Data splitted in the dataset as training, validation, and test dataset. We used 90% of the dataset for training, 10% of the dataset for validation from the training dataset, and 10% of the total dataset used to test the model. In labelling the dataset two people were employed, they are not a linguistic expert, but their mother tongue is Gamo language.

## 4.2 Context Encoder

After having a good data representation, the next step on the GLNER model is to learn the BiLSTM context encoder from the **input representations**. BiLSTM used to capture long-term dependencies and obtain the whole representation of an input sequence. Long short-term memory (LSTM) cells are the building block of recurrent neural networks. While LSTM cells in the feed-forward neural network process text from left to right, BiLSTMs also consider the opposite direction, this allows the model to uncover more patterns as the amount of input information increases. In other words, the model takes into account not only the sequence of tokens after the token of interest but also before the token of interest.

BiLSTMs make efficient use of past information (via forward states) and future information (via backward states) for a specific time frame[51]. Thus, a token encoded by a BiLSTM will contain evidence from the whole input sentence. BiLSTM operates on sequential data, it takes as input a sequence of vectors  $(x_1, x_2, x_3, \dots, x_n)$  that generated from character and word level embedding, and return another sequence  $(h_1, h_2, h_3, \dots, h_n)$

that represents information about the sequence at every step in the input. For a given sentence  $(x_1, x_2, x_3, \dots, x_n)$  containing  $n$  words each represented as a 100-dimensional vector, the LSTM encodes the representation  $h_t$  of the left and right contexts of the sentence at each word  $t$ . Then concatenate the left and right context representations,  $h_t = [\vec{h}_t; \overleftarrow{h}_t]$ . These representations effectively include a representation of a word in a context that is useful for sequence labeling. Thus, the token encoded by the BiLSTM will contain evidence from the entire input sentence. Then, encoded features (context-dependent representations) fed to **tag decoder**.

### 4.3 Tag Decoder

Tag decoder is the final step of the GLNER model. It takes context-dependent vector representations encoded by the context encoder as inputs and predicts the sequence of tags corresponding to the input sequence. We used the CRF layer as the tag decoder on top of the bi-directional LSTM layer. In contrast to softmax, CRF takes into account the correlations between neighborhood labels and decodes the best chain of labels for a given input sentence[51]. Formally, we used  $x = x_1, x_2, \dots, x_n$  to represent a generic input sequence where  $x_i$  is the input vector of the  $i^{th}$  word.  $y = y_1, y_2, \dots, y_n$  represents a generic sequence of labels for  $x$ .  $Y(x)$  denotes the set of possible label sequences for  $x$ . The probabilistic model for sequence CRF defines a family of conditional probability  $p(y|x; W, b)$  overall possible label sequences  $y$  given  $x$  with the following form:

$$p(y|x; W, \mathbf{b}) = \frac{\prod_{i=1}^n \psi_i(y_{i-1}, y_i, x)}{\sum_{y' \in Y(x)} \prod_{i=1}^n \psi_i(y'_{i-1}, y'_i, x)} \quad (4.1)$$

Where  $\psi_i(y', y, x) = \exp(W_{y', y}^T x_i + b_{y', y})$  are potential functions, and  $W_{y', y}^T$  and  $b_{y', y}$  are the weight vector and bias related to label pair  $(y', y)$ , respectively. For CRF training, we use the maximum conditional likelihood estimation.

For a training set  $\{(x_i, y_i)\}$ , the logarithm of the likelihood (a.k.a. the log-likelihood) is given by:

$$L(W, b) = \sum_i \log p(y|x; W, b) \quad (4.2)$$

Maximum likelihood training selects parameters such that the  $L(W, b)$  log-likelihood is maximized. For a sequence CRF model (only interactions between two successive labels are considered). Decoding is to search for a sequence of labels  $y^*$  that has the highest conditional probability:

$$y^* = \operatorname{argmax}_{y \in Y(x)} p(y|x; W, b) \quad (4.3)$$

## 4.4 Evaluation Techniques

Entity recognition involves identifying both entity boundaries and entity types. In "exact-match evaluation," a named entity (NE) is considered to be correctly recognized only if its boundary and type match the ground truth. Precision, Recall, and F-score are computed on the number of true positives (TP), false positives (FP), and false negatives (FN)[19].

- True Positive (TP): Entities that are recognized by NER and match ground truth.
- False Positive (FP): Entities that are recognized by NER but do not match ground truth.
- False Negative (FN): Entities annotated in the ground truth that are not recognized by NER.

Precision measures the ability of a NER system to present only correct entities, Recall measures the ability of a NER system to recognize all entities in a corpus, and F-score is the harmonic mean of precision and recall, and the balanced F-score is most commonly used.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (4.4)$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (4.5)$$

$$F_1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4.6)$$

As most of NER systems involve multiple entity types, it is often required to assess the performance across all entity classes. Two measures are commonly used for this purpose: macro-averaged F-score and micro-averaged F-score. Macro-averaged F-score computes the F-score independently for each entity type, then takes the average (hence treating

all entity types equally). Micro-averaged F-score aggregates the contributions of entities from all classes to compute the average (treating all entities equally). The latter can be heavily affected by the quality of recognizing entities in large classes in the corpus. we used Micro-averaged F-score for our study.

# Chapter 5

## Experimental Result Discussion and Evaluation

### 5.1 Experimental Scenarios

The Experiment was done based on two scenarios, depending on the data representation with the same encoding and decoding:

1. The first scenario is based on word level embedding and
2. The second scenario is based on word level embedding along with character level embedding.

Those experiments were done to analyze the effect of different features that extracted at data representation on learning using deep neural network. In doing so, we evaluated two models and compared them with each other, and we took the best performing model of them. Then the best performing model were again compared with ANER[44], in order to show that our model is best performing with our dataset.



We have experimented different word embedding algorithms by varying window size and we took the best performing algorithm and window size to carry out further experiment. We experimented with two models, the first experiment was done using word-level embedding alone, with the same context encoder and tag decoder, the result is specified in Table 5.2. The Result in Table 5.2 and 5.3 is measured in terms of f1-score.

TABLE 5.2: Effect of embedding algorithm on the first model

Training algorithm	window size(ws)					
	ws=3	ws=5	ws=7	ws=9	ws=11	ws=13
cbow	70.7%	71.14%	68.48%	67.46%	69.89%	70.44%
skip gram	70.83%	70.89%	71.04%	70.98%	71.19%	72.22%
fastText	71.06%	70.56%	70.04%	70.02%	68.44%	66.78%

The second experiment was done using word-level embedding in conjunction with character level embedding, with the same context encoder and tag decoder, the result is specified in Table 5.3.

TABLE 5.3: Effect of embedding algorithm on the second model

Training algorithm	window size(ws)					
	ws=3	ws=5	ws=7	ws=9	ws=11	ws=13
cbow	70.04%	72.24%	69.31%	71.56%	73.54%	71.34%
skip gram	74.01%	71.52%	72.60%	72.10%	74.11%	71.15%
fastText	73.19%	73.13%	71.08%	72.43%	73.18%	73.21%

As shown in the table 5.3 skip gram algorithm with window size of 11 outperforms others. So, we took it for further experiment, in the rest of the experiments word embedding of skip gram algorithm with window size of 11 was used.

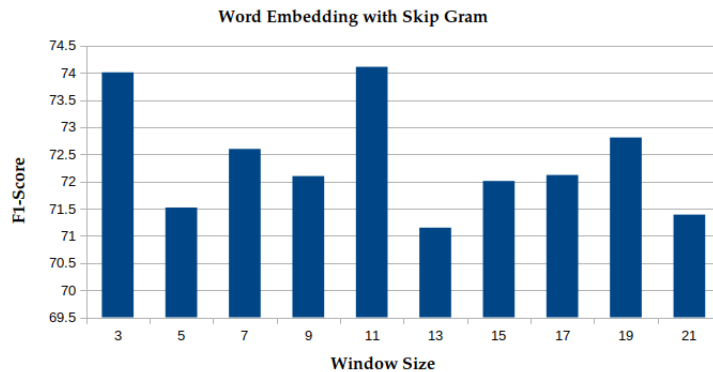


FIGURE 5.2: Word embedding with skip

As shown in figure 5.2 we got a better result with a window size of 11, when the window size increases we didn't find a better result. Our result confirms, larger window sizes appear to produce word embeddings with less syntactic information[68].

### ***Character-level embedding***

Our model works with character embeddings as well, which is generated from pre trained word embedding. For each word, convolution and max-pooling are applied to extract a new feature.

Words are padded with special padding characters on both sides depending on the window size of CNN. We randomly initialized a embedding with values drawn from a uniform distribution with range  $\left[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}\right]$  to output a character embedding of 25 dimensions with filter size of 30 and character window size 5. The same set of random embeddings was used for all experiments. The resulting vector of character embedding is concatenated with other word-level feature vectors and fed to **BiLSTM** context encoder.

We have experimented our model by varying the number of character window size and number of filter in our preliminary experiments only character size of 5 and number of filter 30 better than others.

### ***Labelled dataset***

TABLE 5.4: Labelled dataset of Gamo language

Class	Trianing		Validation		Test	
	Begning	Contiounm	Begning	Contiounm	Begning	Contiounm
Person(PER)	2668	2521	262	233	284	272
Location(LOC)	1719	1973	268	227	168	195
Organization(ORG)	999	4190	82	338	98	403
MISC	4402	1240	575	157	396	115

We developed labeled dataset for Gamo language, it contains the number of named entities per each category, as specified in the table 5.3. We made use of this labeled dataset for trianing, validation and testing our model.



***Effect of context encoder***

TABLE 5.5: Selection of encoding method

	Accuracy	Precision	Recall	F1-Score
Bivanilla	93.08%	74.59%	62.34%	67.92%
BiGRU	93.87%	74.35%	65.48%	69.63%
BiLSTM	94.59%	76.15%	72.49%	74.28%

As shown in the table 5.4, bidirectional LSTM best suits our model comparing to the other context encoders. So, we applied BiLSTM as the basic architecture to encode sequence context information.

***Effect of number of hidden layers***

Number of hidden layers is the most important criteria, it affects optimization algorithm upon training the NNs.

TABLE 5.6: Effect of number of hidden layers

	Number of hidden layers						
	1	5	10	20	40	80	160
Accuracy	84.20%	92.76%	93.67%	94.59%	94.08%	94.46%	93.87%
Precision	49.00%	72.86%	76.48%	76.15%	75.79%	78.87%	75.66%
Recall	12.76%	60.67%	66.32%	72.49%	67.78%	69.87%	69.25%
F1-Score	20.25%	66.21%	71.04%	74.28%	71.56%	74.10%	72.31%

We have varied the number of hidden layers for the given input sets and the results are shown in the table 5.5. It is noted that best fitting number of hidden layer are 74.28% and 74.10% of F1-Score with 20 and 80 number of hidden layers respectively.

***Effect of optimization algorithm***

Parameter optimization is performed with stochastic gradient descent (SGD). To reduce the effects of “gradient exploding”, we use a gradient clipping of 5.0.

TABLE 5.7: Optimization algorithm selection

	Accuracy	Precision	Recall	F1-Score
Adam	91.98%	66.56%	63.49%	64.99%
SGD	94.59%	76.15%	72.49%	74.28%

TABLE 5.8: Mini-batch SGD

	Batch size					
	4	8	16	32	64	128
Accuracy	92.60%	92.34%	92.68%	91.66%	91.59%	90.72%
Precision	73.70%	73.52%	69.47%	70.83%	70.74%	71.29%
Recall	60.67%	62.45%	66.63%	56.90%	55.13%	47.80%
F1-Score	66.55%	67.53%	68.02%	63.11%	61.96%	57.23%

we experimented our model with SGD, adam and mini-batch optimizer, from three of them SGD optimizer performed the best result in preliminary experiments as shown in the table 5.6 and 5.7.

### ***Effect of Learning Rate***

Another crucial issue in training neural network is the learning rate. In order to ensure the gradient descent method would perform better, it requires the value of the learning rate within appropriate range. If the learning rate is too large, it makes unstable learning for the models, and if it is set at too small a value, the training time will take much too long. The appropriate learning rate can achieve high efficiency under stable training, which can reduce the training time.

TABLE 5.9: Effect of Learning Rate

	Learning rate			
	0.0001	0.001	0.01	0.1
Accuracy	86.70%	92.26%	94.59%	78.47%
Precision	55.00%	77.04%	76.15%	16.85%
Recall	24.16%	57.22%	72.49%	14.54%
F1-Score	33.58%	65.67%	74.28%	15.61%

For SGD optimizer, we experimented the learning rate in the range [0.0001 - 0.1], with a weight decay of 0.05, and learning rate of 0.01 resulted better than others as shown in table 5.8.

After getting all the above suitable parameters and hyper-parameters, we experimented our model in two scenarios, and measured the performance in terms of precision, recall and F1-score for each class of named entity in the dataset of Gamo language. The figures shows the comparison between the three standard evaluation measures for classes of named entity.

The intention of this comparison is to show the difference in percentage of each measure in each class. The precision measure in each class gave higher percentage over the other 2 measures.

### *The first scenario*

The experment was done by making use of features that extracted from word embedding along with deep learning, the result is shown in the figure 5.3 below. The performace of this expermanet are accuracy 93.96%, precision 74.25%, recall 70.29% and F1-Score 72.22%.

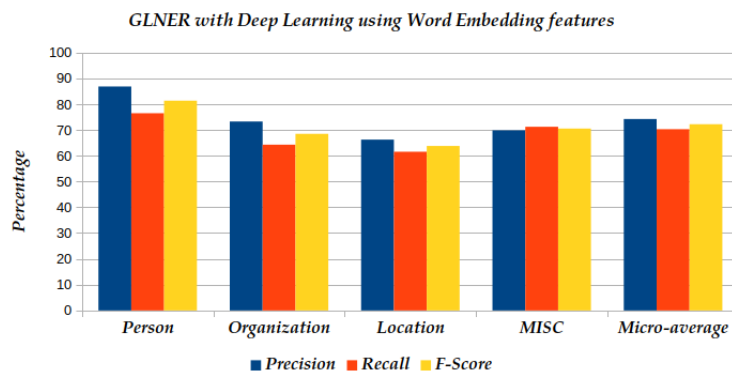


FIGURE 5.3: The precision, recall, and F1-Score of each categories using word embedding features.

### *The second scenario*

The experment was done by making use of features that extracted from word embedding and character level representation with the same encoding with the first experment, the result is shown in the figure 5.4 below. The performace of this expermanet are accuracy 94.59% precision 76.15% recall 72.49% and F1-Score 74.28%.

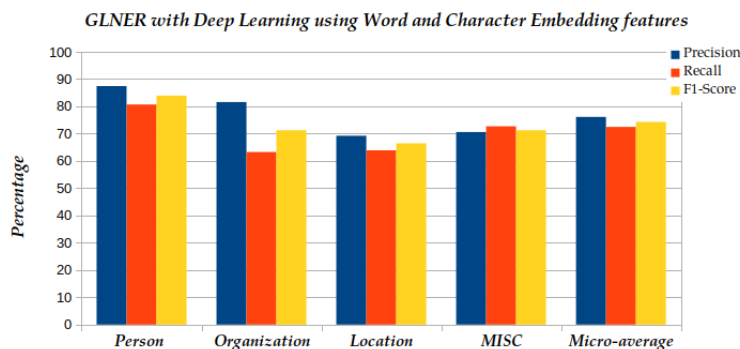


FIGURE 5.4: The precision, recall, and F1-Score of each categories using word and Character embedding.

### *Comparative analysis of two scenarios*

In this section the experiments based on two scenarios compared with one another, which are learning based on word-level embedding, and word-level and character-level embedding.

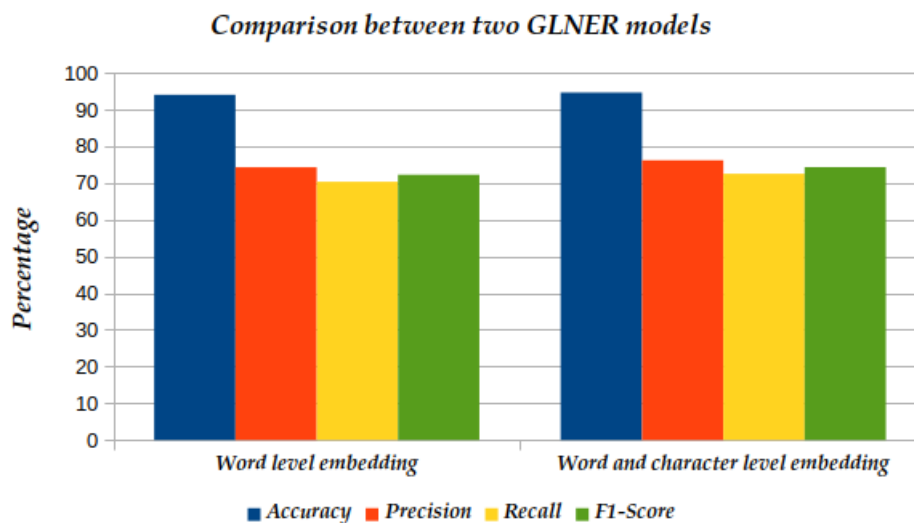


FIGURE 5.5: The precision, recall, and F1-Score of two GLNER models

As shown in the figure 5.5 above, NER for GL with deep learning based on features that extracted with word-level embedding in conjunction with character level embedding outperforms, deep learning based on feature extracted with word-level embedding. NER for GL with deep learning based on features that extracted from word-level embedding alone achieved performance of precision 74.25%, recall 70.29% and F1-Score 72.22%, and word and character level embedding together achieved achieved performance precision 76.15%, recall 72.49%, F1-Score 74.28%. So the second model improves precision 1.9%, recall 2.2% and F1-Score 2.06 respectively.

### *Comparative analysis of GLNER with other Language*

Comparing GLNER model with other deep learning model is inappropriate because in deep learning the number of data set matters the performance. To compare the dataset is needed to be same for the models. Because of which we simply compared our model with ANER[44], to show that our model is best performing with our dataset.

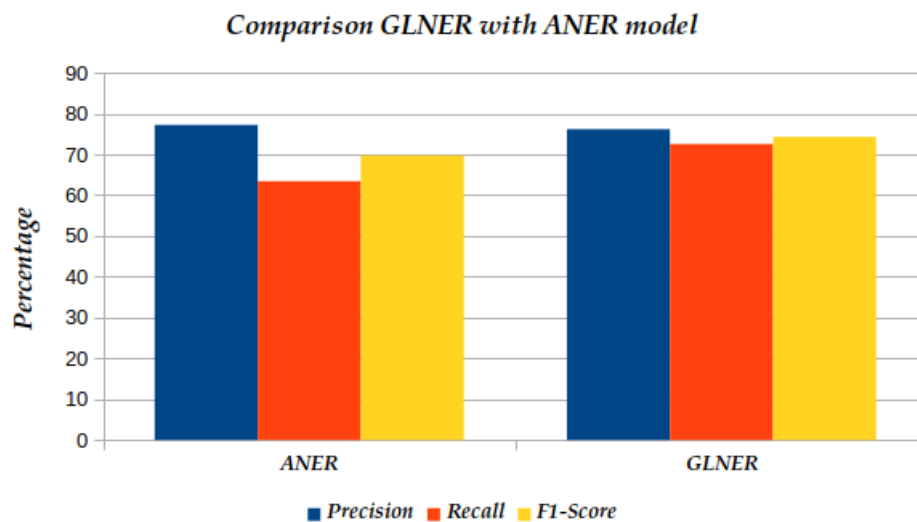


FIGURE 5.6: The precision, recall, and F1-Score of ANER and GLNER model.

In ANER, they found average precision (77.2%) and recall (63.4%), for an overall 69.7% F1-score and we found precision 76.15%, recall 72.49% and F1-Score 74.28%.

# Chapter 6

## Conclusion and Recommendation

This chapter is conclusion of the study based on the expemental analysis. It also contains recomandation to show further researches that can be done in the future.

### 6.1 Conclusion

This paper has proposed a model for Gamo Language named entity recognition and classification, an under-resourced language. In doing so, we assessed the structure of GL and the linguistic patterns that refer to Nes.

Gamo is a strict final verb (SOV) language. The object appears immediately before the verb (SOV). When the verb is ditransitive, the order between the direct object and the indirect object is free. Gamo language nouns are words used to name or identify any of a class of things, people, and places or ideas.

Gamo language nouns can be classified in to tangible (countable nouns), intangible (non-countable), collective nouns, common nouns and proper nouns. The main focus of this research is proper nouns and Common nouns.

We have reviewed related studies that are conducted so far in different languages, this consists of professional journals, formal research reports, university-affiliated bulletins, reports, monographs, and NER systems developed for other languages.

---

Two models have been developed: the first model consists of word-level embedding for data representation using skip-gram, context encoding using BiLSTM, and tag decoding using CRF. The second model is consists of Character and word-level data representation using CNN and word2vec(skip-gram) respectively, context encoding using BiLSTM, and tag decoding using CRF.

We used python programming language to develop a prototype for the Gamo language Named Entity recognition(GLNER) model. Packages and libraries used are gensim, pandas, scipy, sklearn, NLTK, numPy, and pyTorch.

As a baseline research, we found promesing result. According to the exprment, we conclude that named entity recognition with charater and word level embedding outperforms, named entity rocogintion with word level embedding alone for Gamo language text. Our model GLNER achieved accuracy 94.59%, precision 76.15%, recall 72.49%, and F1-Score 74.28%.

## 6.2 Recommendation

This research demonstrated features that extracted from character and word level embedding are used for NER task. It also shows that high performance NER system can be build from character and word embedding features. But to have full fledged GLNER further researches could be conducted. Based on our observations, we recommend the following future research areas. When conducting our experiments we have used small amount of corpus for representing data, and for training, validation and testing. So, to get better result the amount of data need to be large. Inaddition to the size of dataset there are number of studies that need to be done such as:

1. Part of speech tagging task
2. Our research focused on only basic named entity classes (Person,organization, location, and MISC), further researches could consider additional categories of named entities to have complete GLNER system.
3. Data leak prevention through named entity recognition
4. Question and answering
5. Machine translation



# Bibliography

- [1] R. Sharnagat, “Named entity recognition: A literature survey,” Center For Indian Language Technology, 2014.
- [2] R. Grishman and B. Sundheim, “Message understanding conference-6: A brief history,” in Proc. COLING, vol. 1, 1996.
- [3] E. F. Tjong Kim Sang and F. De Meulder, “Introduction to the conll-2003 shared task: Language-independent named entity recognition,” in Proc. NAACL-HLT, 2003, pp. 142-147.
- [4] Ling, X. and Weld, D.S., 2012, July. Fine-grained entity recognition. In Twenty-Sixth AAAI Conference on Artificial Intelligence.
- [5] L. d. Corro, A. Abujabal, R. Gemulla, and G. Weikum, “Finet: Context-aware fine-grained named entity typing,” in Proc. EMNLP, 2015, pp. 868–878.
- [6] Balog, K., 2018. Entity-oriented search (p. 351). Springer Nature.
- [7] J. Guo, G. Xu, X. Cheng, and H. Li, “Named entity recognition in query,” in Proc. SIGIR, 2009, pp. 267–274.
- [8] H. Raviv, O. Kurland, and D. Carmel, “Document retrieval using entity-based language models,” in Proc. SIGIR, 2016, pp. 65–74.
- [9] D. Nadeau and S. Sekine, “A survey of named entity recognition and classification,” *Linguisticae Investigationes*, vol. 30, no. 1, pp. 3–26, 2007.
- [10] V. Yadav and S. Bethard, “A survey on recent advances in named entity recognition from deep learning models,” in Proc. COLING, 2018, pp. 2145–2158.

- 
- [11] Marrero, M., Urbano, J., Sánchez-Cuadrado, S., Morato, J. and Gómez-Berbís, J.M., Computer Standards Interfaces, 5,482–489, Named Entity Recognition: Fallacies, Challenges and Opportunities, 35, 2013.
- [12] Rong, X. (2014). word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.
- [13] Babych, B. and Hartley, A., 2003. Improving machine translation quality with automatic named entity recognition. In Proceedings of the 7th International EAMT workshop on MT and other language technology tools, Improving MT through other language technology tools, Resource and tools for building MT at EACL 2003.
- [14] Oudah, M. and Shaalan, K., 2012, December. A pipeline Arabic named entity recognition using a hybrid approach. In Proceedings of COLING 2012 (pp. 2159-2176).
- [15] Hassel, M., 2003. Exploitation of named entities in automatic text summarization for swedish. In NODALIDA'03–14th Nordic Conference on Computational Linguistics, Reykjavik, Iceland, May 30–31 2003 (p. 9).
- [16] Nadeau, D. and Sekine, S., 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1), pp.3-26.
- [17] Kaur, D. and Verma, A., 2014. Survey on Name Entity Recognition Used Machine Learning Algorithm. (IJCSIT) *International Journal of Computer Science and Information Technologies*, 5(4), pp.5875-5879.
- [18] Mansouri, A., Affendey, L.S. and Mamat, A., 2008. Named entity recognition approaches. *International Journal of Computer Science and Network Security*, 8(2), pp.339-344.
- [19] Li, J., Sun, A., Han, J. and Li, C., 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*.
- [20] Tkachenko, M. and Simanovsky, A., 2012, September. Named entity recognition: Exploring features. In KONVENS (pp. 118-127).

- 
- [21] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J., 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).
- [22] Athavale, V., Bharadwaj, S., Pamecha, M., Prabhu, A. and Shrivastava, M., 2016. Towards deep learning in hindi ner: An approach to tackle the labelled data scarcity. arXiv preprint arXiv:1610.09756.
- [23] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *nature*, 521(7553), pp.436-444.
- [24] Britz, D., Goldie, A., Luong, M.T. and Le, Q., 2017. Massive exploration of neural machine translation architectures. arXiv preprint arXiv:1703.03906.
- [25] Chiu, J.P. and Nichols, E., 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4, pp.357-370.
- [26] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.
- [27] Kumar, A. and Sebastian, T.M., 2012. Sentiment analysis: A perspective on its past, present and future. *International Journal of Intelligent Systems and Applications*, 4(10), pp.1-14.
- [28] WONDIMU GAGA GASHE. (2010). Sociolinguistic facts about the Gamo area, South Ethiopia. Addis Ababa, ARCCIKCL.
- [29] Král, P., 2011. Features for named entity recognition in czech.
- [30] colah's blog. Understanding LSTM Networks. <http://colah.github.io/posts/2015-08-Understanding-LSTMs/> . Accessed march 4, 2020.
- [31] Ekbal, A., Haque, R. and Bandyopadhyay, S., 2008. Named entity recognition in Bengali: A conditional random field approach. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.
- [32] Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

- 
- [33] Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural computation*, 9(8), pp.1735-1780.
- [34] Graves, A., Mohamed, A.R. and Hinton, G., 2013, May. Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 6645-6649). IEEE.
- [35] Bender, M.L., 2003. Northeast Africa: a case study in genetic and areal linguistics. In Ms (based upon a presentation at the International Workshop on ‘The connection between areal diffusion and the genetic model of language relationship’, held at the Research Centre for Linguistic Typology, Australian National University, 17-22 August 1998).
- [36] Thomassen, A.L., 2015. Contributions to the description of the phonology of the Bonke variety of Gamo (Master’s thesis).
- [37] Taylor, Nicholas (1994). “Gamo Syntax”. PhD thesis. London: University of London.
- Theil, Rolf (2006). *Is Omotic Afroasiatic? A Critical Discussion*. Accessed:
- [38] Hirut Woldemariam (2004). “Gamo: a dialect variant or a group with its own dialects?” In: *Cushitic-Omotic Studies*. Ed. by Yoichi Tsuge. Kanazawa: Kanazawa University, pp. 67–78.
- [39] Hayward, R.J. and Chabo, E., 2014. *Gamo-English-Amharic Dictionary: With an Introductory Grammar of Gamo*. Harrassowitz Verlag.
- [40] Ahmed, M., 2010. *Named Entity Recognition for Amharic Language* (Doctoral dissertation, Addis Ababa University).
- [41] Alemu, B., 2013. *A named entity recognition for Amharic* (Doctoral dissertation, Addis Ababa University).
- [42] Tadele, M., 2014. *Amharic Named Entity Recognition Using A Hybrid Approach* (Doctoral dissertation, Addis Ababa University).
- [43] Dagimawi, D., 2017. *Amharic Named Entity Recognition Using Neural Word Embedding as a Feature* (Doctoral dissertation, AAU).

- 
- [44] Gambäck B, Sikdar UK. Named entity recognition for Amharic using deep learning. In 2017 IST-Africa Week Conference (IST-Africa) 2017 May 30 (pp. 1-8). IEEE.
- [45] Aboaga, M. and Ab Aziz, M.J., 2013. Arabic person names recognition by using a rule based approach. *Journal of Computer Science*, 9(7), p.922.
- [46] Rong, X., 2014. word2vec parameter learning explained. arXiv preprint arXiv:1411.2738.
- [47] Kejela, M.L., 2010. Named Entity Recognition for Afan Oromo. *Computer Science*, Addis Ababa University, Addis Ababa.
- [48] Sani, A., 2015. Afaan Oromo Named Entity Recognition Using Hybrid Approach (Doctoral dissertation, Addis Ababa University).
- [49] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semisupervised sequence tagging with bidirectional language models," in *Proc. ACL*, 2017, pp. 1756–1765.
- [50] M. L. Patawar and M. Potey, "Approaches to named entity recognition: a survey," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 3, no. 12, pp. 12 201–12 208, 2015.
- [51] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. NAACL*, 2016, pp. 260–270.
- [52] Balog, K., 2018. Entity-oriented search (p. 351). Springer Nature.
- [53] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna, "The query-flow graph: model and applications," in *Proc. CIKM*, 2008, pp. 609–618.
- [54] G. Saldanha, O. Biran, K. McKeown, and A. Gliozzo, "An entity-focused approach to generating company descriptions," in *Proc. ACL*, vol. 2, 2016, pp. 243–248.
- [55] Zhang, S. and Elhadad, N., 2013. Unsupervised biomedical named entity recognition: Experiments with clinical and biological texts. *Journal of biomedical informatics*, 46(6), pp.1088-1098.

- 
- [56] Collins, M. and Singer, Y., 1999. Unsupervised models for named entity classification. In 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.
- [57] G. Zhou and J. Su, “Named entity recognition using an hmmbased chunk tagger,” in Proc. ACL, 2002, pp. 473–480.
- [58] B. Settles, “Biomedical named entity recognition using conditional random fields and rich feature sets,” in Proc. ACL, 2004, pp. 104–107.
- [59] A. P. Quimbaya, A. S. Múnera, R. A. G. Rivera, J. C. D. Rodríguez, O. M. M. Velandia, A. A. G. Peña, and C. Labbé, “Named entity recognition over electronic health records through a combined dictionary-based approach,” *Procedia Computer Science*, vol. 100, pp. 55–61, 2016.
- [60] Y. Ravin and N. Wacholder, *Extracting names from natural-language text*. IBM Research Report RC 2033, 1997.
- [61] Y. Li, K. Bontcheva, and H. Cunningham, “Svm based learning system for information extraction,” in *Deterministic and statistical methods in machine learning*. Springer, 2005, pp. 319–339.
- [62] Yao, L., Liu, H., Liu, Y., Li, X. and Anwar, M.W., 2015. Biomedical named entity recognition based on deep neural network. *Int. J. Hybrid Inf. Technol*, 8(8), pp.279-288.
- [63] Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C., 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.
- [64] Brownlee, J., 2017. *Deep Learning for Natural Language Processing: Develop Deep Learning Models for your Natural Language Problems*. Machine Learning Mastery.
- [65] Li, B., Drozd, A., Liu, T. and Du, X., 2018, June. Subword-level composition functions for learning word embeddings. In *Proceedings of the second workshop on subword/character level models* (pp. 38-48).
- [66] Hemker, K., 2018. *Data augmentation and deep learning for hate speech detection* (Doctoral dissertation, Master’s thesis, Imperial College London).

- 
- [67] Kiryakov, A., Popov, B., Terziev, I., Manov, D. and Ognyanoff, D., 2004. Semantic annotation, indexing, and retrieval. *Journal of Web Semantics*, 2(1), pp.49-79. Lin, C.C., Ammar, W., Dyer, C. and Levin, L., 2015.
- [68] Unsupervised POS induction with word embeddings. arXiv preprint arXiv:1503.06760.
- [69] Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, pp.135-146.