
**Hawassa University**
**Institute of Technology**
**School of Computing and Informatics**
**Department of Computer Science**

# Developing Koorete Part of Speech (POS) Tagger: an Empirical Evaluation of Neural Word Embedding and N-Gram Based Statistical Approaches

**By: Agegnehu Ashenafi PGCSR/003/10**

**Advisor: Dr. Tesfaye Bayou (PhD)**

**August 12, 2021**

**Hawassa University, Hawassa, Ethiopia**

## Declaration

I hereby declare that this MSc. Thesis in **Computer Science** is my original work and has not been presented for a **Master's of Science** degree in any other university, and all sources of material used for this thesis have been duly acknowledged.

Name: Agegnehu Ashenafi

Place: Hawassa University

Date of Submission: August 12, 2021

Signature: _____

# SCHOOL OF POST GRADUATE STUDIES

# HAWASSA UNIVERSITY

# ADVISORS' APPROVAL SHEET

This is to certify that the thesis entitled "**Developing Koorete Part of Speech (POS) Tagger: an Empirical Evaluation of Neural Word Embedding and N-gram based Statistical Approaches**" submitted in partial fulfillment of the requirements for the degree of master`s, the **Post Graduate Program of the Department of Computer Science**, and has been carried out by **Agegnehu Ashenafi** ID.No PGCSR/003/10, under our supervision. Therefore, we recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

| _____ | _____ | _____ |
|---|---|---|
| Name of Major Advisor | Signature | Date |
| _____ | _____ | _____ |
| Name of Co-advisor | Signature | Date |

**HAWASSA UNIVERSITY**

**SCHOOL OF POST GRADUATE STUDIES**

**EXAMINERS' APPROVAL SHEET**

As members of the Board of Examiners of the final Masters of Science degree open defense, we certify that we have read and evaluated the thesis prepared by **Agegnehu Ashenafi** under the title "**Developing Koorete Part of Speech (POS) Tagger: an Empirical Evaluation of Neural Word Embedding and N-gram based Statistical Approaches**" and recommend that it is accepted as fulfilling the thesis requirement for the degree of **Masters Science in Computer Science**.

| Name of Chair-Person | Signature | Date |
|---|---|---|
| Name of Internal Examiner | Signature | Date |
| *Wondwossen Mulugeta (PhD)* | | *August 09, 2021* |
| Name of External examiner | Signature | Date |
| SGS Approval | Signature | Date |

Final approval and acceptance of the thesis is contingent upon the submission of the final copy of the thesis to the SGS through the DGC/SGC of the candidate's department/School.

Thesis approved by

| DGC/SGC | Signature | Date |
|---|---|---|

Certification of the Final Thesis

I hereby certify that all the corrections and recommendations suggested by the Board of Examiners are incorporated into the final Thesis of this study.

| Name of Researcher | Signature | Date |
|---|---|---|
| Department | Signature | Date |

# Acknowledgements

First, all praise and thanks be to The Almighty God for His graces and guidance so as helping me to conduct this thesis.

Next, I would like to express my heartfelt gratitude and deepest thanks kindly to Dr. Tesfaye Bayu (PhD) for his continuous encouragement, assistance and valuable advice throughout this proposal for Master's program. He kept motivating and supporting me voluntarily accepting my request for help.

Besides, I forward my great thanks to Dr. Samuel Zinabu and Tizita Fabrica, who helped me by giving an expert advice and in composing and editing Koorete language corpus. Most of the translations from Amharic written book 'Diccho' to Koorete is done through Tizita Fabrica, who was a GC student at Dilla College of Teachers Education in 2012 E.C.

Finally, I forward my warm thanks to the department head, Instructor Daniel Tesfaye(MSc.) in that he helped me not to be reluctant at doing my work, and coordinating the academics in IOT registrar.

# Abstract

The Koorete language is spoken by the Koore people in Amaro Kele Special Woreda and in four Kebeles of Burji Special Woreda, Southern regional state. Koorete is written with Latin alphabets (or called '*Diizo Beyta'* in Koorete language). This means, Latin alphabet is adopted to the language by adding additional combinations of letters for peculiar sounds totaling to 31-consonants ('*Artaxita*' in Koorete), 5-vowels ('*Arxaxita*' in Koorete), and one more symbol. The syntax of Koorete sentence structure is **"**Subject (*Zeere utaade*) + Object (*efaxe*) + Verb (*Hanta beyiisaxe*)**".**

This study develops Koorete POS Tagger using the empirical evaluation of Neural Word Embedding and N-gram based statistical POS tagging approaches. Parts-of-speech (POS) tagging is the process of assigning part-of-speech labels/tags to each word from Koorete POS tagset. Neural word embedding are distributed representations of words into vectors applying Bi-LSTM RNN model. N-gram based statistical approach uses probability frequencies of sequence labeling of words from the KPT corpus. Words having similar meanings can be represented similarly, which enable deep learning methods. The behavior of having similar representation orients to the reduction of out-of-vocabulary impact. This means, binary vector |V| dimension reduction. In simple language, word embedding is a language modeling technique which maps words to vectors using Word2Vec package, and would be computed in RNN. This Word2Vec package converts words to arrays of real numbers, and concatenate the original corpus word categories to the generated vectors. Word2Vec has a capability of capturing context of a word (semantic and syntactic similarity) in a document in relation with other words.

For the purpose of sequence labeling method and distributed representation, this study uses Bi-LSTM RNN by achieving the state-of-the-art POS tagging accuracy and N-gram based statistics approaches in contrast to the more classic approaches. Bi-LSTM handles or adds letter case functions to keep the original letter case information of word. This study applies skip-gram algorithm to encode words into a limited vector space. Because skip-gram model is efficient method for learning high quality vector representations of words from large amounts of unstructured text data.

So experiments were practiced on Bi-LSTM RNN model, and N-gram tagger statistical approach. For this, KPT corpus is used about size of 1718 sentences (33220 words), and then divided this corpus into 90% training data and 10% testing data. The experiment on Bi-LSTM RNN word embedding POS tagging approach did better than the N-gram statistical POS tagging approach with the accuracy of 98.53%.

Hence, this study solves the problems of (1) no rich resource in NLP applications, (2) Koorete language not having its own KPT corpus and tagsets for NLP applications, (3) the state-of-the-art tagging performance algorithms accuracy with other relative languages POS tagging model.

# Table of Contents

## List of Figures

# List of Tables

## List of Abbreviations

NLP - Natural Language Processing

CRF - Conditional Random Field

HMM - Hidden Markov Model

MEMM - Maximum Entropy Markov Model

NLTK - Natural language toolkit

POS - Part-of-Speech

RNN - Recurrent Neural Network

Bi-LSTM - Bidirectional Long Short-Term Memory

LSTM - Long Short-Term Memory

CNN - Convolutional Neural Network

DNN - Deep neural network

CBOW - Continuous bag of words

KPT - Koorete POS Tagger

|V| - Vocabulary dimension

RQ - Research Question

SMS - Short message services

TBL - Transformer based learning

NER - Named entity recognition

OOV - Out of Vocabulary

# Chapter 1. Introduction

## 1.1 Background

These days, people are highly interacting to each other within a fraction of time as the world is connected to one village through the Internet. The interaction needs to have knowledge how to attractively communicate with its customer in a formal coordinated way. It means humans use a natural language of its mother tongues, and second or third trained languages in order to express ones idea and its culture. So the way of expressing a natural language to represent idea is termed as a natural language processing (NLP). NLP could be represented at different levels such as word level, phrase level, sentence level or semantic level. The language processing needs an improvement in documentation status for information per the advancing technology [1]. Knowing ones language processing starts from knowing its part-of-speech; to which word-class the word labeled/tagged; its syntactic and semantic values, and the phrase and sentence contexts [8, pp. 1]. Actually, computers cannot understand human language syntax and its semantics in a sentence. It is becoming difficult for humans to analyze and get the necessary contents computerized because of the data requirement of each natural language is increasing. Therefore, we need to develop a computer application to adapt a machine language with respect to natural language, and then a computer can understand a natural language as human understands. So the Koorete language has to be developed in technology aspects. For its contribution share, this proposed study is developing Koorete language part of speech tagging (POS) representation on neural word level embedding and N-gram based approach with a corpus named 'KPT Corpus'.

These days, NLP is a recent field of study that has many applications including parts-of-speech (POS) tagging, named entity recognition (NER), machine translation (MT), information retrieval (IR), information extraction (IE), morphological analysis, syntactic parsing, stemming. So, one of the basic tasks in NLP is part-of-speech (POS) tagging. Parts-of-speech (POS) tagging is the process of assigning part-of-speech labels/tags like noun, verb, pronoun, preposition, adverb, adjective or other lexical class markers to each word in a text [9]. POS tagging is not useful by itself but it is generally accepted to be the first step for understanding a natural language [7]. POS tagging is a precondition or subcomponent for most of NLP tasks rather than being an application to stand by itself. Most other tasks and high-level NLP applications heavily depend on it [8]. So POS tagging is required for many high-level NLP applications, such as word sense disambiguation, syntactic parsing, information extraction, named entity recognition, machine translation, information retrieval, spell checker and stemmer.

Different researchers have been using different learning approaches to automate ones language POS tagging tasks. The POS tagging computerizing task might be simplified in order to give a solution to the target language problem with respect to its appropriate language model can be solved using three basic learning approaches. These are rule-based learning approach, statistical (or stochastic or probabilistic or corpus-based) learning approach, and hybrid learning approach. (1) The rule-based POS tagging approach could be developed based on kinds of

information such as dictionaries, lexicons, hand-coded grammatical rules [36][7]. It assigns all possible tags to the words which are only in the vocabulary bound, and so gives more accurate results. But it is rigid to the rules, and not flexible to determine the word category when new featured words are appeared. It does not predict out-of-the-box of vocabulary created. That is a word may belong to more than one category. (2) The corpus-based POS tagging approach chooses most frequent tag from the featured pattern in training corpus for each word (frequency based probability of a word co-occurrence with respect to its neighboring words), and performs its prediction based on the collected corpus vocabulary |V| tagsets to give an appropriate word class to the given words. Effective process of prediction depends on the training and testing performance results to evaluate its language model realization. (3) The hybrid learning approach combines the positive aspects of the above-mentioned learning approaches ((1) and (2)).

The learning approaches which are taken to be implemented in this study are categorized to corpus-based or statistical based learning approach. There was not any developed Koorete language corpus before [1]. For this, and other reasons presented on the narration part of point (3.1.), the empirical evaluation of N-gram statistical learning approaches and neural word embedding with deep learning word vector representation concept are chosen. Neural network and deep learning POS tagging approach does not require feature engineering i.e., this approach learns directly from original data. So, it is often open for new words because it uses neural networks to predict the words with similar context like neurons of human brain inspired.

For training, instead of dictionary, neural word embedding uses word vector lookup table of Bi-LSTM RNN at hidden layer, and then POS tagging is initialized with the trained word embedding [2]. Bi-LSTM RNN can then benefit from word embedding trained on large unlabeled corpus, and larger training corpus leads to a better performance. For example, '***harre'*** in Kooreete language is both noun and verb. This ambiguity could be removed learning from the context of words syntactic and semantic meaning in vector space. POS Tag set is the set of tags from which the POS tagger is supposed to choose tags and attach them to the relevant word. Every POS tagging will be given a standard tag sets. The tag set may be coarse such as NN(Noun), VB(Verb), ADJ(Adjective), ADV(Adverb), PP(Preposition), and CC(Conjunction) and so on [7][10, pp.5].

### *1.1.1 Neural Word Embedding*

Most works use words as the *smallest units* in the compositional architecture, often using pre-trained word embedding or learning them specifically for the task of interest [3, pp.1]. Word embedding is one of the most popular representations of document vocabulary |V| to word vectors (Word2Vec) as inputs. It is the unified name for a set of language modeling and feature learning techniques in NLP, where words or phrases from the vocabulary are mapped to vectors of real numbers. It transforms human language meaningfully into a numerical form. This means word vectors are simply arrays of numbers that represent the meaning of a word with respect to its index position given in the vocabulary. Word2Vec is a dense two-layer neural network. representation. Why Word2vec? Because it (1) preserves relationships between words, (2) gives better results in lots of deep learning applications, (3) deals with addition of new words in the vocabulary.

Neural word embedding is simply a process of word embedding using neural network for a current word based on given surrounding words. It is a distributed representation of a text which allows deep learning methods to perform well on the challenging NLP problems. This embedding enables words having similar meanings (similar context) can be represented similarly (put in same vector space). This is one of the basic advantages in the reduction of out-of-vocabulary impact [8, pp.15]. This is possible because words will not be completely unknown as far as they have feature vectors even if they may not be seen in the training dataset. Because of the use of neural networks, this embedding method generates the Word2Vec mappings since a word is the underlying input representation.

### *1.1.2 Deep Learning*

The term 'deep' refers to the number N-to-N learning of hidden layers in the neural network. Deep learning technique is a class of machine learning which performs much better on unstructured data. It is outperforming current machine learning, and tackles problems on which shallow architectures (e.g.word2vec) are affected by the curse of dimensionality [14]. It enables computational models to learn features and layered model inputs progressively from data at multiple level abstractions [13]. Deep neural networks are successful in Supervised learning, Unsupervised learning, Reinforcement learning, hierarchical learning, as well as hybrid learning. The algorithms in deep learning use the back propagation algorithm that discovers complex structures from large datasets. Back propagation solves an optimization problem using a gradient-based calculation method function for each and every iterations in Bi-LSTM RNN model [13].

Deep neural network architectures include recurrent neural networks (NN), convolutional neural networks (CNN) and deep neural networks (DNN) that are used in different areas including NLP applications. From the possible architectures of DNNs, RNNs having LSTMs are used in this study. The better performance of about 97.40% accuracy is observed when Peilu Wang et al. applied on English with word embedding as features [2]. The architectures of this study approach is tested on Koorete language. Why Deep Learning? – Because it gains supremacy in terms of accuracy when trained with large amount of data.

### *1.1.3 POS Tagging Applications*

Advanced high-level NLP applications require POS tagging as a significant pre-requisite subcomponent in areas of NLP applications to identify the token word category for determining its morphology type, pronunciation, and context semantics [22]. So POS tagging is such an important bed rock for a researchers who like to conduct the following high-level NLP tasks (such as word sense disambiguation, NER, MT, IR, IE, syntactic parsing, etc.) to develop ones language resource and to train the machine of these applications and other related aspects.

**Named entity recognition** (NER): as POS tagging is assigning tags to a word class, NER's task is classifying proper nouns in a given text into their corresponding predefined categories. NER

recognizes proper nouns of real world objects locations, name of persons, organization, name of product, chemical entities, biomedical entities based on the tags given to them[4][8]. This new way helps to structure, access, and also enrich information.

**Machine translation** (MT): as POS tagging is assigning tags to a word class, MT converts a text from a source language to a target language [4] based on the tags given to them preserving the original meaning and its some properties using software without human intervention. During translation process either human or automated must know the word class. Knowing the meaning of a text in the original language is the second job in translation and must be fully restored in the target language [8]. Human and MT each has their share of challenges because it uses a bilingual dataset to do the translation task. Meaning it is not only a word-for-word substitution but it must interpret and analyze all of the elements in the text and know how each word may influence another. So it needs the quality of translation based on POS.

**Information retrieval** (IR) As IR refers to the process, methods, and procedures of searching, locating, retrieving, and obtaining recorded data on an unstructured nature that are relevant to information based on the POS and index of those resources [8]. It is searching for information out of a database of organization for evaluation of information.

**Information extraction** (IE): POS tagging is one of the pre-requisite for the process of IE. It extracts useful hidden entities information such as names, places, events, dates, times and prices which is relevant to a user's needs [4]. It automatically extracts structured information from unstructured machine-readable documents based on linguistic rules and statistics.

**Morphological analysis** (MA) is a method for identifying, structuring and investigating the input word from the total set of possible relationships of POS based on the base words. And then it detects and segments the various morphological features [8] in the given word and provides a morphological representation to the word. Its objective is to break down analysis problem at hand into its essential and place them in a multi-dimensional matrix. Then find new ideas by searching the matrix for creative and useful combinations.

**Syntactic parsing** refers to finding syntactic structural relationships between words in a sentence based on the POS of sentences. This means finding the grammatical arrangement of words in a sentence and their relationship with each other sketched as a tree. It processes the data through lexical analysis and obtains the internal structure of sentences in natural languages so as to contribute ordering and dependency connotation [4].

## 1.2 Statement of the Problem

As POS tagging is a precondition for most of NLP applications, it uses several sources of information such as dictionaries, lexicons, rules, and so on. It means a word may belong to more than one category in a dictionary. Named entity recognition, word sense disambiguation and syntactic parsing are some of the advanced high-level NLP applications [4]. For developing the high-level NLP applications, POS tagging is a basic task to be implemented practically and applied feasibly to the real world communication extension. For example in Word sense disambiguation (WSD), WSD does the task of understanding the correct sense or meaning of a word in a given context; whereas knowing POS tag in WSD helps for ordering sentence structure and proposing the next word. Other POS tagging usage in high-level NLP applications are discussed in the aforementioned section (1.1.3.).

Koorete needs to have a contribution of POS tagging in high-level NLP applications to enhance language resources. But to the extent of my exploration, no studies have been conducted on Koorete language POS tagging [1]. This shows that Koorete language has no good documentation status meaning it is not rich in language resources, which need to be applied in advanced NLP applications linguistic study [1]. Therefore, it is under resource.

Koorete is being delivered as a subject for Elementary students; however, there is no literature. This may be a crucial new research area for some NGOs have to intervene [1]. Despite the fact, there is the morphology of Koorete conducted especially on Verb morphology (Beletu, 2003). Also I have interviewed a Kooreete language expert from Dilla Teachers college of education, where there are many Kooreete language instructors; but there has not been any Koorete POS tagging done yet.

Koorete POS tagging also need to reach the state-of-the-art tagging performance of the other languages POS tagging mode[2]. For the reason this study uses neural word embedding model. This model takes into action Bi-LSTM RNN in sequence labeling to predict a tagging probability distribution of each word. And it achieves performance to the extent of prediction in the language modeling, language understanding and machine translation without using the morphological features [2][3].

The other problem can be absence of already prepared Corpus on the Koorete language for POS tagging. For POS tagging, any language has to have its own language corpus as a dataset. Corpus-based POS taggers build models from the training dataset using one or more algorithms and apply the models on unseen instances of the language [8].

## 1.3 Research Questions

The research questions, which are raised on this study in order to implement theirs prototype with a convenient and enough dataset, are here as follows.

**RQ1**: How better could Bi-directional LSTM RNN based neural word embedding POS tagging approach performs with respect to N-Gram Based Statistical POS tagging approach for Koorete language?

**[***How could neural word embedding simplify Koorete POS tagging relative to Ngram based statistical approach?***]**

**RQ2**: What are the most appropriate tagsets need to be used for Koorete POS tagging?

**[***Adapt appropriate tagsets***]**

## 1.4 Objective

### *1.4.1 General objective*

The general objective of the study is to evaluate two known POS tagging approaches (neural word embedding and N-gram based statistical) in the development of Koorete language POS tagger.

### *1.4.2 Specific objective*

The specific objectives of the study are the followings:

1. To have tagged Koorete language corpus.
2. To have adapted tagsets which are appropriate with Koorete
3. To have automatically generated word features in real numbers
4. To have a trained values of neural word embedding and N-gram based statistical POS tagging models
5. To evaluate the performance of the two POS tagging models, and then determine appropriate approach for Koorete

## 1.5 Significance of the Study

The purpose of this study is mainly for high level NLP application researchers in addition to making it to be rich in documentation resources. These can be:

- It limits the range of following words and range of meaning
- It adds documentation status richness on Kooreete language literature
- It helps to develop the language from spoken to written in linguistic material
- It helps a researcher who conduct on Specific parser, word sense disambiguation, named entity recognition, machine translation

## 1.6 Scope and Limitation of the study

### *1.6.1 Scope*

The Kooreete Language POS Tagger using neural word embedding relative to N-gram tagger model brings forth the following findings on Koorete POS corpus as a contribution:

- Identify techniques of the state-of-the-art POS tagging accuracy
- Develop the Koorete POS Corpus
- Design an algorithm to build the model on both stochastic and neural approaches
- Implement the model
- Evaluate the model

### *1.6.2 Limitation*

Because morphological features are discrete, so it has to be easily represented as one-hot vector. But using this type of rich features leads to too large input layer, which is difficult to maintain and update the words original root or stem. Reach feature refers to a number of morphologies being produced from the lexical or root words. Therefore, this study avoids using such rich features. Instead, it applies word level representation without morphological consideration with simple letter case functions. This means, this study involves word embedding in a low dimensional real-valued vector to represent word [2]. This is considered as containing part of syntactic and semantic information for various language processing tasks.

Even though POS Tagger will be developed, the study does not perform the following areas.

- The Kooreete language structure order is "Subject + Object + Verb". But this study modeling tool does not train the order of the language to the machine.
- It does not consider morphological features.

## 1.7 Methodology

This research study has presented its methodology with a detailed discussion about the procedures, methods, materials and tools under Chapter **4** of this document.

## 1.8 Thesis organization

As this is Chapter **1**, the remaining chapters of this study can be organized according to the following partitions. Chapter **2** presents the overview of Koorete Language. Chapter **3** presents the literature review and explains the theoretical concepts on different POS Tagging approaches. Chapter **4** presents methodology such as materials, methods. Chapter **5** presents experimental results and its discussion. Chapter **6** presents conclusion with future enhancement of the study.

# Chapter 2. Overview of Koorete Language

## 2.1. Background of Koorete Language

Language is a communication tool in understanding of others' lives that considered as a light of mind and it represents all instruments, means, vehicles or modes of communication to express ones idea [19]. It is important for social interaction either formally and/or informally in articulated manner. So the language Koorete is the Koore people mother tongue language used to express their feelings by adding the suffix -te to the ethnonym Koore to get the language named Koorete. The Koore ethnic group homeland lies in the southern Ethiopia in the Amaro Special woreda, which is east of the Abaya and Chamo Lakes. The language of the Koore, which is called Koorete, belongs to East Ometo Language sub-family with Kachama-Ganjule and Zayse[18]. It is the only Ometo language east of Lake Abaya.

The Koorete language is vastly spoken by the Koore people and in four Kebeles of Burji Special Woreda, Southern Regional state, and in some western Gujji Oromia region by Koore people. Besides, Koorete is spoken by "about 60 Harro families in Harro village on Gidicho Island" in Lake Abbaya, and by people who emigrated from Amaro in a distant past [17, pg.2]. The language of Koorete is used for all social and administrative purposes and it is taught as a subject in elementary school as a medium of instruction [18].The administrative center of Koore is known as Kele, which is located in SNNP region at Amaro special Woreda [1] on a latitude and longitude of $06^0$05'N $38^0$02'E with an altitude of 1424 meters above sea level. Geographically, Amaro Special Woreda is positioned to east Gujii Oromia, to west Konso and Derashe, to north Gamo, to south Burji. Amaro Special Woreda is totally inhabited by Koore. However, there are other ethnic groups who came for government work and business, and these are limited to Kele Town. It has 350,000 speakers (Awoke, Koore People, 2020).

The Koorete language is written using Latin alphabets (or called '*Diizo Beyta'* in Koorete) about 37-letters, of those 26 are single letters, 10 are double letters, and one more is symbol. It has 31-consonants ('*Artaxita*' in Koorete) and 5-vowels ('*Arxaxita*' in Koorete). Koorete is a simple tone language in which a pitch enters into the lexical realization of rise and fall of voice during speaking [17]. The 37-letters classified into 22 Bidza artaxi xheyssita (single consonant sounds such as B,C,D,F,G,H,J,K,L,M,N,P,Q,R,S,T,V,W,X,Y,Z, '), 10 Dhaanqo artaxi xheyssita (double consonant sounds such as BH, CH, DH, DZ, JH, XH, SH, NY, TH, PH), 5 Arxaxi xheyssita (vowel sounds such as A,E,I,O,U). From the 37-letters, we have 15 Miima bidza artaxi xheyssita (stressed single consonant sounds such as NN, KK, TT, MM,YY, SS, DD, BB,WW, GG, RR, LL, CC, ", PP), 5 Miimma dhaanqo artaxi xheyssita (stressed double consonant sounds such as BBH, DDH, JJH, SSH, PPH), 5 Galala arxaxi xheyssita (long vowel sounds such as AA, EE, II, OO, UU), 4 Meddhona holi'a (naturally unstressed sounds such as F, H, Z, V), 3 Meddhona miima (naturally stressed sounds such as X, Q, J), 4 Meddhona miima dhaaniqita (naturally stressed double sounds such as DZ, XH, TH, CH), and 1 Meddhona holi'a dhaanqo (naturally unstressed double sound such as NY). In general, it contains *62* sounds i.e., 62 small (*Barse*) letters and 62 capital (*Anguse*) letters.

The syntax of Koorete sentence structure is **"**Subject (*Zeere utaade*) + Object (*efaxe*) + Verb (*Hanta beyiisaxe*)**".** Examples: Yede na'ii zawa keexho (That guy built a house).
It has Koorete dictionary called "A-Z Koorete Agric" functioning now. This dictionary translates a given word into three languages (Koorete←→English←→Amharic). Besides, Elementary education is delivered in Koorete language as medium of instruction, and, student text from 1st to 4th grades[28], and New Testament Bible is written using Koorete language. Scriptures published are such as New Testament Bible (2011), Portions (1999 — 2007), ክሉ ፍኩቾ(2009), FCBH(Bible Society of Ethiopia) [9], Koorete verb morphology[18], and Koorete segmental phonology[17]. However, the language documentation status is still not rich in language resources like Morphology Analyzer, Stemmer, Syntactic Parser, POS Tagger, Language exception handler, etc. applied in NLP linguistic study. This is why this study attempts to develop Koorete language POS (or 'Zeere Suma' in Koorete) tagging.

## 2.2. Koorete Phonetics

Phonetics describes the study of speech sounds and how their patterns are produced or articulated using the mental-oriented process [31]. For this process, words are stored and accepted in the mental lexicon with their extensive phonetic detail [17]. The smallest indivisible unit of sound of phonemes distinguishes one word from another word. As described in (2.1) Koorete has 31- consonant phonemes (B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Y, Z, BH, CH, DH, DZ, JH, XH, SH, NY, TH, PH), 5-vowel phonemes and one more symbol called pooxhe (ʾ). ROLF THEIL research [17] identified Koorete has the 23 consonant phonemes but he have not considered the 10-doubled consonant sounds; Beletu Redda research [28] presents by Hayward (1982) as this researcher identified 29 consonant phonemes for Koorete, while Ford (1990) identified 30 consonant phonemes.

*Koorete vowel phonemes*

| Monophthongs | | | | Diphthongs | | | |
|---|---|---|---|---|---|---|---|
| Short | | Long | | Short | | Long | |
| i | u | ii | uu | | ui | | uui |
| e | o | ee | oo | ei | oi | eei | |
| a | | aa | | ai | | aai | |

Table 2.1: a) Phonemes of vowels (picture credit: [17])

| | Front | Central | Back |
|---|---|---|---|
| **High** | i , ii | | u , uu |
| **Mid** | e , ee | | o , oo |
| **Low** | | a , aa | |

b) Phonemes of vowels (picture credit: [28])

Example:
word versus its pronunciation sound in Koorete 'five'-ʔítʃitʃe, grindstone-ʔílmaanʃu, 'dance' - dólle.

*The consonant phonemes of Koorete*

| Labial | Coronal | | Dorsal | Laryngeal |
|---|---|---|---|---|
| | Alveolar | Postalveolar | | |
| Non-sibilant | Sibilant | | Non-sibilant | |
| p | t | s ʃ | k | h |
| b | d | z ʒ | g | |
| ɓ | ɗ | s' ʃ' | k' | ʔ |
| m | n | | | |
| | r | | | |
| | l | | | |
| w | | | j | |

| | Bilabials | Alveolars | Palatals | Velars | Glottals |
|---|---|---|---|---|---|
| **Stops** | | | | | |
| **Vl** | | t | | k | ? |
| **Vd** | b | d | | g | |
| **Imp** | ɓ | ɗ | | | |
| **Ejective** | (p') | ( t') | | k' | |
| **Affricates** | | | | | |
| **Vl** | | ʦ | č | | |
| **Vd** | | dz | j | | |
| **Ejective** | | | č' | | |
| **Fricative** | | | | | |
| **Vl** | P | s | š | | h |
| **Vd** | | z | | | |
| **Ejective** | | s' | ž | | |
| **Nasals** | m | n | | ŋ | |
| **Lateral** | | l | | | |
| **Flap** | | r | | | |
| **Semi-vowels** | w | | y | | |

Table 2.1: c) Phonemes of consonants (picture credit: [17]) d) Phonemes of consonants (picture credit: [28])

Example:

word versus its pronunciation sound in Koorete 'money'-**míʒe**, 'penis'-**ɓàwwa**, breast-**ɗánse**

## 2.3. Koorete Morphology

Koorete has several different and distinctive morphological feature forms for nouns, pronouns, adjectives, verbs, adverbs, postposition and numerals. In Koorete, all nominal forms terminate in a vowel such as **/a/, /e/** and **/o/.** These terminal vowels formation are often deleted when another morphological feature is suffixed to the noun.

| **Class of terminal vowel /a/** | | **Class of terminal vowel /e/** | | **Class of terminal vowel /o/** | |
|---|---|---|---|---|---|
| maata | 'grass' | toke | 'leg' | kallo | 'stick' |
| tama | 'fire' | ts'oolinte | 'star' | k'ino | 'head' |

Koorete nouns are inflected to indicate different grammatical categories such as number, gender, definiteness and case.

Number Case: denotes the inflection of nouns for number, and formed by adding suffix (-ita) to nouns. Example: Abebey **harrita** zaliido (Abebe sold donkeys).

| **Singular** | **Gloss** | **Plural** | **Gloss** |
|---|---|---|---|
| grime | 'pig' | girm-**ita**~-ate | 'pigs' |
| harre | 'donkey' | harr-**ita**~-atse | 'donkeys' |

Definiteness: definite noun is expressed by suffixing the definite formative element **{-iyo}** to the base nouns. Example: Ziine yoodech girawiyo.

| **Nouns** | **Gloss** | **Definite Nouns** | **Gloss** |
|---|---|---|---|
| Woya | 'dove' | Woy-**iyo** | 'the dove' |
| Maydo | 'bull' | Mayd-**iyo** | 'the bull' |
| Girawe | 'cat' | Giraw-**iyo** | 'the cat' |

Locative Case: the noun that indicates the location of a person or a thing by suffixing the case marker **{-ka}** to the noun. Example: Buushe qaam'o utulmaka gusuwa.

| **Nouns** | **Gloss** | **Locative Nouns** | **Gloss** |
|---|---|---|---|
| Utulma | 'mortar' | Utulma-**ka** | 'in the mortar' |
| Daxha | 'loft' | Daxha-**ka** | 'on the loft' |

Ablative Case: a noun that is used as source of something or place of departure and formed by suffixing morpheme **{-afa}** to the nouns.
Example: Guliixhiafa zayte nu degesso.

| **Nouns** | **Gloss** | **Ablative Nouns** | **Gloss** |
|---|---|---|---|
| Guliixhe | 'fish' | Guliixhi**afa** | 'from fish' |
| Saamo | 'cabbage' | Saamo**afa** | 'from cabbage' |
| Amarro | 'Amarro' | Amarro**afa** | 'from Amarro' |

Commutative Case: the noun that indicates the notion of 'with' or 'accompanied by' and this case is expressed by the morpheme **{-ra}**. Example: Ade wontora kardiitofu.

| **Nouns** | **Gloss** | **Commutative Nouns** | **Gloss** |
|---|---|---|---|
| Boorsa | 'guest' | Boorsa-**ra** | 'with guest' |
| Wonto | 'God' | Wonto-**ra** | 'with God ' |

Instrumental Case: the noun that indicate the instrument with which an action is performed and the morpheme **{-na}** is commonly suffixed to the noun.
Example: Ye miixe kaliitena bursuwa.

| **Nouns** | **Gloss** | **Commutative Nouns** | **Gloss** |
|---|---|---|---|
| Kaliite | 'adz' | Kaliitena | 'with adz' |
| Kallo | 'stave' | Kallona | 'with stave' |

In Koorete, like nouns, pronouns have different types such as personal, possessive, demonstrative and interrogative pronouns.

Table 2.2: (a) The independent personal pronouns classification

| **Independent Personal Pronouns** | | | | | **Independent Object Pronouns** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Person | Sg.nominative | Gloss | Pl.nominative | Gloss | Person | Singular | gloss | Plural | Gloss |
| 1st per/sg | **ta-n-i** | 'I' | **nu-n-i** | 'We' | 1st per/sg | **ta** | 'me' | **numba** | 'us' |
| 2nd per/sg | **ne-n-i** | 'You' | **hi-nu-ni** | 'You' | 2nd M/F | **niya** | 'you' | **hinumba** | 'you' |
| 3rd sg/mas | **?es-i** | 'He' | **?us-i** | 'They' | 3rd mas | **?esa** | 'him' | **?uso** | 'them' |
| 3rd sg/fem | **?is-i** | 'She' | | | 3rd fem | **?iso** | 'her' | | |

Example: Zine **numba** xheyda na'i modhiikke. Iitanaa waxi **hinumba** lacco?

Table 2.2: (b) Dependent and Possessive Personal Pronouns

| **Dependent Personal Pronouns** | | | | | **Possessive Pronouns** | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Person | Singular | Gloss | Plural | Gloss | Person | Poss.Pron | Gloss | Possed.Noun | Gloss |
| 1st sg | **Ta** | I | **Nu** | We | 1st sg | **ta** | My | **ta** maydo | My ox |
| 2nd M/F | **Ne** | You | **Hi** | You | 2nd sg | **ne** | Your | **ne** maydo | Your ox |
| 3rd mas | **?e** | He | **?u** | They | 3rd M | **?e** | His | **?e** maydo | His ox |
| 3rd fem | **?i** | She | | | 3rd F | **?i** | Her | **?i** maydo | Her ox |
| **Possessive Pron. Indicate possessor and possessed Nominative pronoun** | | | | | 1st pl | **nu** | Our | **nu** maydo | Our ox |
| | Sg. Nom | Gloss | Pl. Nom | Gloss | 2nd pl | **hi** | Your | **hi** maydo | Your ox |
| 1st sg | **ta-s-i** | mine | **nu-s-i** | Ours | 3rd pl | **?u** | Their | **?u** maydo | Their ox |
| 2nd M/F | **ne-s-i** | yours | **hi-nu-s-i** | Yours | | | | | |
| 3rd mas | **?e-s-i** | his | **?u-su-s-i** | Theirs | | | | | |
| 3rd fem | **?i-s-i** | hers | | | | | | | |

There are also demonstrative pronouns which are used in Koorete language to indicate distance and proximity of objects or people from the speaker perspective such as **se?esa** – 'that', **ha?esa** – 'this'; besides, there are interrogative pronouns which are used in asking questions about place, person or things such as **?oone** – 'who', **?ayde** – 'when', **wayse** – 'how'.
Example: **Waysiwu ha?esa ne** eeto?(How did you like this?) **?oonewu hinuni**? (Who are you?)

In Koorete, morphological pattern production of verbs has three groups on the basis of their formal behavior from verb stems. These classifications are infinitive form, perfective form and imperfective form. Verbs do not change their base forms in the infinitive; imperfect and perfect verb stems.

| **Inf. Stem** | **Imperf. Stem** | **Perfect Stem** | **Gloss** |
|---|---|---|---|
| wod'- | wod'- | wod'd'- | 'kill' |
| bot- | bot- | bott- | 'forget' |
| žaš- | žaš- | žašš- | 'fear' |

The perfective stem expresses complete action by taking suffix {-O}, which is past tense marker example tuccuto**so**; the imperfective expresses actions that is not completed; the infinitive verb is the base form of a verb with a word 'to' in front of it. In Koorete, the simple past tense is based on the perfective aspect and is marked by the past tense marker **{-o},** which is attached next to the perfective form of the verb.

Past Continuous tense: a tense that indicates a continuous action based on the perfective aspect and it is expressed by combining the past counterpart **yeča** 'exist' (past) with the perfective verb stem that shows a period of time in the near past. Example: Tani gehiya-**ko** ta **yeca**.

Future tense: a tense that indicates a future action and it is expressed by combining the future morpheme (**-se)** to verb stem that shows a time in the future. Example: Zawa ta hango**se**.

In addition to the above word class morphological derivations, there are also Adjectives, Adverbs, Numerals and Postpositions morphological feature derivations in Koorete language [28].

## 2.4. Challenges in Koorete POS Tagging

Even though any language shapes thoughts and emotions determining one's perception of reality, we might also face challenges understanding ones language. The challenges in Koorete language could be as it is difficult to segment its morphology i.e., stemming is difficult, postpositions and conjunctions are attached to the root words either to the beginning, middle and/or ending, singular and plural forms has no a simple pattern, knowing ones article can either be definite article or indefinite article of different forms.

Morphological feature challenges could be the attaching of prefix, suffix, infix morphemes to the lexical word, and are somewhat difficult to segment its morphology feature from the base morpheme.

Example 1: Tani taa kuxorosa **worguwasso** (I am not seeking honor for myself). The word **worguwasso** has lexical item **worg**- meaning seek and –**uwasso** as suffix, which deviates the base word **worge** to logically negate the original meaning not to seek. Its stemming is also difficult in that it has no identical derivational forms.

Example 2: U **wordonikewo** e indo (He handed over as they wanted). The word **wordonikewo** has lexical word worg-(seek), –nikewo(as) and there is the unconsidered infix morpheme between lexical word and the suffix.

Postpositions and conjunctions are also attached to the lexical words. So in Koorete, there are no morphological feature forms for prepositions and conjunctions. But there is for Postpositions.

Example: **nuunaana** (with lip) –ana replaces a preposition 'with' and **Abebese** (for Abebe/belongs to Abebe) -se is a preposition which shows belongingness 'to'.

Adey tamaa edho (The father stood in the fire). Adey tama gara edho (The father stood over the fire).

Adey tamaka edho (The father stood in the fire). Adey alepa yodo(The father come from the highland).

Indoy sorona burso (The mother cut with knife). Nayshi Adese kaxa indo (children gave food to father).

Conjunctions attached in different ways to the main word.

Example: Adeyaara, Adeni

In Koorete, singular and plural word formation does not have a simple pattern.

Example: Maydo→Maydiita/Maydaxiita;Zawa→Zawiita; Orge→Orgiita/Orgaxiita.

Knowing Koorete language article type is also a challenging to decide either definite article or indefinite article because of its concatenation to the lexical word. This means there is no article part of speech nomination. For example:
Kana'i yodo (A dog come). Question raised→Ani kana (which dog?)➔indefinite (dog is unknown).
Kanii yodo(The dog come). Question raised →Ani kana(which dog?)➔definite (dog is known).
Ye ade (that father). Question raised →Ani ade(which father?)➔definite (father is known).

Demonstrative pronouns do not show any gender or plurality cases, especially for short and long distance consideration. For example: ha ade, wo ade, se ade, yede ade, ye ade.
Pronouns that does not stand themselves might also attached as prefixes to the lexeme such as (ta, e, hi, u, ne, nu,i) hando. Adverbs are limited in number. Those adverbs could be such as iitana, osaxe, orijhena, fetona, modhena, odhena.

## 2.5. Previous Studies on Koorete Language

The literatures reviewed on the Koorete language are not on part of speech tagging. Because there is no POS tagging conducted. The following, unrelated works literature, reviews are done for the purpose of introducing the Koorete language background, other NLP tasks on Koorete language by other researchers before.

Rolf Theil [17] presents the analysis of Koorete Segmental Phonology on vowels, consonants and consonant clusters, wherein segmental, the individual sounds make up a speech. When studying on Koorete, diphthongs sound also establishes a smart share on Koorete phonology, and so Koorete has no distinctive opposition between affricatives and fricatives. But it has distinctive oppositions between single and geminated sibilants in consonant clusters. The Koorete has two orthographies in which the system of spelling in a language being represented. The first one is based on the Latin alphabet and the other is based on the Ethiopian alphabet. The Latin-based orthography is clearly based upon the phonological analysis, in which a traditional, structuralism definition of the phoneme is immersed. There are different significant terms used in this phonological analysis such as *phonemes* – an indivisible unit of sound or a bundle of distinctive features; *allophones* - any of two or more alternative pronunciations for a phoneme; *prosody* – the pattern of sounds in relation with stress and intonation; *tones* – quality of somebodies voice pitch to the lexical realization; *syllabification* – the division of words into syllables; *accents* – way of pronouncing the words that shows which country, area a person comes from; *fricatives* – sound production (like sibilant, hissing, buzzing quality) by air flowing through oral cavity; *affricatives* – a sound produced using a combination of plosive and a fricative. By nature Koorete language is a tone language and accentual language. Accentual language is a lexical contrast between tone and no tone allowing only one tone specified to a

surface. Generally speaking, words are proved to be stored in the mental lexicon with wide phonetic detail and its similarity relations among parts of phonetic strings.

Beletu Redda [28] presents Morphology of Koorete to include a structural descriptive account of the inflection and derivation of nouns, pronouns, verbs, adjectives, adverbs, postpositions, numerals. In this study, it is shown that Koorete nouns are inflected for number, gender, definiteness and case.

Researcher identifies that geminating consonant and vowel length are phonemic in the language. Concerning the verbs inflection, this study classifies and describes the verbs into three groups on the basis of their forms, such as the infinitive, the perfective and the imperfective forms. The Koorete language consists of 29 consonant and 5 vowel phonemes as (Hayward, 1982) presented on the title “Notes on the Koyra Language”, and also 30 consonant and 5 vowel phonemes as (C. Ford, 1990) presented on the title “Notes on Koorete Phonology”. For phonemic inventory, this study used consonant phonemes, vowel phonemes, phonotactics, consonant clusters, vowel length and consonant germination, morphophonemic processes (such as vowel insertion, vowel deletion, assimilation, and epenthesis).

# Chapter 3. Approaches of Part of Speech Tagging

## 3.1. Types of POS Tagging Approaches

As the existence of different natural languages, there is different sentence structure and one's language morphological features are relatively different from another. Natural language is not easy to process as it is ambiguous, subjective and complex [20]. Because of these, many researchers has used different learning approaches to automate such as rule-based, probabilistic corpora-based, neural network and deep learning approaches.

### *3.1.1. Rule-based POS Tagging Approach*

This approach uses dictionaries, lexicons and hand-crafted rules. It assigns the more probable tag to a word from the dictionary but it depends on the collection of rules it has. Then after write rules by hand to selectively remove tag that is irrelevant, but leaving the correct tag from each word. But it is rigid because of the collection of rules it depends on. This means, it is not flexible to determine the word category when new featured word appears [8].

**Transformer based learning(TBL) approach** is an algorithm that categorize the word class tagsets based on rules it learns directly from original training corpus without expert knowledge[22]. It needs human only to correctly prepare annotated training corpus. Then after, the system can derive lexical and contextual information from the given corpus so as to facilitate performing the task of POS tagging to words theirs respective tags. Such as NLTK Regular expression tagger, Brill tagger, etc. does this type of tasks.

**NLTK Regexp Tagger:** Regular expression is a formal language for specifying text strings word class. It allows defining one's own word patterns for determining the POS tag.

Regexp tagger is used together with N-gram tagger using '**backoff'** keyword for handling unknown words as it gives better score than the default tagger [2]. Backoff is used to refer to the lower tagger when the higher tagger fails to determine a POS tags. Most of the time, regular expression patterns are tested at the middle, at the end and at the beginning of the POS tagger chain. Example regular expression in Koorete language:

```
KooreetePattern_1 = nltk.RegexpTagger ([
        (r'.*sso$','VBP'),                    ## Verb perfect
        (r'.*oko$','ADV'),                    ## Adverb
        (r'.*ita$','NPL'),                    ## Noun plural
        (r'.*nxo$','NUMO'),                   ## Ordinal number
        (r'.*atse$','ADJ'),                   ## Adjective
        (r'.*iyo$','DFN'),                    ## Definite noun
        (r'.*ka$','LCN'),                     ## Locative noun
        (r'.*ra$','CCN'),                     ## Commutative noun
        (r'^-?[0-9]+(.[0-9]+)?$','CD')  ],  ## Cardinal number
                backoff = dtTagger)
```

**Brill Tagger:** it does a quick job first using frequency from temporary corpus, and then revises using contextual rules or transformation rules. This starts with simple and less accurate rules to train better ones first from tagged corpus. During the training it does the following activities repeatedly in stepwise.

1) Tag each word initially with most likely POS
2) Survey set of transformations to see which improves tagging decisions compared to temporary tagged corpus
3) Re-tag corpus using best transformation
4) Repeat until the performance doesn't improve
5) Obtain the result: tagging procedure with ordered list of transformations can be applied to new untagged text

Example of transformation rules for English language:

```
NN -> NNP if the tag of words i+1...i+2 is 'NNP'
NN -> VB if the tag of the preceding word is 'TO'
NN -> VBD if the tag of the following word is 'DT'
NN -> VBD if the tag of the preceding word is 'NNS'
```

### *3.1.2. Stochastic or Corpora-based POS Tagging Approach*

This approach is also called probabilistic tagging approach because it bases on the probability of certain tags occurring given various possibilities, and there is no probability for words not in corpus. Stochastic approach chooses most frequent tag in training text for each word (frequency based probability of a word co-occurrence with respect to its neighboring words i.e., it chooses the most suitable tag to a word [22][8]). It is somewhat moderately accurate by fine-tuning the context based on the features from training corpus. Its approaches easily classified with the help of many different stochastic NLP modeling algorithms such as CRF, HMM, N-Gram, SVM, etc.

**Conditional Random Field** (**CRF**) is rich in feature information when compared with others POS Tagging models. It does not only specifies the probability of possible label sequences given an observation sequence (on current observation) [10], but also on previous and next words. CRF is a powerful and efficient framework for sequence labeling tasks, such as named entity recognition, information retrieval, information extraction and POS tagging as well. It handles linear and non-linear features, parametric and non-parametric features in order to tune behavior of the classifier on over-fitting and under-fitting regulation, and a cut-off threshold for feature frequency [11].

**Hidden Markov Model** (**HMM**) chooses the highest priority tag for a word given a sequence of words with their own potential tag sequences. Words can have one or more tags and a sequence of tags with the highest probability is chosen. In a HMM, the words in a sequence are considered as observations. Each observation relates to a hidden event associated with the word in sequence, and then a matrix A holding transition probabilities of one state moving into another state and emission probabilities B is generated from i state [3].

**N-Gram Method** uses the frequency of parts-of-speech tag, word or both word and corresponding POS tag sequences to produce the probability of a word by considering contextual information within a given sentence [8]. It can calculate the probability of tag sequence given the previous n-1 tags and it can also calculate the probability of the word sequence and the probability of tag and word sequence. An N-gram is a sequence of a specified number (n) of words occurring in a sentence from the given vocabulary. In the usage of n-grams algorithm (with n > 1) the probability of a certain word is computed based on the previous word(s), and then predict the word class of the current word [36]. So the upcoming words are influenced by the previous words context with the words context window bound mobility to the forward and backward direction. As this study deals about next word prediction, N-gram deals with guessing what word comes after based on some current information.

This study does using the N-gram language model up to the Trigram tagger. To estimate each Trigram probability $P(\vec{w})$ from the tagged corpus, this study used the chain rule for N-gram based statistical POS tagging approach. This rule is as follows.

$$P(w_1, \ldots w_n) = \prod_{i=1}^{n} P(w_i|w_1, w_2, \ldots w_{i-1}) \quad \text{.....(1)}$$

$$\approx P(w_1)P(w_2|w_1)\prod_{i=3}^{n} P(w_i|w_{i-2}, w_{w-1}) \quad \text{.....(2)}$$

$$P_{MLE}(w_i|w_{i-2}, w_{i-1}) = \frac{C(w_{i-2}, w_{i-1}, w_i)}{C(w_{i-2}, w_{i-1})} \quad \text{.....(3)}$$

The chain rule conditional probabilities for each individual tagger up to the Trigram tagger are

**trigram model**: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1})$
**bigram model**: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i|w_{i-1})$
**unigram model**: $P(w_i|w_1, w_2, \ldots w_{i-1}) \approx P(w_i)$

For example the probability of sentence “Zeere denxoy accaaka axe miidheko” using a Trigram tagger would be:

P(Zeere denxoy accaaka axe miidheko) = P(miidheko|Zeere denxoy accaaka axe)
= P(miidheko|accaaka axe)

**Combining Taggers using Backoff**

Backoff is used to refer to the lower tagger when the higher tagger fails to compute and determine POS tags [2]. It is used to address the increase in accuracy, and its coverage is to the more accurate algorithms calling back to the lower tagger. For this we combine the results of a trigram tagger, a bigram tagger, a unigram tagger, a regexp tagger, and a default tagger using backoff keyword, as follows:

1. Try tagging the token with the Trigram tagger.
2. If the Trigram tagger is unable to find a tag for the token, try the Bigram tagger.
3. If the Bigram tagger is unable to find a tag for the token, try the Unigram tagger.
4. If the Unigram tagger is unable to find a tag for the token, try the Regexp tagger.
5. If the Regexp tagger is also unable to find a tag, use a Default tagger.

Example of code segment: the code used '**train'** as the training sentences; '**tokens'** as the validation checkup of new corpus; '**tag_fd**' parameter in default tagger to refer to the frequency distribution of tags in the training data.

```
dtTagger = nltk.DefaultTagger(tag_fd.most_common()[0][0])
dttagged = dtTagger.tag (tokens)
KooreetePattern_1 = nltk.RegexpTagger ([
        (r'.*yaaka$','VBG'),    (r'.*o$','VB'),   (r'.*sso$','VBD'),
        (r'.*ese$','VBF'), (r'.*oko$','ADV'),    (r'.*ita$','NPL'),
        (r'.*nxo$','NUMO'),    (r'.*atse$','ADJ'),        (r'.*yeca$','VBPC'),
        (r'.*wayte$','VBIP'),    (r'.*iyo$','DFN'),        (r'.*ko$','GCN'),
        (r'.*ka$','LCN'),         (r'.*ra$','CCN'),         (r'.*fa$','ACN'),
        (r'^-?[0-9]+(.[0-9]+)?$','CD')       ], backoff = dtTagger)
rtagged = KooreetePattern_1.tag(tokens)
uniTagger = nltk.UnigramTagger(train, backoff = KooreetePattern_1)
unitagged = uniTagger.tag (tokens)
biTagger = nltk.BigramTagger(train, backoff = uniTagger)
bitagged = biTagger.tag (tokens)
triTagger = nltk.TrigramTagger(train, backoff = biTagger)
tritagged = triTagger.tag(tokens)
```

The discussions for tagsets such as VB, VBP, VBG, VBF, ACN, CCN, LCN, CD, etc. are presented on chapter 4 under KPT corpus tagsets.
The default tag used in this study is noun NN, which is obtained by counting the most frequent tags distribution using the default tagger from the KPT corpus. Besides, the regular expression patterns are shown in this section with the hand-coded rule using Regexp tagger.

### *3.1.3. Neural Network Learning & Deep Neural Network POS Tagging Approach*

#### *3.1.3.1. Neural Word Embedding*

Most work uses words as the *smallest units* in the compositional architecture, often using pre-trained word embedding or learning them specifically for the task of interest [3, pp.1]. Word embedding is one of the most popular representations of document vocabulary |V| to word vectors (Word2Vec) as inputs. It is the collective name for a set of language modeling and feature learning techniques in NLP, where words or phrases from the vocabulary are mapped to vectors of real numbers. It transforms human language meaningfully into a numerical form keeping the syntactic and semantic context, not the actual meaning. This means word vectors are simply arrays of numbers that represent the contextual vector space meaning of a word with respect to its index position given in the vocabulary.

Neural word embedding is simply a process of word embedding using neural network for a current word given surrounding words. It is a distributed representation of a text which allows deep learning methods to perform well on the challenging NLP problems. This embedding enables words having similar meanings (similar context) would be represented similarly (put in same vector space). This is one of the basic advantages in the reduction of out-of-vocabulary (OOV) impact [8, pp.15]. This is possible because words will not be completely unknown as far as they have vector features even if they may not be seen in the training dataset. Because of the use of neural networks, this embedding method generates the Word2Vec mappings for raw word since a word is the underlying input representation. The word embedding practice can be obtained by training a neural network model designed for a specific task [2]. Words are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space. This brings to the idea of generating distributed representations.

When we represent a word as per its index position in the vocabulary, it will be given *one vector*, and all others index position *zero*. The main idea here is that every word can be converted to a set of real numbers N-dimensional vector |V| for the sake of good representation.

This word vector representation has a capability of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc. The basic idea behind word embedding is words in similar context have similar meaning by occupying close spatial positions to each other [12].

Example 1: using *one-hot* vector for word's index position in the vocabulary to represent sentences

**"**Have a good day" and **"**Have a great day**"**

having different meaning, construct a vocabulary V = {Have, a, good, great, day}.

Length of one-hot encoded vector equal to the size of V = 5. Then we get a vector of **zero**s except for the element at the index would be **one**, which represent the corresponding word in the vocabulary.

Have = [1,0,0,0,0]`; a = [0,1,0,0,0]` ; good = [0,0,1,0,0]` ; great = [0,0,0,1,0]` ; day = [0,0,0,0,1]` (symbol "`" represents transpose)

Therefore, we get a 5-dimensional space, where each word occupies ***one*** of the dimensions and has nothing to do with the rest. This means 'good' and 'great' are as different as 'day' and 'have', which is not true.

Example 2:

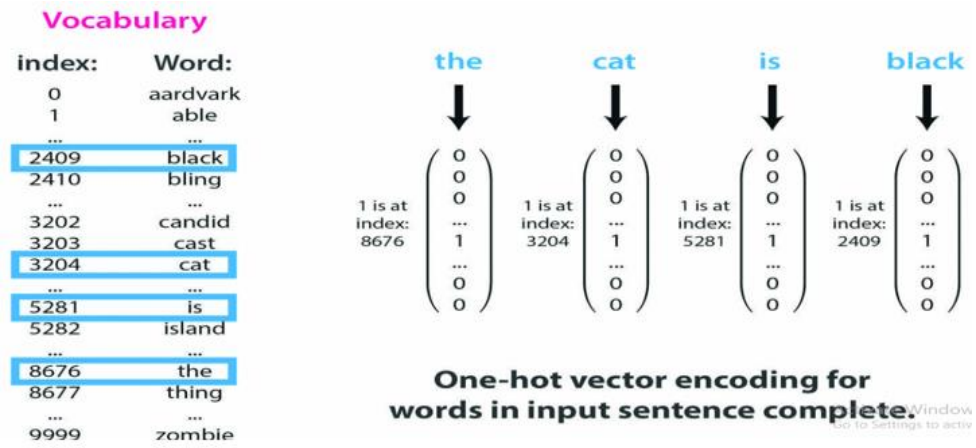



Figure 3.1: word embedding in a one-hot encoding

This figure shows the vector representation for first vocabulary word "aardvark" will be [1, 0, 0, 0, … , 0], which is a "1" in the first position followed by 9,999 zeroes. Such like representation process is called one-hot vector encoding. It refers to a representation where only one bit in a vector space is 1, and all others (a lot of other bits) are zeroes.

Words can be encoded into limited vector spaces or neural word embedding in one of the two methods, the continuous bag of words (CBOW) method and the skip-gram method [15].

CBOW takes the average of the possible contexts of a word in representing a word in a limited dimensional vector space. CBOW model predicts the current word given context words within specific window. The input layer contains the context words and the output layer contains the current word. The hidden layer contains the number of dimensions in which we want to represent current word present at the output layer.

For example, '*harre'* can refer to either the noun form of the donkey or the verb form of frightening person in slang speech. Continuous bag of words places the vector of the word 'harre' to a medium position of the two contexts.

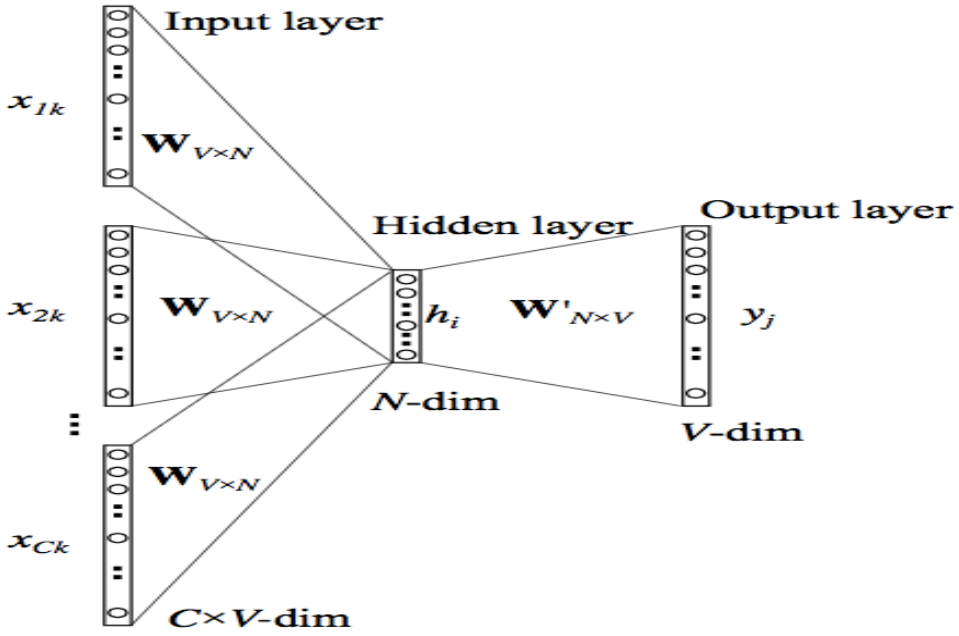



Figure 3.2: Continuous bag of words (CBOW) model (Picture credit: [8])

Notations: x is the one-hot encoded vector of a given word, W is the weight matrix between layers of given dimensions, h is the hidden layer vector and y is the output vector. When $W_{VN}$ is used to calculate hidden layer inputs, we take an average over all these C context word inputs.

Skip-gram model is an efficient method for learning high quality vector representations of words from large amounts of unstructured text data [8, pp.15]. The Skip-gram method learn word embedding through pre-trained word vectors on the unlabeled data using shallow neural network [9, pp.1]. Skip-gram model predicts the surrounding context words within specific window given current word. The input layer contains the current word and the output layer contains the context words. The hidden layer contains the number of dimensions in which we want to represent current word present at the input layer. For Word2Vec is usually dense vector representation, and holds enormous information compared to their size, it uses skip-gram method. Because as the authors of Skip-gram model stated it is efficient and more accurate than CBOW in that it does not involve dense matrix multiplications. This method, on the other hand, can assign two vector values for the above two contexts of 'harre' like that of 'great and good'. Based on the above preferences, skip-gram model is used in this study.

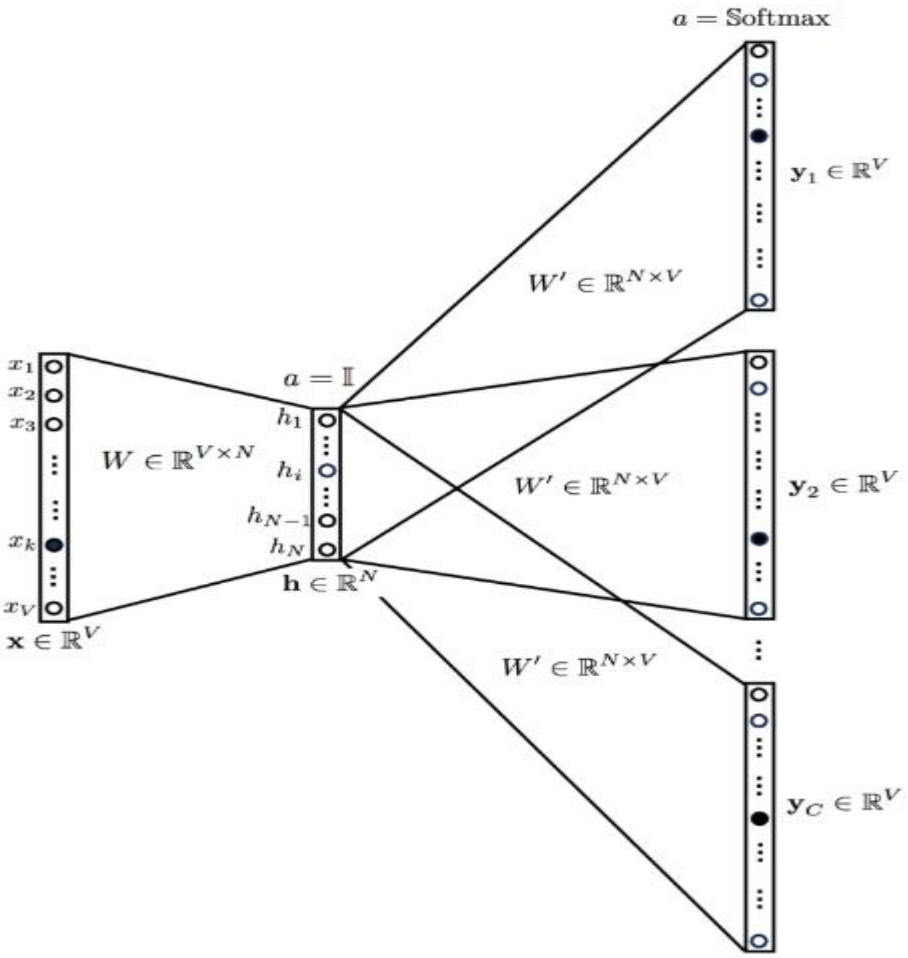



Figure 3.3: Skip-Gram Model (Picture credit: [8])

We input the target word K into the network, and the model outputs C probability distributions. For each context position, we get C probability distributions of V probabilities, one for each word.

This study approach uses NNs (such as ANN, RNN) to simulate human like decision making. It does not require feature engineering i.e., this approach learns directly from original data. So, it is open for new words because it uses neural networks to predict the words with similar context like neurons of human brain inspired.

**Artificial Neural Network** (**ANN**) is a powerful program which is determined by its component processing elements, activation signals, neurons, which have weighted inputs, transfer function, and an output [8]. The activation signals of the neurons in the network are the weighted sum of inputs passed through the transfer function to produce an output for each neuron. Besides, the behavior of a neural network is determined by the transfer functions of its component neurons, the learning rule and the architecture of the network itself. ANNs learn patterns and relationships from experience obtained in the input data and not from programming.

**Recurrent Neural Network (RNN) and Long Short Term Memory (LSTM)**

**RNN** is a powerful and robust type of neural network which computes on sequential data. Sequential data is basically an ordered data in which related things follow each other, and also it is a time series data where a series of data-points are listed in time order. RNN is the first state-of-the-art algorithm [2] that remembers its immediate input, due to an internal memory, which makes it perfectly suited for machine learning problems that involve sequential data. It remembers the input it received, which allows it to be very precise and preferred algorithm in predicting what is coming next for sequential data like time series, speech, text, audio, video, weather, and much more. Hence, it can form a much deeper understanding of a sequence labeling and its context compared to other algorithms. RNN is preferred for sequence labeling representation because of its ability to connect the previous information with the current sequence of tasks [8]. For RNN is repeated copies of the same network putting in sequence, each network passes a message to its successor as shown in figure below.

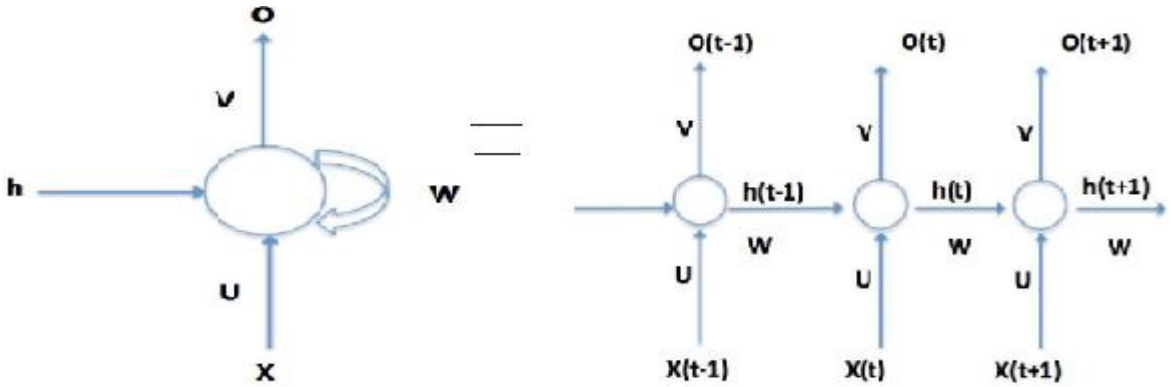



Figure 3.4: Typical and Unrolled representation of RNN. (Picture credit: [8])

The notations U, V and W represent the weight matrix for each time step; whereas t is the time and X, h and O, on the other hand, represent the input layer, hidden layer and output layer vectors.

This study uses RNN and LSTM sequential models to perform an algorithm for sequential data. RNN and LSTM take a sequence of input, give back a sequence of output, and remember their previous input due to an internal memory [2]. These models just address a sequence-to-sequence problem setting in which sequential data needs to be computerized for good real-time representation. RNN remembers and stores a previous recent sequence record; whereas LSTM enables better preservation of "long-range dependencies" over a long period of time. This means LSTM remembers the long previous sequence of data depending on the time series.Besides, LSTM allows for a better control over the exploding and vanishing gradient flow due to the input and forget gates as sketched here below.

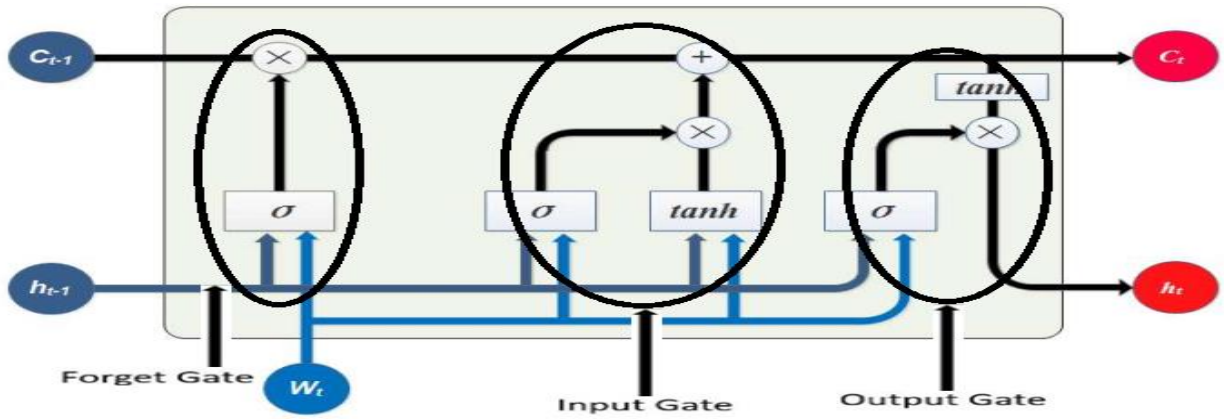



Figure 3.5: The repeating module in LSTM. (Picture credit: [8])

As the figure 3.5 above representing, the top horizontal line in the repeating module of LSTM is called a cell state. Information passes or denied through the cell state of LSTMs by using gates made from a sigmoid network layer and a point-wise multiplication operation. *Forget gate* determines what details to be discarded from the memory block, and discarding is decided by the sigmoid function. This gate looks at the previous state ($\mathbf{h_{t-1}}$), the content of input ($\mathbf{X_t}$), and outputs a number between **0(***omit this*) and **1(***keep this***)** for each number in the cell state $\mathbf{C_{t-1}}$. *Input gate* takes the input value from memory and modify the current active memory. Sigmoid function decides which values to let through 0, 1, and hyperbolic (tanh) function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1. *Output gate* used to decide the output based on the data from the input and memory block. Sigmoid function decides which values to let through 0, 1. Besides, tanh function gives weightage to the values, which are passed deciding their level of importance ranging from-**1** to **1** and multiplied with output of Sigmoid**.**

**Bi-directional Long Short-Term Memory** (**Bi-LSTM RNN**)

**Bi-LSTM RNN** are designed to handle remembering long context dependencies [1][6], and to handle both the previous and the future information required for the current task, including the hidden layer to get the whole information about the context of words. For the long remembrance, it has the forget gate which keeps all the states in it.

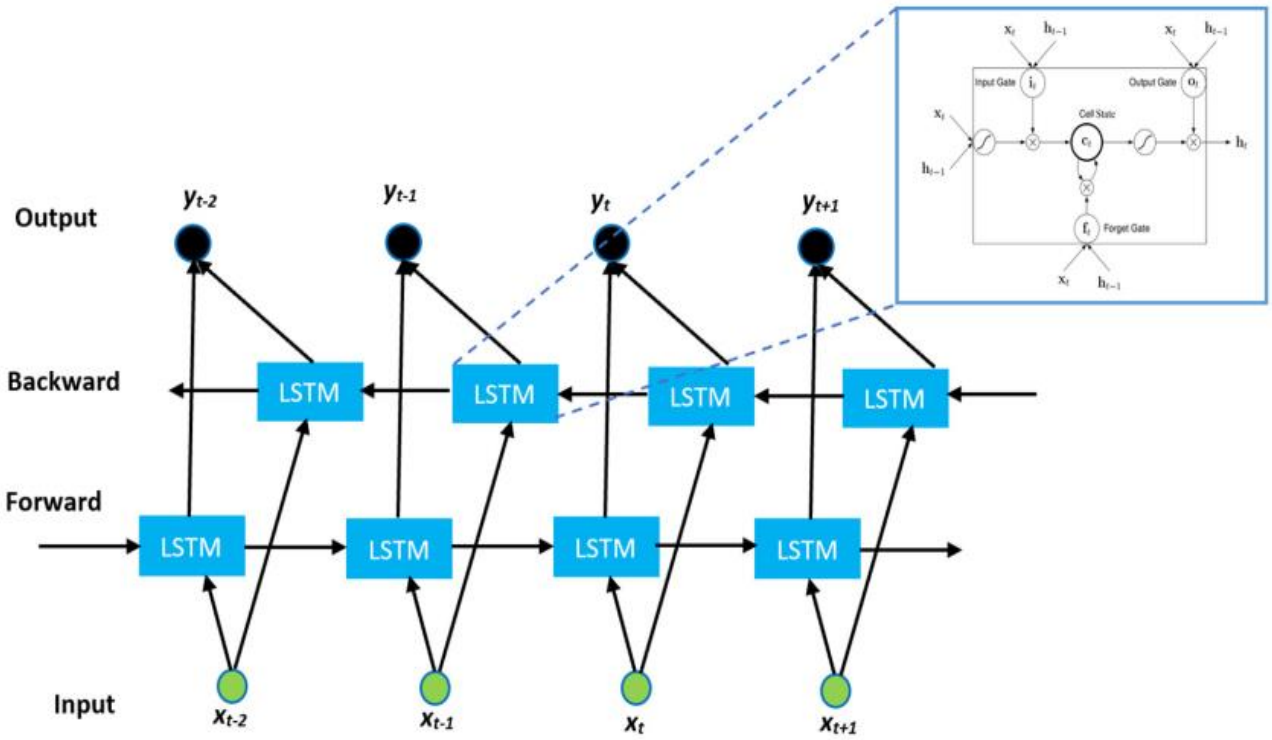



Figure 3.6: Overview of the bidirectional LSTM model (picture credit: [35])

The above figure 3.6 works the same as the work of LSTM. In addition to the LSTM works, it doubles the job of sequence labeling prediction to the reverse direction-, which is Backward LSTM and Forward LSTM; - around the current word using the context window size for a long range of times order. Time t is the index of the word in the sequence S for handling the two time directions, input data from the past and future of the current time framed; whereas standard RNN requires the delays for including future data.

## 3.2. Literature Review

The following papers are reviewed for the scientific study testimonial before, and for the sake of enough evidence gathering from published, and deliberation of new study undergoing.

### *3.2.1. Rule-based POS Tagging Approach*

Birhanesh Fikre[22] presents Wolaita language POS tagging using transformation based learning (TBL) model to enrich the language in resource for high-level NLP tasks such as syntactic parser, NER, machine translation, information retrieval, spelling and grammar checking, natural language dialogue interfaces to databases, stemming, …, etc. This paper used TBL algorithm rule to train up on the Wolaita language corpus (26-POS tag sets, 1134 sentences, 14358 words) by deriving lexical and contextual information from correctly

annotated corpus. TBL rule has initial state tagger phase and learning phase. Actually, TBL take untagged corpus as input, and initial tagger likely tags for the tokens in the untagged corpus, and then yields temporary corpus as output. Next, the learning phase fed with temporary corpus and reference (goal) corpus for rule derivation comparison. The temporary corpus passes through the learning phase iteratively to derive rule transformation. The learning phase trains continually until no change of rule occurrence, and then produce ordered list of rules to apply on untagged corpus. It used 90% for training and the remaining 10% for testing. Finally, it performed accuracy of about 92.96% on Trigram tagger.

### *3.2.2. Stochastic POS Tagging Approach*

N.X.Bach et al. [24] presents an empirical study on POS tagging for Vietnamese social media text in order to specify the word class tag given a social media text, and making the noisy data a general, conventional text using conditional random field (CRF). Because social media text does not always conform to formal grammars and correct spelling, it needs some aspects of convention to accept even its informal as a formally written form. There is also the usage of abbreviations, foreign words, and emoticons frequently in social media. Developing POS tagger for conventional text would perform poorly on such noisy data. CRF fixes this problem as it has various kinds of features such as parametric or non-parametric, linear or non-linear features. This paper investigated the effect of features extracted from word clusters under the Brown and canonical correlation analysis (CCA) based clustering in semi-super-vised settings such like on Facebook media. It performed a series of experiments to evaluate the model. So then the experiment achieved 88.26% and 88.92% tagging accuracy in supervised, and semi-supervised scenarios, respectively, which are nearly 12% improvement over vnTagger, a state-of-the-art and most widely used Vietnamese POS tagger developed for general, conventional text.

Duressa Tamirat [32] discussed on N-gram word prediction by preparing corpus-based Afaan Oromo words at word level and then demonstrated using the Java NetBeans IDE 8.0.2. N-grams do the probability of token sequences of length *N* and a given knowledge of counts of N-grams such as unigram, bigram, etc.; and then it can guess likely next words in a sequence.
This article paper had modeled the word prediction task as the ability to assess the conditional probability of a word given the previous words in the sequence P ($w_n|w_1,w_2 \ldots w_{n-1}$).

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)\ldots P(w_n|w_1^{n-1}) = \prod_{k=1}^{n} P(w_k|w_1^{k-1})$$

For evaluation this used precision and recall metrics. The corpus was composed of a set of 2,242 sentences with 37,272 token words. In this article, words occurring zero times in the corpus were considered as irrelevant words and the corpus had not have standards for Afaan Oromo languages.

This study approach worked at one, two, three or four letters even it was worked at sequence of words to make a sentence. But the accurate performance of the system was not much surprising as the experiment argued that it gave 83.84%, 61.4%, 54.8% of unigram, bigram and trigram respectively. This evaluation analysis shown us that unigram model performed modestly, but bigram and trigram models obviously performed poor.

Grigori Sidorov et al. [37] presented Syntactic N-grams as Machine Learning Features for Natural Language Processing. This paper is saying that N-gram based techniques are predominant in modern NLP and its applications, where 'n' corresponds to the number of elements in a sequence in which one sequence encounter after another in texts. The syntactic n-grams (sn-grams) are n-grams that are constructed using paths in syntactic trees structure. There are various types of sn-grams elements.

- Word sn-grams: the elements of sn-gram are words(or stems or lemmas – alexical unit),
- POS sn-grams: the elements of sn-gram are POS tags,
- SR (syntactic relations) tags: Sn-grams of tags, the elements of sn-gram are names of syntactic relations, character sn-grams
- Mixed sn-grams: sn-grams are composed by mixed elements like words (lexical units), POS tags and or SR tags, which are composed of a syntactic link.

Finally, this paper used SVM, NB, and J48 to construct experiment in comparison with sn-grams syntactic trees for several profile sizes.

J.H. Yousif [25] presents HMM Tagger for Applications-Based Arabic Text as a review for efficient information transfer on the Internet. Because of there was a complexity for efficient POS tagger on the Arabic language itself. Besides, there was a challenge of tagging disambiguation and unknown words. This paper explored and examined a group of research papers that applied the POS tagging to the Arabic language using the HMM model. HMM is a finite set of states machine classifier, which involves a set of events with specific input values. The probability of current word N will be computed based on the probability of the previous two words N-1and word N-2, and then determine probability of the unidentified word's P(W|T). The experiments used 40 sentences with 485 words. From a higher accuracy achieved number of researchers two are scored rate of 97.6% and 97.4%.

T.B.Shahi et al. [26] presents SVM based POS Tagging for Nepali Text, in which getting accurate POS tagger is the challenging task because of the morphologically richness nature of the Nepali language, tag set it used and size of the corpus. For the sake of attaining good accuracy, analytical SVM based POS tagger is implemented and tested for various instances of input (linguistic units' word, phrase, & sentence) to verify the accuracy level. SVM is a supervised machine learning algorithm that handles binary classification problems. So that the SVM tagger operated the feature vectors for each word as input, and classify the word into one of two classes (1 or 0). SVM can handle many large features and resist of over-fitting that rule-based and HMM can't accommodate. So SVM performs good due to it is efficient, portable, scalable and trainable behavior. The test data contains total of 10775 randomly selected tokens

out of which 82% are unambiguous, 13% are unknown tokens and 5% are of ambiguous. Finally, SVM achieved accuracy of overall for known words and unknown words about 93.27%.

### *3.2.3. Deep Neural Network POS Tagging Approach*

Anastasyev et al. [9] presents the performance improvement of POS tagging using Word2vec framework on unlabeled data over pre-trained word vectors on each row, and using CharFF model to increase the slower speed of Bi-LSTM variant of character embedding. The paper used Bi-LSTM for sequence of labeling from forward and backward passes of LSTMs with the hidden states to receive the whole information about both contexts in predicting POS tags. Bi-LSTM tagger is state-of-the-art tagging accuracy based on RNN for its ability to handle long context dependencies on whole sentence in contrast to the classic model prediction conditioned on narrow context window. It demonstrated on three datasets: PTB (45 tags), Gikrya (304 values) and Syntarus (908 values), and when experimented on the three datasets yielded accuracy in between 96% and 98.5%.

PeiluWang et al. [2] discusses word embedding as a low-dimensional real-valued vector to represent word. The word embedding contains part of syntactic and semantic information, and so it represent word as a one-hot vector by training a neural network design for a specific task. Word embedding helps for effective tagging of sequential data as the most work expressed 'word is the smallest data' for out-of-vocabulary word. This paper used Bi-LSTM RNN in sequence labelling to predict a state-of-the-art tagging probability distribution of each word. This model achieves superior performance in language modeling, language understanding and machine translation without using the morphological features. The experiment exerted using word embeddings trained on the first 10 million words (WE(10m)), first 100 million words (WE(100m)) and all 530 million words (WE(all)) of North American news corpus, and the finally the last trial on BLSTM-RNN+WE(all)+suffix2 gave the maximum accuracy of 97.40%.

J. Wieting et al. [3] presents embedding words and sentences via a character N-gram count vector as evaluating word sequences through word similarity, sentence similarity, and POS tagging. Charagram represent a character sequence by a vector containing counts of character n-grams. This vector is embedded into a low dimensional space using a single non-linear transformation to produce effective sequence embedding when a summation performed over the character n-grams in the sequence. As multiple word similarity and sentence-level semantic textual similarity CHARAGRAM outperforms RNNs and CNN architectures, POS tagging use Bi-LSTM tagger, achieving state-of-the-art performance. It used Bi-LSTM RNN on characters to embed arbitrary word types, showing strong performance for language modeling and POS tagging. For word embedding training and tuning, it used a lexical section of PPDB XXL consisting of 770,007 word pairs. It used datasets either WS353 or SL999 for model selection with best result containing 173,881 n-gram embeddings, and using WS353 for model selection and training for 25 epochs, achieved 70.6 on SL999. For sentence embedding the training

model pass through PPDB XL, which consists of 3,033,753 unique phrase pairs with 68.7% and POS tagging 97.1% .

Erick R et al. [5] evaluates Portuguese POS tagger using word embeddings, and its revised corpus, by representing words in terms of real-valued vectors in a multidimensional space of size V. This is about how to map a given unlabeled corpus to a vector space, where each word has a corresponding real-valued vector, regarding its syntactic and semantic information similarity. The concept of similar words has similar vectors applied according to Euclidian distance or cosine similarity. Words having similar meaning encoded in a same contexts or space and belongs to a same POS category. POS tagger is a preprocessing step in NLP tasks. This article used Mac-Morpho corpus by preprocessing features such as the upper/lower cases, prefixes/suffixes, singular/plural naming tags, tag mistypes, different writing styles, missing words, and repeated sentences to provide a more reliable resource and to reduce sparsely word distribution. Taggers take window of tokens to map tag tokens in a sentence to its vector using neural network. Skip-gram modeling is used for generating vectors.

It has carried out another experiment on POS tagging for Oromo language using Maximum entropy Markov Model and used a manually annotated corpus of 452 sentences (total of 6094 words) with 33 tag-set and achieved accuracy of 93.01% which is evaluated by using tenfold cross validation.

Bin et al. [6] discusses the evaluation of popular word embedding models and good representation properties by demonstrating word similarity, word analogy, concept categorization, outlier detection, and QVEC. It evaluated word embedding methods based on their techniques such as neural network language model, CBOW and skip-gram, Co-occurrence Matrix, FastText, N-gram model, Dictionary model, and Deep contextualized model. Evaluators can be of two types: 1) intrinsic evaluators test the quality of encoding independent of specific NLP tasks, and, 2) extrinsic evaluators use word embeddings as input features to a downstream task and measure changes in performance metrics specific to that task e.g., NER, POS, chunking, sentiment analysis, neural machine translation. Even though there is no unified evaluator that analyzes word embedding model comprehensively, they measure syntactic or semantic relationships among words directly. Good properties of embedding models aim for non-conflation, robustness against lexical ambiguity, demonstrations of multi-facetedness, reliability, and good geometry. Good word embedding evaluators aim for good testing data, comprehensiveness, high correlation, efficiency, statistical significance. The evaluator goal is comparing the characteristics of different word embedding models with a quantitative and representative metric. It presented a stochastic approach to the coding of English language which shows one of the efforts made to provide solution on these word class problems or part of speech tagging. Chirag and Karthik (2008) used Conditional Random Fields (CRF) to develop POS tagger for Gujarati language. They used 600 sentences tagged and 500 sentences untagged for training the POS tagger. They achieved an accuracy of 92% using 10,000 words for training and 5,000 words for testing with 26 tag sets.

Khwlah et al. [20] studies on Arabic POS tagging using neural network and deep learning to automate NLP applications. Natural language is naturally not easy to process as it is ambiguous, subjective and complex. Many researchers has used different learning approaches to automate such as rule-based study, probabilistic corpora-based study, and neural network and deep learning. The rule-based approach is not flexible because it depends on the collection of rules whereas the probabilistic/stochastic approach is somewhat moderately accurate by fine-tuning the context based on the features. This paper designed a model using LSTM RNN to represent Arabic words and morphemes from Quranic Arabic Corpus (QAC). LSTM has the ability to learn long-term dependencies more accurately but does not require feature engineering. LSTM cell has a forget gate that used to remember relevant data for current state, and forgets irrelevant things in the past. LSTM input layer is used to control the flow of new inputs and updates the state. The output of cell state is performed using a sigmoid layer, in which the tanh determines the values between -1 and 1. But according to this study of the word2vec one-hot encoding the cell state uses values between 0(does't represent word) and 1(represent word). LSTM tagger achieved 99.72% accuracy for tagging morphemes and 99.18% for tagging words, while the Word2Vec tagger achieved 99.55% for tagging morphemes and 97.33% for tagging words.

### *3.2.4. Semantic Similarity*

P. SRAVANTHI1 [29] presents the measurement of semantic similarity between sentences which is closely related to semantic similarity between words. This measurement makes a relationship between words to enhance the concepts of semantics over the syntactic measures to categorize the pair of sentences effectively. Semantic similarity relation is calculated at document level, term level and sentence level based on the terms which describe the internal concepts. There are many relation types on which one researcher does a comparison of similarity between words and sentences. These relations are synonymy, autonomy, hyponymy, member, similar, domain and cause, and so on. This paper exploits preprocessing of pair of sentences which identifies the bag of words and then applying the similarity measures like cosine similarity measures. Sentence similarity relation in vector space determines how similar the meaning of two sentences is. Cosine similarity is a kind of relation to measure the given pair of sentences similarity to each other, and specify the score based on the words overlapped in the sentences.

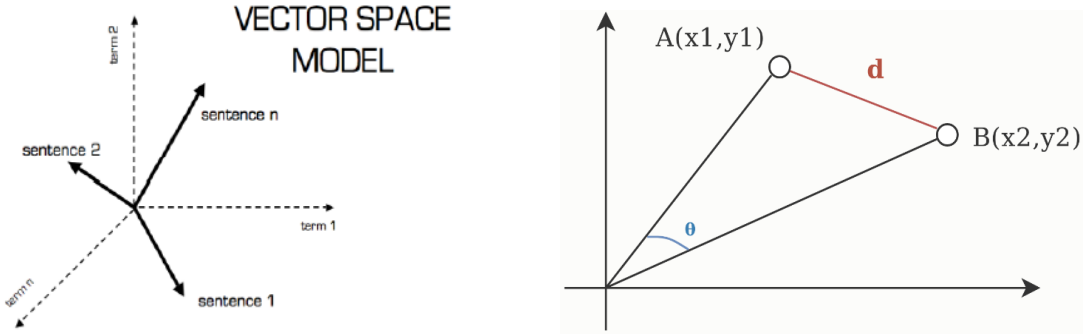



Figure 3.7: a) Vector space model b) Cosine similarity distance d

Similarity = COS (A, B) = COS (Θ) = A.B/||A|| ||B||

Where ||A|| ||B||= sqrt(A).sqrt(B)

Sentence ➔Terms/Words ➔Vectors

Semantic similarity of words or sentences is based on the meanings of the words and the syntax of the sentence; here syntax refers to the different symbolic and structure information that convey the same meaning. Semantic similarity measure can fall into syntactical and lexical categories. The syntactic detects semantic similarity using syntactic dependency relations to construct a picture of the meaning of the compared texts by identifying whether a noun is considered the subject or the object of a verb. This means the syntactical similarity of two words or sentences corresponds to the correlation between the vectors; the cosine of the angle *Θ* between vectors.

The higher the score, the more similar the meaning of the two sentences; in cosine similarity the distance value is bounded in between -1 and 1. The cosine similarity increases when the distance between the two words/sentences decreases; the cosine similarity decreases when the distance between the two words or sentences increases. If the two words of the sentence matches, it returns a score as 1; otherwise 0.

Serkan et al. [21] presents content-based SMS classification application by using word2vec based feature extraction. SMS(short message service) is considered a more reliable privacy-preserving technology for mobile communication while mobile instant messaging applications such as WhatsApp, Messenger, Viber offer benefits to phone users such as price, easy usage, stable, collective and direct communication. SMS directs the institutions to perform the product promotions such as advertising, informing and promotion. Spam filtering is a problem because spam messages sent from unknown sources constitute a serious problem for SMS recipients. Content-based classification model uses machine learning to filter out unwanted messages. Its classification model is created with the help of word2vec word embedding tool so as to calculate the distance of messages to spam and ham words. Spam e-mails are dispatched over the internet while SMS are sent over the mobile phone. The proposed model used word2vec to find the semantic similarity relationships between words in the sentence in the SMS message. Words in the message are transformed into vectors, and the distances between them are calculated and an analogy is established between the words by using word2vec. Based on the analogy similarity integration into the existing SMS apps, the proposed study detect spam (unwanted) SMS from the normal SMS message, and prevent spam messages. The distribution of created features of word2vec spam SMS message looks like the following based on the word versus distance to spam versus distance to ham in the sample message feature extraction.

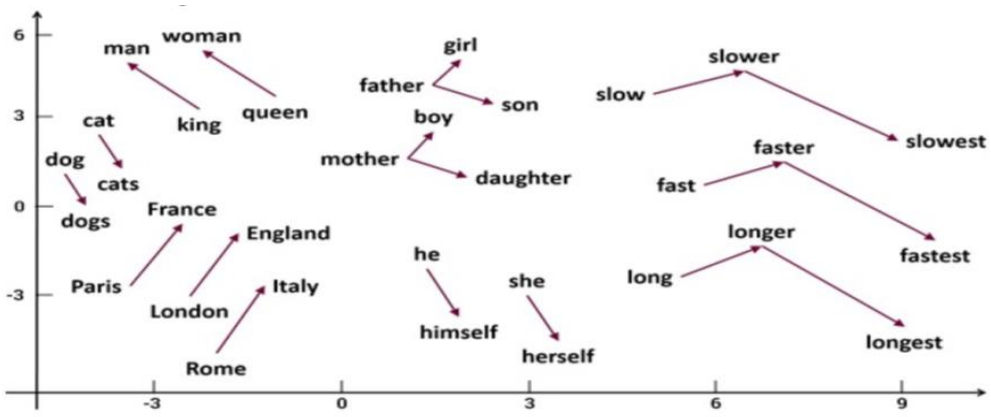



Figure 3.8: Semantic relations of words in vector space

Pinky et al.[23] presents a comparison of the semantic similarities between short texts to maximize human interpretability by applying lexical matching, linguistic analysis or semantic features. It measures semantic similarity to take the degree of semantic equivalence between two linguistic items (concepts, sentences, or documents). The basic idea behind computing text similarities is by determining feature vectors of the documents and then calculating the distance between these features, especially applying cosine similarity using word embedding. It means a small distance between these features give a high degree of similarity, whereas a large distance means a low degree of similarity. It used neural word embeddings (word2vec) by training in neural network over n-dimensional vector of numerical features that represent some object or context. Methods used to calculate text similarity are cosine similarity using tf-idf vectors (76.8%), cosine similarity using word2vec vectors (75.9%), soft cosine similarity using word2vec vectors (76.05). Similarity function or measure is a real-valued function that quantifies the similarity between two objects.

F. Rahutomo [30] presents semantic cosine similarity where a text expressed as a vector of terms, and the similarity between two texts is derived from cosine value between two texts' term vectors. The query-to-document relation using cosine similarity can't handle the semantic meaning of the text perfectly but of same dimensional vectors. It aims to increase the similarity value between two term vectors, which contain semantic relation between their dimensions with different syntax. The higher similarity score between document's term vector and query's term vector means more relevancies between document and query. Cosine similarity of two vectors can be applied for vectors with same dimension only. A new dimension could be built with zero value in a term vector if there is a dimension with no pair in another term vector.

Table 3.1: Literatures Review Summary of related works

| Source and Year | Objective | Methods | Dataset | Performance |
|---|---|---|---|---|
| Anastasyev et al. [9], 2018 | Improve POS tagging performance using Character embedding on English and Russian languages | Bi-LSTM, RNN, CharFF, Word2vec | PTB(45 tags), Gikrya(304 values), Syntarus (908 values) | Accuracy achieved between 96% and 98.5% |
| PeiluWang et al. [2], 2015 | Represent POS tagging using word embedding | Bi-LSTM, RNN | - 10 million words<br>- 100 " "<br>- all 530 " " | Maximum accuracy achieved 97.40% |
| JohnWieting et al. [3], 2016 | Embedding words and sentences on character n-gram | Bi-LSTM, RNN, CNN | -PPDB XXL 770,007 words pair<br>-3,033,753 unique phrase pair | 70.6% on SL999 & POS tagging 97.1% |
| Pinky et al.[7], 2019 | Maximize human interpretability by semantic similarity between texts | Word2vec with Cosine similarity | 2000 news articles of which D1-D5 documents comparison | Accuracy of 76.8%, 75.9%, 76.05% |
| Birhanesh F.[22],2020 | Predict next words word category | TBL | 26 tag sets<br>1134 sentences<br>14358 words | 92.96% accuracy on Trigram tagger |
| N.X. Bacha et al. [24], 2018 | Specify word class tag | CRF | 4000 sentences from Facebook | 88.26% & 88.92% rated |
| J.H. Yousi et al. [25], 2019 | Predict probability of tagging | HMM | 40 sentences<br>458 words | 97.6% and 97.4% |
| T.B. Shahi et al. [26], 2013 | Handle challenging features in binary forms | SVM | 10775 tokens | 93.27% for known and unknown words |
| Grigori Sidorov et al. [37], 2012 | Training machine learning SN-grams as features | Syntactic N-grams, SVM | Profile size of 400 for bigram, and 700 for trigram | SVM and SN-gram achieved 100% both with bigram and trigram |
| Erick R. et al. [5], 2015 | How map unlabeled data to vector space | MEMM | Mac-Morpho corpus<br>452 sentences | 93.1% accuracy |
| Bin et al. [6], 2019 | Evaluate word embedding | DNN, N-gram, Dictionary | 26 tag sets<br>1100 sentences | 92% accuracy |
| Serkan et al.[21], | SMS classification | Word2vec | 5574 lines of short | Accuracy rate |

| | | | | |
|---|---|---|---|---|
| 2018 | into spam and ham | with semantic relations | messages 4827 ham and 747 spam | of 99.64% |
| Khwlah et al.[20], 2019 | Automate NLP applications | LSTM RNN and word2vec | 77,915 words | 99.55% for tagging morphemes and 97.33% for tagging words |
| Duressa Tamirat [32], 2016 | Corpus-based word prediction | N-gram tagger | set of 2,242 sentences with 37,272 token words | 83.84%, 61.4%, 54.8% of unigram, bigram and trigram respectively |

**SUMMARY**

Based on the aforementioned related literatures and its approaches used for POS tagging performance simplicity, and NLP applications used to represent ones language on the computer for the sake of computerized resource building, this study also has cascaded appropriate approaches. So this research studies on the empirical evaluation of neural word embedding (Bi-LSTM RNN) and N-gram based statistical approach on KPT corpus of 33,220 words. It means, these approaches assign POS tags to words based on context similarity and then this study distribute words in the vector space.

The reason why this study used Bi-LSTM RNN is, this approach gives the state-of-the-art performance results about 97% and above tagging accuracy in the new corpus [**2**, **3**, **5**, **6**]. These literatures are saying Bi-LSTM RNN is a 'state-of-the-art' algorithm for better accuracy. This means, it is the current and recent trend that most researchers are using this approach for POS tagging on several languages without the consideration of morphological features. Besides, it is deep learning algorithm. This approach operates well on the dataset of minimum size **10K**.

The reason why this study used N-gram statistical based learning approach is, most of the Ethiopian languages researchers are using statistical based learning approaches. Because the Ethiopian languages need new corpus preparation for statistical based learning patterns; and, most of the time, they do not need rule-based approaches. Because the Ethiopian languages do not have computerized pre-prepared rules, lexicons, dictionaries, and phonemes. For the NLP applications, especially during POS tagging, mostly Ethiopian researchers use HMM and N-gram to predict words class. Besides, according to Grigori Sidorov et al. [**37**] syntactic N-gram based techniques are predominant in the modern NLP and its applications.

The language Koorete is not well studied in the linguistics computation of computer-aided NLP applications. This paragraph is the most reason why this study is choosing the mostly used POS tagging approaches (N-gram statistical and neural word embedding) by many recent researchers in contributing theirs shares.

### 3.3. Evaluation Metrics for Testing

Depending on frequency of each tags instances, this study tested the performances of tagger on set of test data, and evaluated using confusion matrix for each tagsets. Confusion matrix is table form of matrix that contains information about test data and desired tags. To perform the analysis, it requires two parameters namely; reference tags are actual tags of each word supposed to be tagged manually, and predicted tags are basically what the tagger generates using input of test data.

So assessing the evaluation of word class prediction with respect to ground truth from KPT corpus requires labeled data. Accordingly to the confusion metrics parameters(TP, TN, FP, FN) below, this study performs its evaluation task using accuracy, precision, recall, and F1-score measurements to address how much test data are correctly or wrongly tagged.

Table 3.2: Confusion matrix for model evaluation (Table picture credit: [8])

| | | Actual: Positives (1) | Actual: Negatives (0) |
|---|---|---|---|
| Predicted | Positives (1) | TP | FP |
| | Negatives (0) | FN | TN |

True Positive (TP) represents the number of instances categorized correctly to their corresponding class. True Negative (TN) represents the number of instances classified as they are *not* part of a category where they actually are not members. False Positive (FP) represents the number of instances classified into other categories in which they are not actually members. Finally, False Negative (FN) represents the number of instances that are not classified as member of their corresponding category where they actually are members. The metrics evaluation formulae and sample are performed under chapter 5 on point (5.7) subtitled with model evaluating.

# Chapter 4. Research Design

Even though Koorete language is extremely poor resourced in computer-aided NLP applications, it has specific resources in social science researches and linguistics department. The language is researched for Koorete morphology, Koorete segmental phonology, the structure of noun phrase in Koorete, notes on the Koyra language, and the Koore people history. All these researches are not computerized applications since researchers studied only for the sake of quality fulfillment in their corresponding departments under the social science. But only the application of 'Koorete Agric A-Z' dictionary, and New Testament Bible in Koorete have been computerized yet.

This study has chosen N-gram statistical based POS tagging approach and neural word embedding POS tagging approach depending on some reason. These two are chosen because firstly, most Ethiopian languages POS tagging are practiced using N-gram tagger, and secondly, neural word embedding (Bi-LSTM RNN) achieves new state-of-the-art performance on several similarity tasks as cited on [2, 3, 6]. According to Grigori Sidorov et al. [37] paper, syntactic N-gram based statistical techniques are predominant in the modern NLP and its applications.

The procedural steps used in this study are as follows:

- Find a well-documented sources of the language
- Adapt tag sets from other researched literature[17, 22, 28]
- Develop Tagged Corpus - construct the word, and word category
- Preprocessing tokens, types, vocabulary, context window, word vector generating, subsampling, and noise removal to reduce sparsely data distribution
- Build neural word embedding (Bi-LSTM RNN) and N-gram based POS models
- Train the Bi-LSTM RNN and N-gram models by classifying the corpus to training data and testing data.
- Evaluate the models performance measurement using precision, recall, F-measure, accuracy on python environment

## 4.1. Proposed Study Model for Training and Testing

Neural feature extractor is composed of an input layer, a hidden layer, and an output layer [2, 16] and it is responsible for creating dense vector representations for each word features [16].

The Input Layer takes the vector representations of each word as input in the corresponding word features. Each word features is a sequence of tokens, represented as a vector. The vector representation for each token is composed of the word embedding corresponding to the token, concatenated with a one-hot representation of the token's POS tag [2, 6].

The hidden layer takes the token vectors as input, and processes them in a forward and a backward passes. In the forward pass, the content value of the hidden layer at a specific time-step is calculated using the value of the input at the current time-step, and the content value of the hidden layer in the previous time-step at the backward.

The output layer is a function that automatically assigns weights to the output of the hidden layer at each time-step, and calculates the weighted sum of the outputs using their corresponding weights using softmax function.

There is as follows the flow diagram of the proposed approach framework, wherein the input corpus is fed to the Word2Vec skip-gram model and Bi-LSTM RNN model besides to the N-gram tagger. So that this study is comparing the relative performance of neural network and the stochastic N-gram tagger approaches.

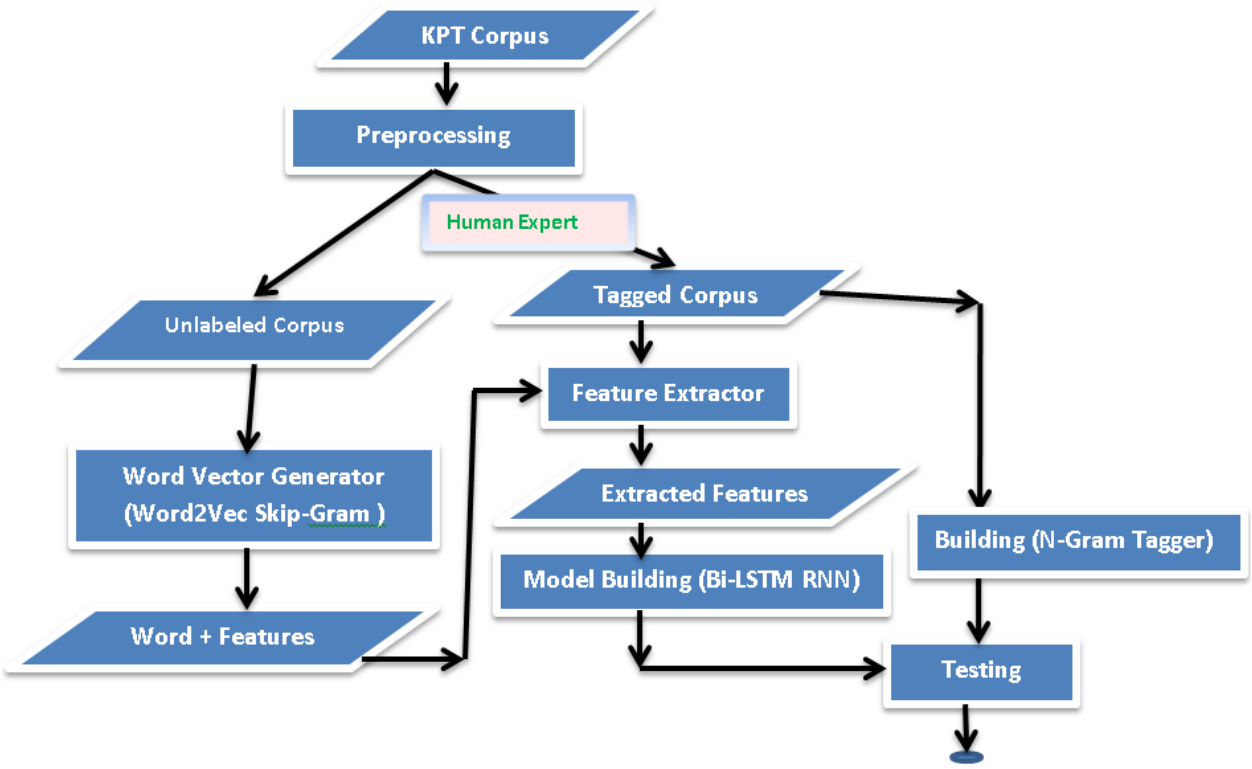



Figure 4.1: Proposed Approach Framework

## 4.2. Development Tools

This study has used the following Python programming language packages and libraries on Anaconda Jupyter Notebook, which used as an editor environment.

### A) ANACONDA

It is a package manager, an environment manager, and Python distribution that contains a collection of many open source packages (nltk, numpy, scikit-learn, scipy, pandas, keras, tensorflow, matplotlib, gensim, …, etc) after installation.

**PANDAS:** This study experiment tools take the corpus in a CSV or TXT format, and creates a Python object with rows and columns called a data frame. It does indexing and plot data with histogram or box plot in vector space. In this study, it is used to handle data frames of the corpus in a tabular representation.

**NUMPY:** This study experiment tools used it to handle the array list of train and test corpus.

**SCIKIT-LEARN**: As it bases on NumPy and SciPy, this library provides algorithms for handling the performance of the metrics evaluation for accuracy, precision, recall, and F1-measure of the model.

**MATPLOTLIB**: As it is the plotting library it is used in this study to sketch the word vector similarity distribution at vector space.

**NLTK**: In this study, this library is used to tokenize untagged words.

**GENSIM**: As it is a Python library for robust semantic analysis and vector-space modeling, it is used in this study to incorporate word2vec library.

**KERAS**: As it is a high-level library for working with neural networks running on top of TensorFlow. It is used in this study to represent neural word embedding layers and model libraries at the front end of TensorFlow. This means, it is used to develop the recurrent neural networks easily. It is user friendliness, modular, and extensible for easy and fast prototyping [8].

**TENSORFLOW**: As it is a popular framework for deep neural network and machine learning with multiple data sets in numerical computation [34], it is used to represent neural word embedding layers and model libraries at the back end of Keras application. It strongly supports machine learning and deep learning at the backend of Keras application program interface (API), and also has flexible numerical computation core that can be used in other scientific domains [8].

**B) Bi-LSTM RNN model**

In this study, it is used to process deep neural network learning based on the keras and tensorflow libraries. Since Bi-LSTM is an effective model for sequence labeling in deep neural network learning. Given a sentence $W_1$, $W_2$,…,$W_n$ with tags $y_1$,$y_2$, …, $y_n$, Bi-LSTM is used to predict the tag probability distribution of each word. Bi-LSTM layer incorporates information from the past and future histories when making prediction for current word, and is updated as a function of the entire input sentence. Bi-LSTM consists three layers per function such as the activation function of input layer is identity function; hidden layer is logistic function, while the output layer uses softmax function for multiclass classification [2]. RNN is a kind of ANN that contains cyclic connections, which can model contextual information dynamically for sequence labeling applications. A sequence of vectors is given for the RNN network and returns another sequence. Its preference for using RNN in sequence labeling comes from its ability to connect the previous information with the present task [8].

The proposed study uses Bi-LSTM RNN word embeddings to build the words mapped to its corresponding vectors since sequence labeling is done from fore-and-aft of LSTM passes. Besides, Word2Vec software package is used with its skip-gram algorithm to map word(s) to the target variable, which is also a word. This technique learns weights, which act as word vector representations. For getting embedding of a word, Bi-LSTM RNN multiplies the one-hot vector |V| of each individual word by the embedding weight matrix (or word vector lookup

table) in hidden layer. The individual word's embedding taken by looking at ones index number position in V-dimensional rows and the word vector lookup table of hidden layer. It uses predictive analysis from a lookup table to make a weighted guess of a word co-occurring with respect to its neighboring words proximity in the entire space.

This means for training, instead of dictionaries and morphological features, neural word embedding uses the hidden layer lookup table of Bi-LSTM RNN, then POS tagging is initialized with the pre-trained word embedding [2]. Bi-LSTM RNN can then benefit from word embedding trained on large unlabeled corpus, and larger training corpus leads to achieve better performance accurately.

Because of the LSTM's ability to learn long-term dependencies and the use of neural properties, Bi-LSTM RNN predict hidden semantic relationships between words based on word analogies and word similarity in vector space proximity. It uses predictive analysis from a lookup table to make a weighted guess of a word co-occurring with respect to its neighboring words proximity correlation in the entire space. Words with similar distributional properties in vector space have similar meaning.

C) **N-Gram Tagger Statistical Approach**

This study used N-Gram statistical based learning approach to compare the performance relatively and empirically to the neural network performance.

### 4.3. Corpus Development

The corpus is collected from Koorete dictionary, New Testament Bible of Koorete, Amaro Wereda Culture and Tourism Office, Book of '**Dicchoo'** published in 2006 E.C[38], Amaro Wereda Education Office, and Koorete language department at Dilla College of Teacher Education. The Koorete dictionary has embedded-in tags of Koorete part of speech. Most portion of the KPT corpus is prepared from two sources. The first source is Matthew Gospel of New Testament Bible of Koorete and the second one is Book of 'Dicchoo'. Book of 'Dicchoo' is translated from Amharic to Koorete by Tizita Fabric, who was a graduate student of Koorete language at Dilla College of Teacher Education in 2012 E.C. This book narrates about Koore cultures, norms, stories, heritages and jewlery, municipality and Kebeles structure, Koore Kings administration hierarchy, cultural food preparation, and staple food of Koore termed 'Inset' (or '**Shuncha'** in Koorete). The other source is about Koore Kings funeral system including its material which is house-like shaped called '**Komboxe'**, wailing ceremony, weeping orchestra during burial, burial style, and its condolences. Besides, the other source is the marriage process from dowry to wedding in Koore people culture. Finally, the corpus is presented to Dr. Samuel Zinabu, who is the Koorete language expert. He had delivered Koorete language at Hawassa College of Teacher Education for Koorete department students until the Koorete department shifted to the Dilla Teachers Training Center (TTC). The size of corpus prepared has **1718** sentences (***33220*** words). This large size is why the corpus of Bi-LSTM RNN developed above needs minimum size of **10k** for enough data usage, and for efficient performance on Skip-gram algorithm.

### *4.3.1. Koorete Language Corpus Tag sets*

This corpus preparation is written using Koorete language in Latin alphabets, and its annotation is done by the help of the language experts at College of Teachers' Education at Hawassa and Dilla. According to the preparation of this corpus, the number of tag sets consist about **11** original tagsets, and expanded to **24** tagsets. There has not been any other tag sets prepared yet in a documented and computerized fashion. So the following table 4.1 of KPT tag sets are cascaded from the analysis of Koorete Segmental phonology [17], Part of Speech Tagging for Wolaita Language using Transformation Based Learning (TBL) Approach[22], and Morphology of Koorete[28] for the relativity representation. Because Wolaita language has relatively similar phonetics in addition to the languages are written in Latin alphabets.

Table 4.1: The KPT Tagset having **11**-original and expanded **24**-tagsets

| No | Basic Tags | Derived Tags | Definition | Example |
|---|---|---|---|---|
| 1 | Noun | NN | Noun | Adey **accho**/NN muudo (Father ate a meat). |
| | | NPL | Plural Noun | Gerri meqet**ita**/NPL uqusuwa (You can approach intimacy with genealogical descendence). |
| | | CAN | Ablative Case | Guliixhi**afa**/ACN zayte nu degesso. |
| | | LCN | Locative Case | Buushe qaam'o utulma**ka**/LCN gusuwa. |
| | | CCN | Commutative Case | Ade wonto**ra**/CCN kardiitofu. |
| 2 | Pronoun | PRPRON | Proper Pronoun | **Hawassa**/PRPRON katamay/PRPRON orijheena akooko (The town of Hawassa is very vast). |
| | | DSPRON | Distributive Pronoun | Neera **Wola**/DSPRON handzo gaddhesa hanguwaso (Today I wil not go to the market). |
| | | DMPRON | Demonstrative Pronoun | **Yede**/DMPRON kanay woxhenike beewa(Look how that dog is running). |
| | | IRPRON | Interrogative Pronoun | Doo **ayiidewu**/INTPRON ne yoodo (When did you come)? |
| 3 | Adjective | ADJ | Adjective | **Jhileta**/ADJ maatawo wonguwa (Buy green grass only). |
| 4 | Adverb | ADV | Adverb | Ne busshi **iitanako**/ADV modhe (Your girl is very beautiful). |
| 5 | Verb | VB | Verb Simple | Ne busshi iitanako **modhe**/VB (Your girl is very beautiful). |
| | | VBP | Verb Phrase | Ne bushanchey iitanako modho**sso**\|**oose**/VBP (Your girls are all very beautiful). |

| | | | | |
|---|---|---|---|---|
| | | VBG | Verb Gerund | Zine hano wotiyako nu **yaaca**/VBG (Yesterday this time, we were digging) |
| | | VBF | Verb Future | Haya zawa hanta**ko**/VBF ta han**te** |
| 6 | Conjunction | SCONJ | Subordinate Conjunction | Yede **badena**/SCONJ e'uwa (stand aside to the below position). |
| | | CCONJ | Correlative Conjunction | Muwa **ooyne**/CCONJ denxuwa (Eat or else take it off). |
| 7 | Numerical | NUMO | Ordinal Number | Handzoy **lanxo**/NUMO wontako (Today is 4th day). |
| | | NUMC | Cardinal Number | Handzo **lam'i**/NUMC keexita nu keexho (Today we made two houses). |
| | | CD | Cardinal Decimal | 36.96 |
| 8 | Preposition or Postposition | PREP or POST | Preposition or Postposition | Horobhilay yeke saha **e**/PREP hodho (Airplane landed to the land). |
| 9 | Punctuation | PUNC | Punctuation | Se axi modheko**.**/PUNC |
| 10 | Interjection | IJ | Interjection | **Eyyi**/IJ aba iita axewu! |
| 11 | None | NONE | None | Used for words having no tags in the corpus |

### *4.3.2. Corpus Preprocessing*

#### 4.3.2.1. Corpus Cleaning

In this study preprocessing includes the task of cleaning, tokenization, and feature subsampling activities. Cleaning includes the task of removing unwanted characters such as correcting many misspelled words, tagging inconsistencies, miss-tagged words, and miss-repeated special characters.

Table 4.2: Unique Tagsets instances obtained after Corpus Cleaning

| Nº | POS Tag Sets | Number of total frequency of instances counted in a KPT corpus |
|---|---|---|
| 1 | NN | 7596 |
| 2 | VB | 6821 |
| 3 | PUNC | 5090 |
| 4 | PRPRON | 2936 |
| 5 | PREP | 2153 |
| 6 | VBP | 1844 |

| | | |
|---|---|---|
| 7 | ADJ | 1765 |
| 8 | NUMC | 1741 |
| 9 | SCONJ | 1361 |
| 10 | NPL | 608 |
| 11 | ADV | 403 |
| 12 | DMPRON | 242 |
| 13 | NONE | 164 |
| 14 | RLPRON | 137 |
| 15 | CCONJ | 135 |
| 16 | DSPRON | 44 |
| 17 | LCN | 39 |
| 18 | NUMO | 37 |
| 19 | CAN | 28 |
| 20 | CCN | 21 |
| 21 | VBG | 17 |
| 22 | IRPRON | 16 |
| 23 | IJ | 13 |
| 24 | VBF | 9 |
| Total Number | | 33, 220 |

#### 4.3.2.2. Vocabulary, Tokenization

In this study, tokenization is used in the processes of word vector generation and feature extraction. Since tokenization is the process of splitting sentences, phrases, words and their word class tags from one another to make them suitable for analysis.

The following figure depicts the statistics of Koorete tagged and untagged numerical representation of the language corpus. As it is in figure the total number of tagged corpus is of 33,220 words, 24 tags, VB as most common tag, 1546 training and 172 testing sentences; while the untagged corpus is of 33,220 words.

```
----- Tagged Corpus Statistics ------
Number of Tagged Sentences:  1718
Total Number of Tagged words:  33220
Number of Tags in the Corpus:  24
The most frequent tag is:  ('NN', 7596)
Number of Sentences in Training Data:  1546
Number of Sentences in Testing Data:  172
```

Figure 4.2: Statistics of KPT corpus

The following figure shows the assignment of word index to each tokens of a vocabulary in a tokenized corpus. This indexing is quite important in word vector generation and feature extraction after one-hot encoding and padded sentences. Indexation is helpful in the process of appending the tags and extracted features to its correspondent respective words from the generated word vectors and the original tagged corpus, respectively so as to realize the study.


Figure 4.3: Tokenized word indexes

Generating the word and its correspondent tag together for word indexing is useful for searching the word from the tagged corpus based on the indices, and mapping the vector feature generated with its word class.

### 4.4. Bidirectional LSTM RNN Word Embedding Stages

Word embedding using bidirectional LSTM model building on untagged KPT corpus could be done according to the following algorithm steps and stages in order to have an extracted input feature for deep neural network to accept either in CSV or TXT file format.

Steps:

Step 1: Prepare vocabulary |V|

Step 2: Prepare one-hot encoding and one-hot padding

Step 3: Prepare embedding weights based on one-hot encoding

Step 4: Feed length of unique tokens or word2indx as the number of neurons and embedding weights as embedding dimension output to the model → Create Architecture

Step 5: Compile, fit and build the model →Compute model

Step 6: Summarize model summary

Step 7: Evaluate model accuracy

Step 8: Test the model

Step 9: Plot word2vec and word pairs in vector space using semantic similarity distance

Step 10: Associate ‘word2vec’ to part of speech ‘tags’ in tagged corpus to know word class of the generated word vector

# Chapter 5. Experiment, Result and Discussion

So far this study has mentioned the two empirical language modeling evaluation approaches; those would be practiced on experimentation, which is listed under chapter 4 on the methodology parts, and why these approaches are chosen narrated at the end of chapter 3 after the literature review summary Table 3.1. These two chosen POS tagging approaches experiment would be demonstrated according to here after bulleting and numbering orders.

## 5.1. Experimental Installed Hardware and Software Specifications

The hardware used during this study is minimum sized Intel CORE™ i5 vPro™ 830 laptop, and the operating system used is Windows 10 Enterprise.

Table 5.1: Used devices specifications

| Device specifications | | Windows Specifications | |
|---|---|---|---|
| Device name | DESKTOP-INELHIK | Edition | Windows 10 Enterprise |
| Processor | Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz 2.50 GHz | Version | 20H2 |
| Installed RAM | 4.00 GB (3.88 GB usable) | OS Build | 19042.985 |
| System Type | 64-bit operating system, x64-based processor | Experience | Windows Feature Experience Pack 120.2212.2020.0 |

For experimental set up Python 3.7.1(Anaconda3 2018.12 64-bit) IDE is installed, and the study used its source code an Anaconda3 Jupyter Notebook as editor opening on Yandex browser. For KPT corpus edition Notepad++ was used; for document formatting purpose Microsoft Office Professional plus 2010 is used; and for information searching Google Chrome browser was engaged.

## 5.2. Input Corpus Preparation

At the time of experimentation the KPT corpus must be divided into two parts (training data 90% and testing data 10%) before the model is being built for Bi-LSTM RNN and N-gram tagger. The corpus is packaged about 1718 sentences, 33220 words, 11-original and 24-expanded tagsets, and NN is the most frequent word in the corpus. The training sentences is about 1546, and testing sentences is about 172.

It is significantly an advantage to take raw text for input of word vector (word2vec) generator and for input of N-gram tagger validation purpose. This implies Koorete untagged corpus has to be prepared from a copy of KPT corpus since the input of word vector generator and the output of the word vector generator represents the word features in real number from the given corpus. Then, by extracting features, the corpus input is prepared into two formats (.CSV, .TXT). In this study, the CSV (comma separated vector) format or TXT (text) format are either used as input of the deep neural network (Bi-LSTM RNN) to balance the number distribution of tagsets in vector space.

## 5.3. Define the Neural Model Parameters

This portion depicts about the significant model of neural word embedding algorithms using them by exchanging one model in place of another model, and what activation functions are also included in this study. This means, as the python code segment below shown, we could run Anaconda Jupyter Notebook IDE by replacing LSTM, Bi-LSTM instead of SimpleRNN on the beneath segment. From the activation functions, rectified linear unit (ReLu) and Softmax functions are used in the following code segment on Figure 5.1. Activation functions activate the neurons, which are used to determine the output of neural network on 0, and 1 like yes or no alternatives. It maps the resulting values between 0.0 to 1.0. Dropout reduces dependency between neurons; optimizer 'adam' is used with its default learning rate 0.001 for both Keras and Tensorflow alpha parameter.

ReLu rectifies the values of inputs less than zero forcing them to zero and eliminating the vanishing gradient problem. Softmax function is used in the output layer to compute probability distribution from word vector of real numbers [6]. Then it produces an output which is a range of values between 0 and 1, with the sum of the probabilities being equal to 1. This softmax function gives a tag probability distribution of input word, and its dimension is the number of tag types [2].

- Activation functions activate the neurons, which are used to determine the result mapping values of neural network between 0.0 and 1.0.
- For learning rate Adam optimizer is used with its default name on alpha parameter 0.001 for both Keras and Tensorflow. And for loss estimation 'binary cross entropy' is used

Take consideration to the parameters utilized on this study at appendix (**D**).

## 5.4. Word Vector Generation and Cosine Similarity Vector Space Representation

Word2Vec includes Skip-gram and CBOW algorithms but Word2Vec by itself is neither a deep neural network nor an algorithm; but it maps a given text into a suitable numerical format that the deep nets can understand with limited dimensional vector spaces. Word2vec is a two-layer neural network designed to process an input text and give a set of vectors as an output. Then, the words are stored with their corresponding vector representation to be used for feature extraction. By default word2vec model parameter has sg = 1 on which Skip-gram algorithm is employed, otherwise sg = 0 for CBOW algorithm. This type of algorithm threshold is

determined in the appendix code segment for Skip-gram algorithm with its context window size 5.

The plotting here below is sketched based on the figure appendix code segments to show the distribution of words with their semantic similarity vectors using real number in the vector space. This means the graph tells us words are represented in numbers instead of words itself in the vector space. Word2Vec is demonstrated on Word2Vec Skip-gram algorithm as shown on (5.3) by extracting word features in real numbers, and by sketching the words-to-words distance of word vector generation. This practice is the key part of this study to show the syntactic and semantic similarity of words based on the cosine similarity distance to distribute words in vector space.

```
[('kuulame', 0.5666407346725464),
 ('fuulay', 0.5613170266151428),
 ('maadhese', 0.5229238867759705),
 ('yesse', 0.1210478842584534),
 ('hayiigoko', 0.0807025358080864),
 ('udiwogaka', 0.058135561645030975),
 ('woosute', 0.050018344074487686),
 ('toranawoka', 0.048183459788560 87),
 ('dandadhaasi', 0.037521205842494965),
 ('siiye', 0.030091479420661926)]
```

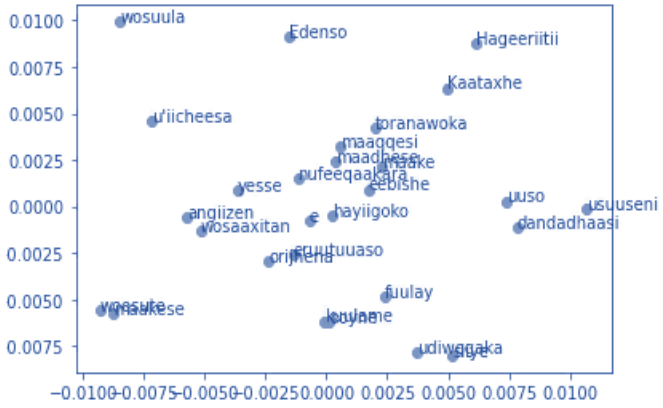



Figure 5.1: Words Semantic similarity distribution in vector space

The word vector is generated by using the key word “most-similar” on the model. wv.Most_similar(parameters) during the measurement of distance of words to words. This distance is called cosine similarity or cosine distance based on the angle in space inclination of words to words.

## 5.5. Feature Extraction

Features are the indivisible distinctive properties of input patterns that help in differentiating between the classifications of input patterns. This study has established the ways how Koorete POS tagging could be predicted using neural word embedding algorithms, and discussion could be done accordingly to the following algorithm designed steps.

Step 1: Arrange the words and corresponding tags in the corpus into ArrayLists of ‘Words’ and ‘Tags’ keeping the indices the same.
Step 2: Take a word from the ArrayList of ‘Words’
Step 3: Use the word to search its corresponding feature from the generated features
Step 4: Copy the line containing the word with its features
Step 5: Append the tag located at the same index at the end of the copied line
Step 6: Write the line into a third file containing the extracted features.
Step 7: Loop to step 2 until the last index of the ArrayLists.

The feature extractor searches for the corresponding ‘word features’ to ‘tags’ in the corpus from the indices matching in ‘word-tag-index’ mapping as shown in Figure (5.3) on tokenized words

indexes. These words indices are used for searching the words and its correspondent tags like the corpus prepared here below, and its word vector features with POS tags appended.

Edenso/CONJ maakese/VB dandadhaasi/VB orijhena/ADV usuuseni/PRON nufeeqaakara/PRON yesse/VB wosuula/NN hayiigoko/VB ./PUNC
Hageeriitii/NN maadhese/VB angiizen/ADV maaqe/CONJ udiwogaka/ADV toranawoka/NN u'iicheesa/VB uuso/PRON eebishe/VB ./PUNC
Kaataxhe/NN fuulay/ADJ kuulame/NN e/PREP maaqqesi/VB wosaaxitan/NN woosute/VB ooyne/CONJ siiye/VB eruutuuaso/VB ./PUNC

The following Figure 5.2(a) is a code segment used to generate word feature with its POS to determine the word class, for instance the word similarity of other words, to word 'wosuula' as the output shown here.

```
from gensim.models import Word2Vec
from keras.layers import Input, Concatenate, Reshape
from keras.layers import Embedding
import numpy as np
max_len = 10 ## To balance the defualt maximum size of command 'modelo.wv.most_similar'
tagss = set([tag for (word,tag) in tagged_words])
pos = Reshape ((max_len, 1)) (tagss)
lst2 = list([pos]) ## call tags
space = np.full(fill_value=' ',shape=len(modelo.wv.most_similar('wosuula')),dtype=object)#b/n word & feature
result = np.array(modelo.wv.most_similar('wosuula'), dtype=object) + space + np.array(lst2, dtype = object)
print(result)
```

Figure 5.2(a): Feature extraction code segment

```
[('kuulame', 0.5666407346725464), NN
 ('fuulay', 0.5613170266151428), NN
 ('maadhese', 0.5229238867759705), VB
 ('yesse', 0.12104788422584534), VB
 ('hayiigoko', 0.0807025358080864), VB
 ('udiwogaka', 0.058135561645030975), ADV
 ('woosute', 0.05001834074487686), VB
 ('toranawoka', 0.04818345978856087), NN
 ('dandadhaasi', 0.037521205842494965), VB
 ('siiye', 0.030091479420661926), NN
]
```

Figure 5.2(b): Extracted Word Vector features with tags appended at the end

After the extraction of words features, the corpus is separated into training data 90% and testing data 10% proportion ratio for model building testimonial purpose. The Bi-LSTM RNN based neural word embedding POS tagging experiment has built based on this division of data as shown below on LSTM Figure (5.6).

## 5.6. Building Bi-LSTM RNN Model

The model deep learning is being built using long-short term memory recurrent neural network word embedding. The Bi-LSTM model shot here sampled example shows the Word2indx value 7182 as input for the input layer, the tag2indx value is 13 for dense layer, the Max_length value is 57 and 128 as the word embedding dimension for output layer using the softmax function.

The values fed to the sequential LSTM model are the length of Word2indx value is 7182, length of tag2indx value is 13 for dense layer, Max_length is 57, and embedding dimension valued 128.

```
Embedding parameter = (I*O) = 7262 * 128 = 929536
 LSTM parameter = 4*((X+h)*h+h) = 263168
```
Dense parameter = I*h+h = 128*13+13 = 6425

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_layer (Embedding)  (None, 57, 128)           929536
_________________________________________________________________
bidirectional (Bidirectional (None, 57, 256)           263168
_________________________________________________________________
dense_2 (Dense)              (None, 57, 25)            6425
_________________________________________________________________
dropout_2 (Dropout)          (None, 57, 25)            0
=================================================================
Total params: 1,199,129
Trainable params: 1,199,129
Non-trainable params: 0
_________________________________________________________________
None
Train on 1236 samples, validate on 310 samples
Epoch 1/10
1236/1236 [==============================] - 11s 9ms/sample - loss: 0.3004 - accuracy: 0.9600 - val_loss: 0.1464 - val_accuracy:
0.9600
Epoch 2/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2561 - accuracy: 0.9688 - val_loss: 0.0635 - val_accuracy:
0.9847
Epoch 3/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2335 - accuracy: 0.9772 - val_loss: 0.0574 - val_accuracy:
0.9839
Epoch 4/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2267 - accuracy: 0.9775 - val_loss: 0.0558 - val_accuracy:
0.9849
Epoch 5/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2257 - accuracy: 0.9777 - val_loss: 0.0547 - val_accuracy:
0.9852
Epoch 6/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2233 - accuracy: 0.9775 - val_loss: 0.0471 - val_accuracy:
0.9863
Epoch 7/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2218 - accuracy: 0.9776 - val_loss: 0.0458 - val_accuracy:
0.9863
Epoch 8/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2183 - accuracy: 0.9774 - val_loss: 0.0448 - val_accuracy:
0.9862
Epoch 9/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2179 - accuracy: 0.9775 - val_loss: 0.0972 - val_accuracy:
0.9826
Epoch 10/10
1236/1236 [==============================] - 7s 6ms/sample - loss: 0.2377 - accuracy: 0.9764 - val_loss: 0.0463 - val_accuracy:
0.9856
1546/1546 [==============================] - 4s 2ms/sample - loss: 0.0470 - accuracy: 0.9854
Accuracy: 98.536819
```

Figure 5.3: Bi-LSTM built model

As the training has done for 10 epochs, we observe that the loss function is decreasing and accuracy rate increasing. This implies the training is striving to reach to the optimal learning point, and we observe the loss function went down on epoch 10 from 0.2377 to 0.0470 with big difference. This truth tells us that the training has reached its optimal learning point.

## 5.7. Building N-gram Based Statistical Model

This study of N-gram taggers approach appends the lower N-gram taggers result using 'backoff' parameter. The reason why this study has done until Trigram tagger is that the N-gram taggers beyond Trigram tagger does not perform better results than the Trigram tagger performance. The most frequent tags in the prepared KPT corpus tagsets are the verbs about ('NN', 7596); whereas the second most frequent tags are nouns about ('VB', 6821). The following Figure 5.6 and Table 5.2 shows the tokenized POS tagging of raw text fed to the N-Gram statistics learning until the Trigram tagger for the sake of validation checking. This approach has evaluated for both the training and testing data of the tagged corpus for each tagger as follows:

Table 5.2: N-gram tagger accuracy for training and testing data

| No | N-gram taggers | Accuracy of Training | Accuracy of Testing |
|---|---|---|---|
| 1 | Default Tagger | 23.57 | 17.04 |
| 2 | Regexp Tagger | 22.62 | 18.77 |
| 3 | Unigram Tagger | 93.75 | 78.30 |
| 4 | Bigram Tagger | 95.56 | 78.05 |
| 5 | Trigram Tagger | 97.10 | 77.29 |

The result from the table shows that default tagger and Regexp tagger achieved approximately proximate value on training and testing data; whereas unigram tagger, bigram tagger and trigram tagger achieved the accuracy with big magnitude difference between training and testing data. This informs us the two lower-level taggers work well on this study, and for the next three higher-level taggers it needs extra training on the tagsets to represent the frequency of each tags instances accurately. But for the training data the three higher-level taggers did much better accuracy than the lowers, especially trigram tagger performed best for about 97.10.

```
----------------New Tagged Words for validation----------------
[('bargoray', 'NN'), ('fetto', 'ADV'), ('yeyiidiwo', 'VB'), ('modha', 'NN'), ('wo', 'DMPRON'), ('matdatse', 'ADJ'), ('yeng
uate', 'NN'), ('hazowafa', 'ACN'), ('gutafa', 'ACN'), ('.', 'PUNC'), ("Na'oko", 'VB'), ('axi', 'NN'), ('kaxhesi', 'NN'), (
'goduwa', 'NN'), ('woxii', 'NN'), ('tuceko', 'VB'), ('hida', 'NN'), ('.', 'PUNC'), ('lam', 'NN'), ("'", 'PUNC'), ('i', 'PR
PRON'), ('sumaako', 'VB'), ('e', 'PREP'), ('maake', 'VB'), ('tamay', 'NN'), ('mudosi', 'NN'), ('.', 'PUNC'), ('nayishara',
'CCN'), ('mataka', 'LCN'), ('nuwoyddho', 'VB'), ('.', 'PUNC'), ('Hate', 'ADV'), ('oggey', 'NN'), ('killongesi', 'NN'), ('o
dunxo', 'NUMO'), ('tapoko', 'VB'), ('.', 'PUNC'), ('Bidzi', 'NUMC'), ('hanta', 'NN'), ('beyisuwa', 'NN'), ('hade', 'SCONJ'
), ('na', 'NN'), ("'", 'PUNC'), ('i', 'PRPRON'), ('hantiyaaca', 'NN'), ('godofa', 'NN'), ('.', 'PUNC'), ('medhiyaaka', 'VB
G'), ('ta', 'PRPRON'), ('hando', 'VB'), (',', 'PUNC'), ('bursiyaaka', 'VBG'), ('ta', 'PRPRON'), ('woydho', 'VB'), (',', 'P
UNC'), ('sholiyaaka', 'VBG'), ('ta', 'PRPRON'), ('dendoyidiko', 'VB'), ('.', 'PUNC'), ('Abebey', 'NN'), ('beyisuwa', 'NN')
, ('kaadho', 'VB'), ('.', 'PUNC'), ('Hanzo', 'VB'), ('beyisuwa', 'NN'), ('nuhamese', 'VBF'), ('.', 'PUNC'), ('ade', 'NN'),
('haree', 'NN'), ('nuwondo', 'VB'), ('.', 'PUNC'), ('asafi', 'NN'), ('iotosi', 'NN'), ('lelisha', 'NN'), ('giwino', 'VB'),
('.', 'PUNC'), ('nuuso', 'VB'), ('hanzo', 'ADV'), ('oydi', 'NUMC'), ('56', 'NUMC'), ('na', 'NN'), ('34.67', 'CD'), ('maayi
idita', 'NPL'), ('nu', 'PRPRON'), ('one', 'NN'), ('.', 'PUNC'), ('wergusoko', 'VB'), ('e', 'PREP'), ('werguse', 'VBP'), ('
.', 'PUNC'), ('Yebo', 'NN'), ('ha', 'ADJ'), ('muudosso', 'VB'), ('madhoose', 'NN'), ('saha', 'NN'), ('aallicira', 'NN'), (
'aallo', 'SCONJ'), ('.', 'PUNC'), ('ha', 'ADJ'), ('zawaka', 'NN'), ('e', 'PREP'), ('gidaka', 'SCONJ'), ('tobe', 'NN'), ('e
seyi', 'NN'), ('sahafa', 'NN'), ('e', 'PREP'), ('denxo', 'NN'), ('shuchara', 'CCN'), ('wola', 'DSPRON'), ('e', 'PREP'), ('
muhurosa', 'NN'), ('minxheko', 'VB'), ('.', 'PUNC'), ('Yede', 'NN'), ('kemuwaka', 'LCN'), ('lukuliyo', 'VB'), ('ta', 'PRPR
ON'), ('wondo', 'VB'), ('hineko', 'VB'), ('hinuni', 'PRPRON'), ('kaxa', 'NN'), ('miyako', 'VB'), ('hiyeca', 'VBPC'), ('.',
'PUNC')]
```

Figure 5.4: N-Gram tagger for raw text validation

For new corpus checkup a raw text is prepared, and validated as the following experiment output for N-gram tagger. Over the above output segment, the patterns for regular expressions are performed well such as ACN, VBF, LCN, VBG, NPL, VBP, and CCN.

## 5.8. Model Evaluation

The model evaluation metrics used to measure this study model are accuracy, precision, recall, and F-Measure. The measures obtained from confusion matrix to determine the metrics are True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). The accuracy score using the sklearn library obtained as from Figure (5.7) based on the number of total words of the Koorete language corpus in this study given to the weighted Word2Vec package. The Word2Vec is used to prepare the computation in Bi-LSTM model upon the tagsets instances. For this the same size corpus words is given for the metrics holds true.

Accuracy = (TP+TN) / (TP+FP+FN+TN) = 0.98

Precision = TP / (TP+FP) = 0.98,

Recall = TP / (TP+FN) = 0.98,

F-Measure = (2*Precision*Recall) / (Precision*Recall) = 0.98.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0.0 | 0.99 | 0.99 | 0.99 | 32337 |
| 1.0 | 0.63 | 0.56 | 0.59 | 883 |
| accuracy | | | 0.98 | 33220 |
| macro avg | 0.81 | 0.78 | 0.79 | 33220 |
| weighted avg | 0.98 | 0.98 | 0.98 | 33220 |

Figure 5.5: Model for frequencies of tagsets instances

## 5.9. Experimental Result and Discussion

This study has just demonstrated assurance experiments to give an effective response for the following research questions, which was raised under chapter 1 that is outlined on (1.3). This research question is:

**RQ1**: How better could Bi-directional LSTM RNN based neural word embedding approach performs with respect to N-gram based POS tagging for Koorete language?

***[How could neural word embedding simplify Koorete POS tagging relative to N-gram based statistical approach?]***

Experimental demonstration has being done on **RQ1** by extracting word vector features, and splitting the KPT corpus to training data and testing data so as to perform neural word embedding for Koorete POS tagging. The same titled research study was done for Amharic POS tagging using neural word embedding in Ethiopia [8], and other researches on abroad

languages. The abroad languages research study has put using this deep neural word embedding model as the-state-of-the-art algorithm for POS tagging to achieve better performance on the sequence labeling tasks [9, 2, 3, 6]. And also N-gram based statistical approach demonstrated for evaluation of relative performance comparison to the neural one.

**RQ2**: What are the most appropriate tagsets need to be used for Koorete POS tagging?

**[*Adapt appropriate tagsets*]**

Experimental demonstration has being done on **RQ2** by adapting **24**- tagsets for Koorete POS tagging. These tagsets are adopted from research articles of Koorete Segmental Phonology [17], Morphology of Koorete [28], and Part of Speech Tagging for Wolaita Language using Transformation Based Learning (TBL) Approach [22]. The reason why this study also cascaded from the Wolaita language is in somewhat extent the Koorete language has relatively similar phonetics in addition to languages are written in Latin alphabets.

### *5.9.1. Experiment Demonstration*

This study is implemented basically on two experiments: (1) Bi-LSTM RNN neural word embedding model, and (2) N-gram based statistical tagger done on separate phases.

#### 5.9.1.1. Experiment on Bi-LSTM Models

An experiment is demonstrated on RNN, LSTM, and Bi-directional LSTM as shown on Figure (5.5). This experiment ensures either the neural word embedding model simplify implementation of the Koorete POS tagging to the state-of-the-art accuracy algorithm. For this sake, KPT corpus prepared in two copies, tagged corpus and untagged corpus a copy of the tagged corpus with equal sized data; but both intra-referred by the ArrayList indices in correspondence of words. The tagged corpus is used in Bi-LSTM RNN architecture manipulation tasks; whereas the untagged is used to encode syntactic and semantic information about words on Word2Vec Skip-gram algorithm.

#### 5.9.1.2. Experiment on N-gram based Model

Experiment on N-gram Tagger is demonstrated up to Trigram tagger using ‘backoff’ parameter to append the lower N-gram tagger so as to handle the previous information to the Trigram tagger for the ease of prediction over all the corpus data. This backoffing is used for the sake of N-gram taggers relativity comparison in either of achieves high accuracy. This study has limited the N-gram tagger up to Tri-gram tagger because the N-gram taggers beyond Tri-gram tagger do not bring the accuracy change any more. This study used N-gram tagger only for the purpose of comparing the accuracy performance on both N-gram tagger, and Bi-LSTM RNN model.

### *5.9.2. Results and Discussion*

#### 5.9.2.1. Results on Bi-LSTM RNN and N-gram based Statistical Model

The experiment demonstration fed of the same data size for SimpleRNN, LSTM, Bi-LSTM at the time of training so as to distinguish the performance difference in accuracy score, and also fed same data size at the time of testing the models. The experiment results recorded are the following accuracy scores as shown on table (5.3). In aspects of Bi-LSTM, there is a slight difference in accuracy occurrence for training and testing data. But there is big difference value between training and testing accuracy for Trigram tagger.

Table 5.3: Summary of accuracy for both training and testing data

| Nº | Model | Accuracy of training | Accuracy of testing |
|---|---|---|---|
| 1 | Bi-LSTM | 98.53 | 98.49 |
| 2 | N-gram(Trigram tagger using backoff parameter) | 97.10 | 77.29 |

In this study, the numbers of N-grams used are default, regular expression, unigram, bigram, and trigram taggers. Finally for performance determination, the Trigram tagger model achieved less effective score than the Bi-LSTM RNN model for both training data and testing data. Besides, Trigram tagger showed a value of big difference between the accuracy of training data and testing data on N-gram based tagger. Bi-LSTM model performed about 98.53% and 98.49% scores which is somewhat nearer approximation for accuracy of both training and testing data, respectively. Also the Bi-LSTM model achieved more effective score value than the N-gram based statistical tagging approach. This performance comparison amplifies that the truth of Bi-LSTM RNN word embedding POS tagging approach performs better than the N-gram statistical POS tagging approach. This truth is supported by Peilu Wang et al. [2] as Bi-LSTM RNN neural word embedding is a state-of-the-art performance of 97.40% tagging accuracy on Penn Treebank POS tagging. Besides, Erick Fonseca et al. [5] presented Portuguese POS tagging word embedding with achievement of the state-of-the-art performance with 97.57% overall accuracy.

# Chapter 6. Conclusion and Future Works

## 6.1. Conclusion

This research study testing is implemented on Koorete language. The language is spoken by Koore people who are located at Amaro Special Woreda, Southern Regional State of Ethiopia, and it has about 37-Latin alphabets. The language is categorized under Ometo language family [18]. It has about 350,000 speakers (Awoke, Koore People, 2020). The corpus is collected from Koorete dictionary, Matthew Gospel of the New Testament Bible in Koorete, Book of '**Dicchoo**' published in 2006 E.C, and Koorete language department at Dilla College of Teacher Education.

This study used the empirical evaluation of Bi-directional LSTM RNN based neural word embedding with respect to N-gram based statistical POS tagging approaches toanswer the research question how better to develop Koorete POS tagging depending the two approaches. So experiments were practiced on Bi-LSTM RNN model, and N-gram tagger statistical approach. For this, KPT corpus is used about size of 33220 words with 24 tagsets, and then divided this corpus into 90% training data and 10% testing data. The experiment on Bi-LSTM RNN word embedding POS tagging approach did better performance than the N-gram based statistical POS tagging approach with the accuracy of 98.53%. This shows Bi-LSTM neural word embedding has better potential on training and testing data accuracy. This truth is supported by Peilu Wang et al. [2] as Bi-LSTM RNN neural word embedding is a state-of-the-art performance of 97.40% tagging accuracy is achieved on Penn Treebank POS tagging. Besides, Erick Fonseca et al. [5] presented Portuguese POS tagging word embedding with achievement of the state-of-the-art performance with 97.57% overall accuracy.

## 6.2. Future Works

From the standing point of this study conclusion, Koorete language POS tagger is evaluated using deep learning neural word embedding and N-gram based statistical approaches for developing other high-level NLP applications. Other researchers can develop Koorete high-level NLP applications beyond POS tagging such as syntactic parsing, word-sense disambiguation, named-entity recognition, machine translation, morphological analysis, information extraction, …, etc. based on this POS tagging application as a pre-condition. Because POS tagging is a pre-requisite for these aforementioned NLP applications, this study has implemented the foot step for advanced NLP applications. Other researchers can use SimpleRNN and LSTM neural word embedding based POS tagging approach because these approaches have a competing performance with Bi-LSTM neural word embedding as features.

# References

[1] Beletu Redda. "The Morphology of Koorete". June, 2003 ADDIS ABABA and Wikipedia:http://en.wikipedia.org/wiki/Koorete_language & https://find.bible/languages/kqy

[2] PeiluWang1, 2, Yao Qian3, Frank K. Soong2, Lei He2, Hai Zhao1. **"**Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network." 1Shanghai Jiao Tong University, Shanghai, China. ArXiv:1510.06168v1 [cs.CL] 21 Oct 2015

[3] JohnWieting Mohit Bansal Kevin Gimpel Karen Livescu. **"**CHARAGRAM: Embedding Words and Sentences via Character n-grams**."** Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA. ArXiv:1607.02789v1 [cs.CL] 10 Jul 2016

[4] Diksha Khurana1, Aditya Koli1, Kiran Khatter1,2 and Sukhdev Singh1,2. " Natural Language Processing: State of The Art, Current Trends and Challenges" Manav Rachna International University 19 Jun 2018

[5] Erick R Fonseca*, Joao Luis G Rosa and Sandra Maria Aluisio. **"**Evaluating word embeddings and a revised corpus for part-of-speech tagging in Portuguese**"** Journal of the Brazilian Computer Society (2015) 21:2 DOI 10.1186/s13173-014-0020-x

[6] Bin Wang_, Student Member, IEEE, Angela Wang_, Fenxiao Chen, Student Member, IEEE, Yuncheng Wang and C.-C. Jay Kuo, Fellow, IEEE. "Evaluating Word Embedding Models: Methods and Experimental Results" arXiv:1901.09785v2 [cs.CL] 29 Jan 2019

[7] Pinky Sitikhu1, Kritish Pahi2, Pujan Thapa3, Subarna Shakya4."A Comparison of Semantic Similarity Methods for Maximum Human Interpretability." arXiv:1910.09129v2 [cs.IR] 31 Oct 2019. Tribhuwan University.

[8] Mequanent Argaw**. "**Amharic Parts-of-Speech Tagger using Neural Word Embeddings as Features**."** Addis Ababa Institute of Technology School of Electrical and Computer Engineering, AA (January, 2019).

[9] Anastasyev D. G., Gusev I. O., Indenbom E. M. "Improving Part-of-Speech Tagging via Multi-task Learning and Character-level Word Representations." ABBYY, Moscow Institute of Physics and Technology, Moscow, Russia Moscow, May 30—June 2, 2018

[10] NgoXuanBacha,b,*,NguyenDieuLinha,TuMinhPhuonga,b . **"**An empirical study on POS tagging for Vietnamese social media text**".** Science Direct Computer Speech & Language50 (2018)1_15 www.elsevier.com/locate/csl

[11] Mariya Koleva, Melissa Farasyn, Bart Desmet, Anne Breitbarth and Véronique Hoste. **"**An automatic part-of-speech tagger for Middle Low German.**"** (2018) Ghent University

[12] "Text Similarity Based on Word Embedding, Syntax Trees - YouTube (360p)").

[13] Amitha Mathew[1], P.Amudha[2] and S.Sivakumari[3]. "Deep Learning Techniques: An Overview." https://www.researchgate.net/publication/341652370. (August 2, 2020) Avinashilingam University

[14] Ludovic Arnold, Sébastien Rebecchi, Sylvain Chevallier, Hélène Paugam-Moisy. "An Introduction to Deep Learning" https://hal.archives-ouvertes.fr/hal-01352061 Aug 5, 2016

[15] Word2Vec in Deeplearning4j. (n.d.). Retrieved September 12, 2018, from Deep Learning for Java: https://deeplearning4j.org/docs/latest/deeplearning4j-nlp-word2vec

[16] Elham Mohammadi, Hessam Amini and Leila Kosseim. "Neural Feature Extraction for Contextual Emotion Detection" Proceedings of Recent Advances in Natural Language Processing, pages 785–794, Varna, Bulgaria, Sep 2–4, 2019. https://doi.org/10.26615/978-954-452-056-4_091

[17] ROLF THEIL. "Koorete segmental phonology". DOI: 10.1515/jall.2011.010 University of Oslo. 21 July 2016. https://www.researchgate.net/publication/269613870

[18] BELETU REDDA. "THE MORPHOLOGY OF KOORETE: Koorete Verb Morphology". Addis Ababa, Addis Ababa University. URL: http://localhost:80/xmlui/handle/123456789/6353 **Date:** 2003-06

[19] Ambrose Bangnia. "Challenges of the Teaching and Learning of French as a Foreign Language in Ghana: The Way Forward." University of Education, Winneba-Ghana Volume 4 Issue 01, January 2020

[20] Khwlah Alrajhi1 and Mohammed A ELAffendi2." Automatic Arabic Part-of-Speech Tagging: Deep Learning Neural LSTM Versus Word2Vec." ISSN (2210-142X) Int. J. Com. Dig. Sys. 8, No.3 (May-2019) Prince Sultan University, Riyadh, Saudi Arabia

[21] Serkan Ballı1, Onur Karasoy1." Development of content-based SMS classification application by using Word2Vec based feature extraction." ISSN 1751-8806 Accepted on 15th October 2018, Turkey doi: 10.1049/iet-sen.2018.5046 www.ietdl.org

[22] Birhanesh Fikre Shirko. "Part of Speech Tagging for Wolaita Language using Transformation Based Learning (TBL) Approach" Volume 10 Issue No.9 Wolaita Sodo University, Ethiopia. IJESC, September 2020

[23] Pinky Sitikhu1, Kritish Pahi2, Pujan Thapa3, Subarna Shakya4. "A Comparison of Semantic Similarity Methods for Maximum Human Interpretability" arXiv:1910.09129v2 [cs.IR] 31 Oct 2019

[24] NgoXuanBacha,b,*,NguyenDieuLinha,TuMinhPhuonga,b. "An empirical study on POS tagging for Vietnamese social media text". www.elsevier.com/locate/cs Computer Speech & Language 50(2018) 1_15, Vietnam

[25] Jabar H. Yousif. "Hidden Markov Model Tagger for Applications Based Arabic Text: A review" International Journal of Computation and Applied Sciences IJOCAAS, Volume 7, Issue 1, August 2019, ISSN: 2399-4509

[26] Tej Bahadur Shahi, Tank Nath Dhamala, Bikash Balami. "Support Vector Machines based Part of Speech Tagging for Nepali Text". Volume 70– No.24, May 2013

[27] Sam Goundar. "Research Methodology and Research Method" Victoria University of Wellington March 2012. https://www.researchgate.net/publication/333015026

[28] Beletu Redda. "The Morphology of Koorete" Master of arts in linguistics, Addis Ababa university. June, 2003

[29] PANTULKAR SRAVANTHI1, DR. B. SRINIVASU2. "SEMANTIC SIMILARITY BETWEEN SENTENCES" Volume: 04 Issue: 01 | Jan -2017 www.irjet.net p-ISSN: 2395-0072. Stanley College of Engineering and Technology for Women, Telangana- Hyderabad, India

[30] Faisal Rahutomo*, Teruaki Kitasuka, and Masayoshi Aritsugi "Semantic Cosine Similarity" Graduate School of Science and Technology, Kumamoto University 23 May 2014.

[31] wilson Gonzalo Rojas Yumisaca, Nancy Georgina Rodríguez Arellano, Nanci Margarita Inca Chunata, María Guadalupe Escobar Murillo. "ARTICULATORY PHONETICS IN THE

ENGLISH LANGUAJE PRONUNCIATION DEVELOPMENT" MAGISTER EN DOCENCIA UNIVERSITARIAY 2018

[32] Duressa Tamirat Gemeda. 'Corpus based Auto-Completion of N-GRAM WORD PREDICTION FOR AFAN OROMO WORDS' Adama, Ethiopia Septemper, 2016

[34] Kåre Hartlapp Lærum." A study of Machine Learning for Predictive Maintenance" Norwegian University of Science and Technology. June 2018

[35] Md. Mostafizer Rahman, Yutaka Watanobe and Keita Nakamura. "A Bidirectional LSTM Language Model for Code Evaluation and Repair". Symmetry 2021, 13, 247. University of Aizu, Aizu-Wakamatsu. https://doi.org/10.3390/sym13020247.

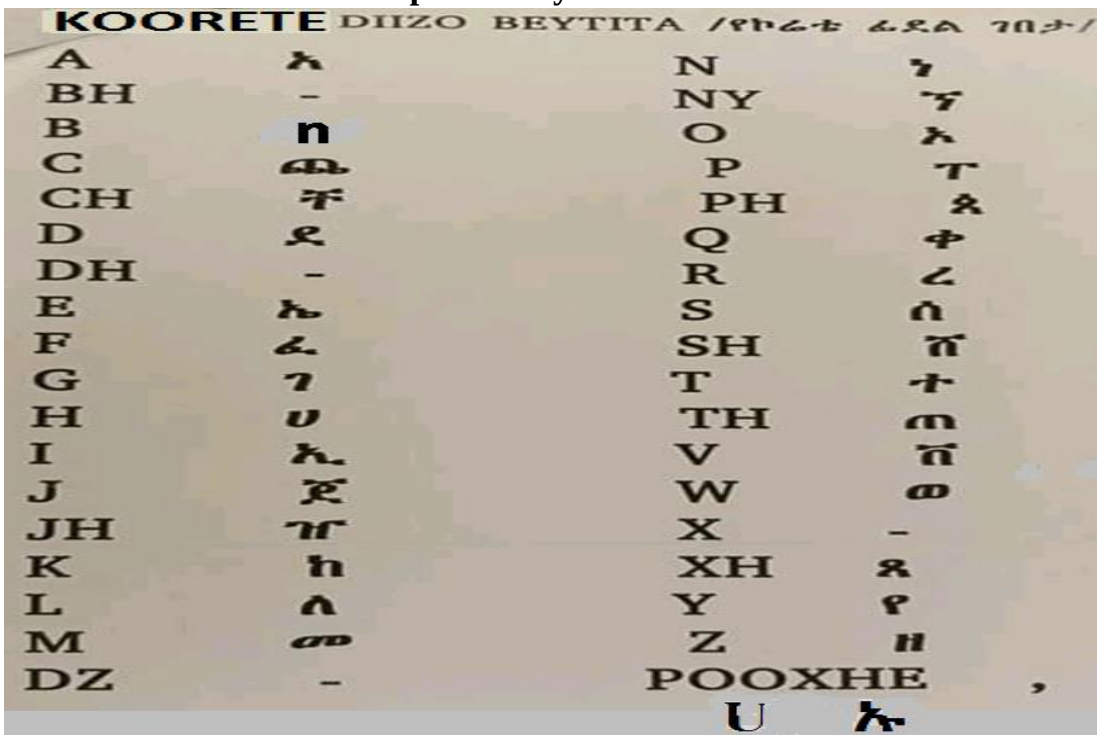
[36] NUNIYAT KIFLE ABEBE'.'WORD SEQUENCE PREDICTION FOR AMHARIC'. February 2011 Addis Ababa University, Ethiopia

[37] Grigori Sidorov1, Francisco Velasquez1, Efstathios Stamatatos2, Alexander Gelbukh1, and Liliana Chanona-Hernández3. 'Syntactic N-grams as Machine Learning Features for Natural Language Processing1'. Mexico, University of the Aegean,Greece MICAI 2012 www.cic.ipn.mx/~sidorov

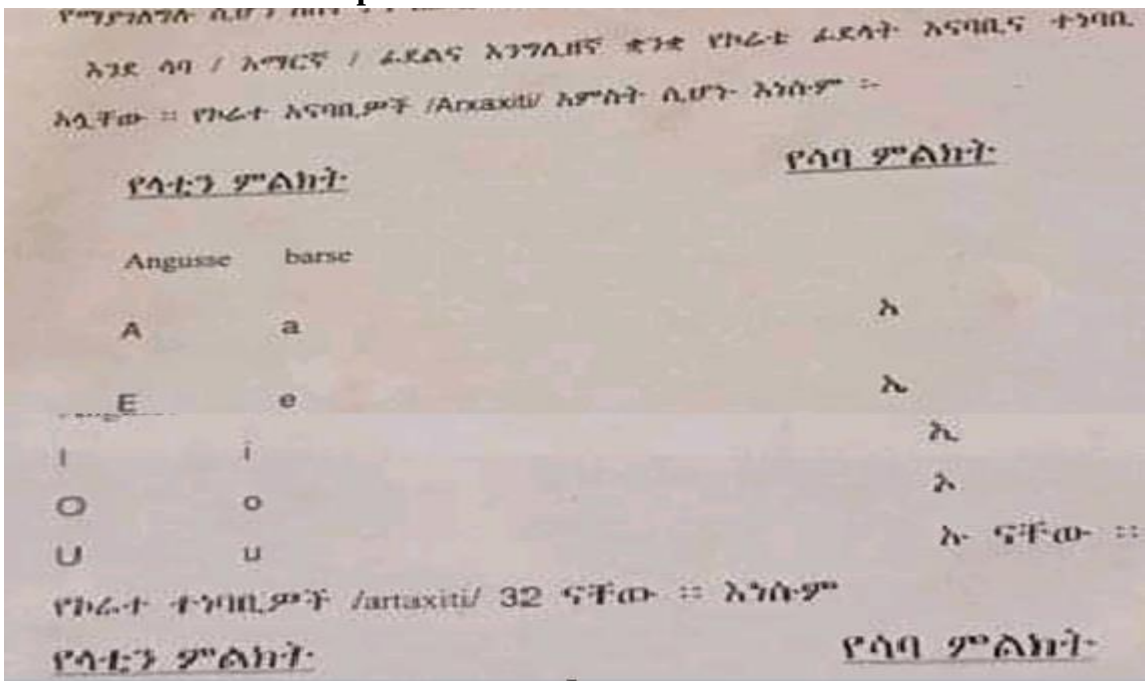
[38] Segen Area Peoples Zone Amaro Wereda Culture , Tourism, and Communication Office Annual Report "**Dicchoo**". 2006 E.C.

# Appendix

## A. The Basic Koorete Alphabets Symbol

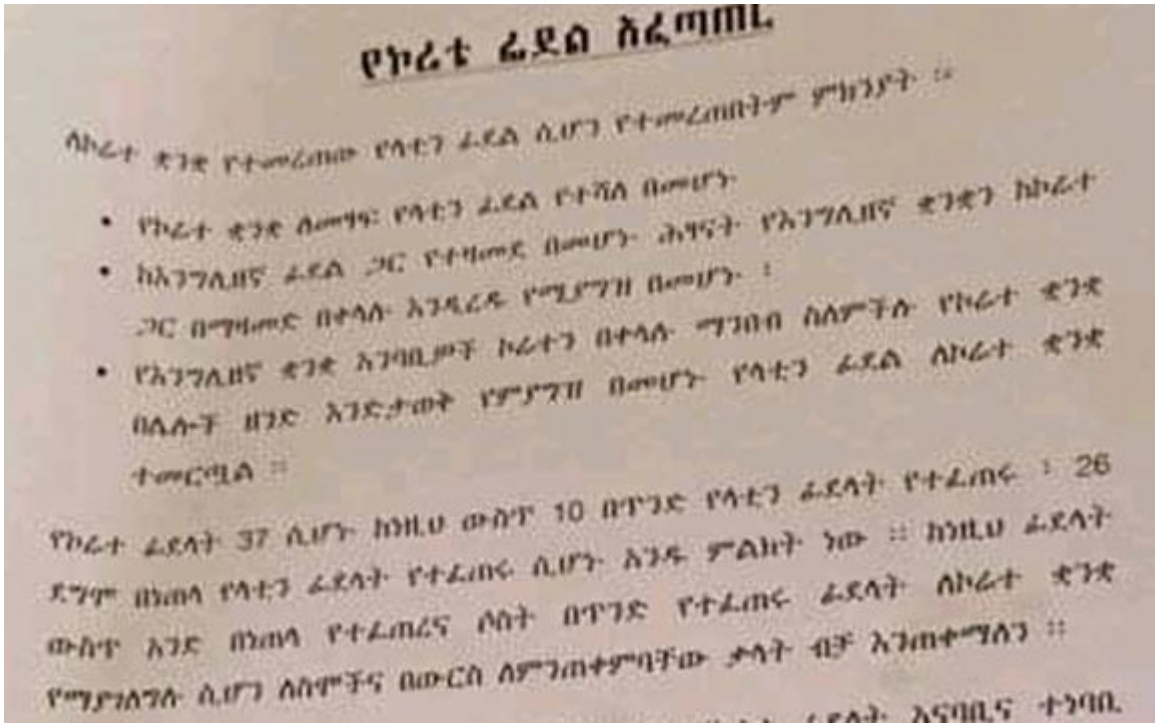



## B. Koorete Vowel Alphabets



## C. Koorete Latin Alphabets Production System

The shot below is taken from Amaro Wereda Culture, Tourism and Communication office. It is put here down as it is so as to keep its originality from the source.




| KOORETE DIIZO BEYTA TATO | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | U - Ha | Uu-Hu/Huu | Ui-Hi/Hii | Ua-Haa | Ue-He/Hee | U-Hi/H | Uo-Ho/Hoo |
| 2 | ለ - La | Lu/Luu | Li/Lii | Laa | Le/Lee | Li/L | Lo/Loo |
| 3 | ሙ - Ma | Mu/uu | Mi/Mii | Maa | Me/Mee | Mi/M | Mo/Moo |
| 4 | ሠ -Xa | Xu/Xuu | Xi/ii | Xaa | Xe/ee | Xi/ | Xo/oo |
| 5 | ረ - Ra | Ru/uu | Ri/ii | Raa | Re/ee | Ri/ | Ro/oo |
| 6 | ሰ - Sa | Su/uu | Si/ii | Saa | Se/ee | Si/ | So/oo |
| 7 | ሸ - SHa | SHu/uu | SHi/ii | SHaa | SHe/ee | SHi/ | SHo/oo |
| 8 | ቀ - Qa | Qu/uu | Qi/ii | Qaa | Qe/ee | Qi/ | Qo/oo |
| 9 | በ - Ba | Bu/uu | Bi/ii | Baa | Be/ee | Bi/ | Bo/oo |
| 10 | ቨ - Va | Vu/uu | Vi/ii | Vaa | Ve/ee | Vi/ | Vo/oo |
| 11 | ተ - Ta | Tu/uu | Ti/ii | Taa | Te/ee | Ti/ | To/oo |
| 12 | ቸ - CHa | CHu/uu | CHi/ii | CHaa | CHe/ee | CHi/ | CHo/oo |
| 13 | ነ - Na | Nu/uu | Ni/ii | Naa | Ne/ee | Ni/ | No/oo |
| 14 | ኘ - NYa | NYu/uu | NYi/ii | NYaa | NYe/ee | NYi/ | NYo/oo |
| 15 | ከ - Ka | Ku/uu | Ki/ii | Kaa | Ke/ee | Ki/ | Ko/oo |
| 16 | ወ - Wa | Wu/uu | Wi/ii | Waa | We/ee | Wi/ | Wo/oo |
| 17 | ዘ - Za | Zu/uu | Zi/ii | Zaa | Ze/ee | Zi/ | Zo/oo |
| 18 | ዠ - JHa | JHu/uu | JHi/ii | JHaa | JHe/ee | JHi/ | JHo/oo |
| 19 | የ - Ya | Yu/uu | Yi/ii | Yaa | Ye/ee | Yi/ | Yo/oo |
| 20 | ደ - Da | Du/uu | Di/ii | Daa | De/ee | Di/ | Do/oo |
| 21 | ጀ - Ja | Ju/uu | Ji/ii | Jaa | Je/ee | Ji/ | Jo/oo |
| 22 | ጰ - PHa | PHu/uu | PHi/ii | PHaa | PHe/ee | PHi/ | PHo/oo |
| 23 | ጠ - THa | THu/uu | THi/ii | THaa | THe/ee | THi/ | THo/oo |
| 24 | ገ - Ga | Gu/uu | Gi/ii | Gaa | Ge/ee | Gi/ | Go/oo |
| 25 | ጨ - Ca | Cu/uu | Ci/ii | Caa | Ce/ee | Ci/ | Co/oo |
| 26 | ጸ - XHa | XHu/uu | XHi/ii | XHaa | XHe/ee | XHi/ | XHo/oo |
| 27 | ፈ - Fa | Fu/uu | Fi/ii | Faa | Fe/ee | Fi/ | Fo/oo |
| 28 | ፐ - Pa | Pu/uu | Pi/ii | Paa | Pe/ee | Pi/ | Po/oo |
| 29 | ኣ - a | | | | | | |
| 30 | ኢ - i | | | | | | |
| 31 | ኤ - e | | | | | | |
| 32 | ኦ - o | | | | | | |
| 33 | ኡ - u | | | | | | |
| 34 | dha e.g dhaanxe | | | | | | |
| 35 | dza e.g bidzo | | | | | | |
| 36 | bha e.g bhikilo | | | | | | |
| 37 | Pooxhe (') | | | | | | |

## D. Built Models (Bi-LSTM, Word2Vec, N-gram) Code Segments

```
model = Sequential() ## build network topology
model.add(Embedding(len(word2index), output_dim = embedding_dim, input_length = MAX_LENGTH, name = 'embedding_layer'))
model.add(Bidirectional(LSTM(128, activation = 'relu', return_sequences = True, name = 'Bi-LSTM')))
## Dense layer
model.add(Dense(len(tag2index),activation = 'softmax'))
## model.add(TimeDistributed(Dense(1, activation='sigmoid')))
model.add(Dropout(0.3))
## Output layer
model.add(Dense(1, activation = "softmax"))
# compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
## Summarize the model
print(model.summary())
## fit the model
model.fit(train_sentences_X, to_categorical(train_tags_y, len(tag2index)), batch_size=128, epochs=10, validation_split=0.2)
## evaluate the model
loss, accuracy = model.evaluate(train_sentences_X, to_categorical(train_tags_y, len(tag2index)), verbose=1)
print('Accuracy: %f' % (accuracy*100))
```

```
import nltk
from gensim.models import Word2Vec
from sklearn.decomposition import PCA ## Principal Component Analysis
from matplotlib import pyplot
# define training data
data = open(r"C:/Users/TOSHIBA/Desktop/SMART/woga2.txt", encoding='utf-8')
unseen = data.read()
tokensss = nltk.word_tokenize(unseen)
# train model
input_data = tokensss
modelo = Word2Vec([input_data], size=200, sg=1, window=5, min_count=1, seed=100)
# fit a 2d PCA model to the vectors
X = modelo[modelo.wv.vocab]
pca = PCA(n_components=2)
result = pca.fit_transform(X)
# create a scatter plot of the projection
pyplot.scatter(result[:, 0], result[:, 1])
words = list(modelo.wv.vocab)
print(words)
for i, word in enumerate(words):
    pyplot.annotate(word, xy=(result[i, 0], result[i, 1]))
pyplot.show()
```

```
#---------default tagger-----------------------------------
dt = nltk.DefaultTagger(tag_fd.most_common()[0][0])
dtagged = dt.tag(tokens)
#---------Regular expression-------------------------------
KooreetePattern_1 = nltk.RegexpTagger( [
          (r'.*yaaca$','VBG'),
          (r'.*o$','VB'),
          (r'.*sso$','VBP'),
          (r'.*ese$','VBF'),
          (r'.*oko$','ADV'),
          (r'.*ita$','NPL'),
          (r'.*nxo$','NUMO'),
          (r'.*atse$','ADJ'),
          (r'.*wayte$','VBIP'),
          (r'.*iyo$','DFN'),
          (r'.*ko$','GCN'),
          (r'.*ka$','LCN'),
          (r'.*ra$','CCN'),
          (r'.*fa$','ACN'),
          (r'^-?[0-9]+(.[0-9]+)?$','CD')],
         backoff = dt)
rtagged = KooreetePattern_1.tag(tokens)
#---------Unigram tagger-----------------------------------
uni = nltk.UnigramTagger(train,backoff=KooreetePattern_1)
utagged = uni.tag(tokens)
#---------Bigram tagger------------------------------------
bi= nltk.BigramTagger(train,backoff=uni)
bitagged = bi.tag(tokens)
#---------Trigram tagger-----------------------------------
tri = nltk.TrigramTagger(train,backoff=bi)
tritagged = tri.tag(tokens)
```