Addis Ababa University

College of Natural Sciences

Amharic Morphological Segmentation Using Sequence-to-Sequence Modeling Approach

Terefe Feyisa Mamo

A thesis submitted to the Department of Computer Science in Partial Fulfillment for the Degree of Master of Science in Computer Science

Addis Ababa, Ethiopia

March 9, 2022

Addis Ababa University
College of Natural Sciences

Terefe Feyisa Mamo

Advisor: *Demeke Asres Ayele (PhD)*

This is to certify that the thesis prepared by Terefe Feyisa Mamo, titled: *Amharic Morphological Segmentation Using Sequence-to-Sequence Modeling Approach* and submitted in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science complies with the regulations of the university and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Demeke Asres (PhD),	Advisor:	_____
Yaregal Assabie (PhD),	Examiner:	_____
Ayalew Belay (PhD),	Examiner:	_____

Abstract

Morphological segmentation is an important task in the field of natural language processing (NLP). The process involves breaking words into their smallest meaning-bearing elements by detecting morphological boundaries. Amharic is an under-studied, under-resourced and morphologically complex language. The language has a much greater number of words than morphemes. As a result, most Amharic NLP applications, that use words as *a minimal unit*, face the challenge of handling rare and unknown words. To improve these issues, we examined Amharic morphological segmentation as a supervised neural *sequence-to-sequence (seq2seq)* and *sequence labeling* approaches using character embeddings. Moreover, we applied an algorithm, based on conditional random field, to construct a corpus size of 1,175,515 segmented words. Considering the bidirectional long short-term memory model, the seq2seq approach, *with F1-score of 97.65%* (state-of-the-art), outperformed the sequence-labeling method, *with F1-score of 94.82%*. Our results, we hope, give insight to the challenges of Amharic NLP.

KEYWORDS

Morphological Segmentation, Under-Resourced Languages, NLP, Amharic, Seq2Seq, Sequence Labeling

Acknowledgements

I would like to express my gratitude to my advisor, Dr. Demeke Ayele, who guided me throughout this project. Besides my advisor, I wish to show my appreciation to Dr. Ayalew Belay, for reviewing my first draft proposal and for his insightful comments.

Dr. Derib Ado offered me valuable data and links which I used in my project. My special thanks goes to Dr. Yemane Keleta Tedla, who sent me his full dissertation paper on Tigrinya morphological segmentation.

I wish to acknowledge the help provided by Dr. Solomon Guade and Mr. Yidnekachew Worku, who allowed me to use the Data Center of Information Network Security Agency (INSA). Thanks also to INSA for funding the school fee.

Last but not least, the assistance provided by my wife and friends is greatly appreciated.

Table of Contents

List of Figures

List of Tables

Acronyms

AAU	**A**ddis **A**baba **U**niversity
AMS	**A**mharic **M**orphological **S**egmentation
ANN	**A**rtificial **N**eural **N**etwork
BiGRU	**Bi**directional **G**ated **R**ecurrent **U**nint
BiLSTM	**Bi**direcional **L**ong **S**hort-**T**erm **M**emory
BMES	**B**egin, **M**iddle, **E**nd, **S**ingle
CACO	**C**ontemporary **A**mharic **Co**rpus
CRF	**C**onditional **R**andom **F**ields
CWS	**C**hinese **W**ord **S**egmenter
ES	**E**xperimental **S**cenario
FST	**F**inite **S**tate **T**ransducers
GRU	**G**ated **R**ecurrent **U**nits
HMM	**H**idden **M**arkov **M**odels
INSA	**I**nformation **N**etwork **S**ecurity **A**gency
LM	**L**anguage **M**odeling
LSTM	**L**ong **S**hort-**T**erm **M**emory
ML	**M**achine **L**earning
MS	**M**orphological **S**egmentation
MT	**M**achine **T**ranslation
NER	**N**amed **E**ntity **R**ecognition
NLP	**N**atural **L**anguage **P**rocessing
NLTK	**N**atural **L**anguage **T**ool**k**it
NN	**N**eural **N**etwork
POS	**P**art-**o**f-**S**peech
RQ	**R**esearch **Q**uestion
RNN	**R**ecurrent **N**eural **N**etwork
Seq2seq	**Sequence to Sequence**
Sp	**Sp**ecific Objectives
WS	**W**ord **S**egmentation

1 Introduction

Morphological segmentation (MS) is one of the basic tasks in the field of natural language processing (NLP) [1]. It aims at breaking words into meaning-bearing sub-components, known as morphs [1, 2, 3]. For example, using a notation by [4], the morphological segmentation of the Amharic lexical form /yəbetu/ "of the house" is given by:

$$/\text{yəbetu}/ \xrightarrow{[\text{MS}]} \{\text{yə}\}\beta\{bet\}\beta\{u\}$$

Where β denotes a boundary symbol that separates lexical entries indicating that "yəbetu" can be segmented into "yə", "bet" and "u". The segment "bet" corresponds to the word stem and "yə-" and "-u" are the prepositional prefix and the definite marker suffix respectively.

MS helps to identify morphs for proper syntactic analysis [5]. Also, it has been demonstrated [6] that training a model using sub-words as a minimal unit is effective for some NLP applications. In addition, it improves the task of handling rare and unknown words [6, 7] and decreases data sparsity significantly [8, 9]. In particular, it is useful to improve the performance of NLP tasks [5, 9] such as POS tagging [10, 11], machine translation [7], words' semantic similarity [6] and speech processing [2].

The effectiveness of a MS relies primarily on the efficiency of the underlying algorithms and approaches used [12]. It also depends on a clean, sizable and properly tagged corpora.

A MS can be modeled using hand-crafted dataset. However, building a sizable hand-crafted corpus is expensive in the amount of human work. It can also be designed automatically using statistical models such as Hidden Markov Model (HMM) and Conditional Random Fields (CRF) [13].

Recently, the traditional machine learning (ML) approaches like HMM and CRF are being replaced by deep learning methods to get a state-of-the-art performance level [14].

In this study, we implement neural network-based Amharic morphological segmentation (AMS). To the best of our knowledge, there has not been any work that attempted to examine neural network techniques for AMS.

Inspired by the work of [15] and the soft-attention encoder-decoder research method of [16], we apply a two-step process to implement an AMS. First, a CRF-based unsupervised method is applied to enrich a small, manually created, and morphologically segmented, Amharic corpus. Then, a supervised neural sequence-to-sequence (seq2seq) learning approach, using character embeddings, is investigated on AMS.

The main contributions of this work are:

i) to introduce an unsupervised technique to enrich an Amharic morphological segmentation corpus;

ii) to examine seq2seq and sequence labeling approaches for AMS;

iii) to make the corpus and the AMS model open-source.

This chapter is organized as follows: §1.1 introduces some background information about morphological segmentation. §1.2 justifies the motive behind doing this study. Then, a description of the central problem (see §1.3), the ultimate and specific objectives (see §1.4), the activities to be carried out (see §1.5), limitations beyond our control (see §1.6) and the possible application of results (see §1.7) shall be discussed. Finally, §1.8 outlines the overall organization of the rest of the thesis.

1.1 Background

Almost all practical applications in the field of natural language processing (NLP) must have a morphological component [4]. Morphological segmentation is a foundational and a preprocessing step in NLP [1, 17]. It aims to detect morphological boundaries of a given word so that word forms are segmented into meaning-bearing sub-words [1, 2]. Morphologically complex languages like Amharic have rich information such as number, gender, person and definiteness at morphological level. In order to unpack those information, an effective morphological segmenter is needed. Indeed, in their *Contemporary Amharic Corpus* preparation, [5] suggested that "Amharic orthographic words should be properly segmented in a preprocessing stage".

If a morphological segmenter is ineffective, the associated defects propagate to, and hence impact the accuracy of, downstream NLP applications such as machine translation [9], part of speech tagging (POS) [10] and morphological analyzer [18, 19]. Thus, it is important to perform an accurate Amharic Morphological Segmentation (AMS).

Nowadays, the traditional machine learning approaches like Hidden Markov Models (HMM) and Conditional Random Fields (CRF) are being replaced by a special type of deep learning method called Recurrent Neural Networks, specially, Long Short Term Memory (LSTM) neural networks.

In 2015, a Chinese word segmenter reached state-of-the-art performance using various LSTM networks [20]. In 2017, [21] presented an LSTM approach to Japanese word segmentation achieving a comparable accuracy to state-of-the-art systems. In 2018, the first LSTM based morphological segmentation research for Tigrinya has reported that the BiLSTM model outperformed when compared with the CRF and LSTM models [1].

Being morphologically rich, Amharic has a much greater number of words than morphemes. Morphemes are obtained by applying a morphological segmentation (MS) task on a given word.

1.2 Motivation

In 2016/17, during an NLP course, an assignment was given to develop an application to analyze Amharic Qinie **(ቅኔ)**. At the time, we [my colleagues and me] had no idea as to how to "teach" a computer for such task. So, we tried to mimic "Synsets" as in "WordNet" to store synonyms of words into a database and applied so many branching statements (*if-else if-else of C# Programming language*). But, to be frank, it was not a "right" way of teaching a computer.

In 2018/19 I joined the CoreNLP Project [to be done in collaboration between my organization INSA and Addis Ababa University]. The project is supposed to develop Amharic "core" NLP tools for morphological segmentation, POS tagging, grammar checking, spell checking, etc. for downstream applications using HMM and CRF. One thing was remarkable — morphological segmentation is the fundamental task for most of Amharic NLP tools and applications.

Another noticeable thing — a community of researchers and academicians, who work on the under-resourced Ethiopian languages, face challenges from scarcity of datasets and state-of-the-art tools and techniques. Such a community depends, mostly, on traditional techniques and tools that demand hand crafted datasets. So, it seems that it is essential to examine state-of-the-art techniques for Amharic morphological segmentation.

1.3 Statement of the Problem

An accurate morphological segmentation (MS) is a fundamental task in most NLP applications [1, 17, 4]. Amharic has a number of inflection possibilities and morpheme structures [24]. In order to unpack the rich information from Amharic words, an effective morphological segmenter is needed. A better performing MS tool, also, helps to avoid word level ambiguity and out-of-vocabulary challenges. Furthermore, if [in preprocessing stage] Amharic orthographic words are properly segmented, downstream applications would be effective [5].

Less-inflected languages, such as English, can use almost all possible word forms when developing NLP tools and applications. Also, these languages have rich language resources for research and development purposes.

On the contrary, under-resourced languages, like Amharic, lacks those digital resources. Consequently, most Amharic NLP tasks suffer from word level ambiguity and out-of-vocabulary challenges when they directly use all possible word forms.

Adopting NN approaches enables to gain a significant performance level for many NLP tasks including morphological segmentation [25]. It has been demonstrated [26] that neural architectures and representation learning discover useful features automatically from data and achieve competitive results.

MS tasks of Semitic languages such as Tigrinya [1] and Arabic [27] are already benefited using neural network approaches. Tigrinya is a very close neighbor of Amharic. And as such, one may expect a direct adoption of methods and methodologies from the work of [1].

However, the approach [1] followed (*window-based sequence labeling*) depends on feature extraction, unlike *sequence-to-sequence* method which does not require any. Specifically, it involves the use of a begin-middle-single-end (BMES) tagging scheme for labeling the morphs. Using this approach, several words may be tagged with identical pattern. For e.g., the pattern "BME-BME-BME-S-S-BE" may be used for the following segmented words, where the "-" indicates the morph-breaks and the bold-words are the stems:

- ስለቤታችንንምኮ/slv-**bet**-acn-n-m-ko/ "about our house"
- ስለሴታችንንምኮ/slv-**set**-acn-n-m-ko/ "about our women"
- ስለቁባችንንምኮ/slv-**qeb**-acn-n-m-ko/ "about our young hen"

Such issues can be alleviated by the seq2seq approach, where every word has a unique representation. To the best of our knowledge, there has not been any other research work [based on seq-2seq] that is devoted for a standalone Amharic morphological segmentation (AMS) system.

In this study, we aim at ascertaining how effective are sequence-to-sequence models in handling the task of Amharic morphological segmentation. This aim can be tailored to the following research questions (RQ):

RQ1 Are sequence labeling models effective on Amharic morphological segmentation task?

- This question is relevant as there has not been any Amharic morphological segmentation task based on sequence labeling models. Some works, such as [1] has claimed an encouraging performance accuracy for Tigrinya morphological segmentation using a window based BiLSTM (Bidirectional Long Short-Term Memory) model.

RQ2 Does a seq2seq model with an attention mechanism perform better than a sequence labeling model on Amharic morphological segmentation task?

- The basis of this question is the result of a systematic comparison [28] and review [29] about seq2seq attention-based models. These works assert that, seq2seq models have higher performance accuracy without using context features.

1.4 Objectives

General Objective

To develop Amharic morphological segmentation models, by augmenting an existing dataset, using sequence-to-sequence approaches and compare the models' performances.

Specific Objectives

- To review literature on deep learning methods related to morphological segmentation.
- To prepare a morphologically segmented Amharic dataset.
- To implement sequence labeling models.

- To compare the performances of the sequence labeling models using the metrics precision, recall and F1-score.
- To develop attention-based seq2seq model.
- To evaluate the attention-based seq2seq model based on the metrics precision, recall and F1-score.
- To compare the performances of the sequence labeling models with the attention-based seq2seq models.

1.5 Methods

Literature review. It is conducted to get insight about the:

- sequence labeling and seq2seq modeling techniques.
- morphological structure of Amharic.
- approaches to prepare morphological segmentation corpus.

Data preparation. It is done to prepare a morphologically segmented corpus.

Model Building. It is performed to design and implement sequence labeling and seq2seq modeling architectures for AMS.

Model Evaluation. Precision, recall and F1-score are used to measure the performances of the models.

1.6 Scope and Limitations

- The infix variations of a given root are considered as different atomic stems. For example, both **assəbabər** and **məsbər** have the same root **s.b.r**. However, each of them shall be taken as separate stems.
- Only the task of boundary detection is done, i.e., there is no concern about root-template-extraction. In other terms, Amharic words are assumed to have *prefix**-*stem*-*suffix** structure (where "*" stands for a repetition of one or more possibilities).
- Only surface segmentation is considered. Unlike canonical segmentation, surface segmentation involves breaking an input word into sub-words without any further string transformation [30].

1.7 Application of Results

AMS system is one of the basic resources to improve the construction of natural language tools for Amharic. It can be used to obtain a set of basic vocabulary units for different tasks in many downstream language technologies such as part of speech taggers, stemmers, machine translators, morphological analyzers, spelling and grammar checkers, speech and text understanding, statistical language modeling [31], and information retrieval. In text retrieval it is customary to preprocess texts by returning words to their base forms, especially for morphologically rich languages like Amharic [2].

Segmentation is critical for reducing the number of lexical units for Amharic [32]. Amharic has a very large unique surface forms. Using the surface forms as is leads to high OOV words in applications such as Machine Translators.

Scholars, NLP researchers and data scientists will be benefited by using the morphologically segmented Amharic corpus to undergo further researches in related fields such as NLP.

The morphological segmentation output enables to understand the properties and language processing challenges of Amharic. It also encourages further research. For example, future researchers can use AMS as a baseline. Moreover, they may incorporate the various modeling techniques of AMS in their own works.

1.8 Organization of the Thesis

A discussion regarding Amharic morphological segmentation and some background information about sequence to sequence NN techniques is presented in Chapter 2. Previous works related to our thesis is summarized in Chapter 3. Chapter 4 outlines an unsupervised technique to generate morphological segmentation dataset, by starting from a small seed corpus. Moreover, both sequence labeling and seq2seq modeling approaches are discussed in detail. Chapter 5 describes the experimental settings, scenarios and results. Finally, conclusion, recommendation and future work are presented in Chapter 6.

2 Literature Review

2.1 Introduction

This chapter provides an overview of the theoretical bases and the fundamental assumptions within which morphological segmentation has been framed.

2.2 Amharic Language and Morphology

Amharic is the second most widely spoken Semitic language after Arabic. It is the largest official language of Ethiopia having its own script (Ge'ez). It is an under-resourced language [5], despite having tens of millions of native speakers. It lacks digital resources, such as segmentation dataset to develop NLP tools and to undertake researches. As a result, the dominant approaches to develop systems are mostly rule-based, such as memory-based learning [18].

Amharic has a complex inflectional morphology, particularly for verbs, employing not only prefixes and suffixes but also modifications of the typical Semitic consonantal root-and-pattern type [24]. Nouns and adjectives are inflected for case, gender, number and definiteness [33].

A single verb may consist time (past, present, and future), gender (male and female), action (command, statement, invitation) and negation (not) which is expressed in a sentence in other languages. Borrowing an example from [19], the Amharic word **"አልደወለችላቸውም"** ≡ "aldeweleclacewum" refers to the English sentence "She did not call them". The Affixes **አል/**'al-, **-ች/**c-, **-ላቸዉ/**lacewu-, **ም/**m add extra meaning of various kinds to the main morpheme **"ደወል"**/"dewel".

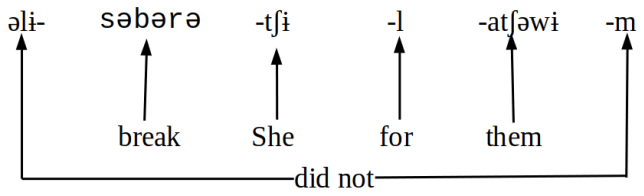

FIGURE 2.1: Morphemes of an Amharic word.

2.2.1 Amharic Morphology

As Amharic is a Semitic language, a major part of word formation is ruled by non-concatenate morphology. A consonantal root which encodes the semantics fuses with a template, i.e. a prototypically vocalization and gemination pattern, into a word base [34].

There are a number of semi-productive templates which transform a consonantal root into a nominal. For example, the template C1äC2aC3i forms agentive nouns, like säbari 'one who breaks' (root: *sbr* 'BREAK'), gäday 'killer' (root: *gdl* 'KILL').

A number of affixes change a nominal stem of a certain category into another nominal category.
For example, by suffixing:

-iya to the verbal noun to form instrumental or locational nouns: mäṭräg-iya 'broom (tool for sweeping)', maräf-iya 'place for sleeping';

-äññña to the abstract noun to form agent noun: qäld-äññña 'joker' (root: qäld 'joke'), gazeṭäññña 'journalist' (root: gazeṭa 'newspaper');

-ññña to the name of the ethnic group to form language names: amarI-ññña (root: amara 'Amhara'), oromI-ññña 'Oromo (language)' (root: oromo 'Oromo (people)');

-nnät to common nouns to form abstract nouns: säw nnät 'body' (root: säw 'man, human');

Common nouns form their plural by the suffix -očč: bet → betočč 'house(s)', färäs → färäsočč 'horse(s)', etc. Adjectives, in contrast, can form their plural by partial reduplication of a consonant with or without the insertion of the vowel a: addis → adaddis 'new' or təlləq → tələlləq 'big'. Instead of using -očč, some nominals may form an optional plural by ablaut, or by the suffixes -an or -at: kokäb → käwakəbt 'star(s)', qän → qän-at 'day(s)', kəbur → kəburan 'honored person(s)'.

The gender distinction is expressed by agreement marking on the verb, on demonstratives or on the definite article. For example: dəmät-wa cat-DEF:f 'the (female) cat'; dəmät-u cat-DEF(:m) 'the (male) cat'.

The accusative is expressed by the suffix -n. For example: "ambässa-w-n gäddäl-ä" 'He killed a lion / the lion'.

Consonantal roots are lexicalized into base stems by conjugational templates, i.e. a specific vocalization and gemination pattern. Generally, four conjugational templates are of importance: the perfective, the imperfective, the jussive/imperative and the converb [35]. Perfective verbs and converbs use a suffix set for subject agreement, but imperfective and jussive verbs a combination of prefixes and suffixes.

The negative marker is *al-* for perfective verbs but *a-* for imperfective and jussive verbs: *al-hed-ä-mm* 'he did not go', *a-y-hed-əmm* 'he does not go'. The suffix *-mm* is an obligatory part of negative perfective and imperfective verbs in main clauses but it does not occur in subordination and with jussive verbs: *k-al-hed-ä* 'if he does not go', *a-y-hid!* 'He should not go!'.

Relational prefixes establish a grammatical relationship that links their complement to another word or phrase in a clause (see Table 2.1).

Table 2.1: Relational prefixes.

Prefixes	Semantics with nominals	(relative) verbs
yä-	of (GEN)	which, who (REL)
l(ä)-	for (i. e. addressee, beneficiary)	in order to
kä-	from, with (comitative)	if (real condition)
b(ä)-	in, at, on, with, through, by, against	if (condition), when (temporal)
ənd(ä)-	like	that (complementizer), just as
səlä-	because of	because
əsk(ä)-	until, to	until
bästä-	towards (restricted use)	-
s-	-	while, when
wädä(-)	towards	-
(y)alä(-)	without	-

Concerning morpho-phonological process, all alveolar consonants, except *r*, can be palatalized when they are followed by the vocalic suffixes *-i* or *-e* which function as second-person-singular gender marker or as first person singular subject marker in converbs, respectively. Table 2.2 depicts the palatalization process from third-person-singular-masculine-perfective (3sm PV) verb to second-person-singular-feminine-imperfective (2sf IMP) verb.

Table 2.2: Palatalization.

3SM PV		2SF IMP
säbbärä	'break'	səbär-i
wäddädä	'love'	wwdäğ(i)
fälläṭä	'split'	fələč̣ (i)
ammänä	'believe'	əmäñ(i)
gäddälä	'kill'	gədey

Due to its morphological complexity, Amharic can not directly adapt the rich techniques, algorithms and tools from well enriched languages. Unlike English, for example, a root-and-pattern morphology is used to form words. To demonstrate what a root-and-pattern is, consider the root s.b.r, which is the basis of many words having to do with the notion of "breaking" in Amharic. Combining this root with the pattern aC1:əC2aC2əC3, for example, forms a manner noun, *assəbabər* "way of breaking" (the C's indicate the consonant slots, and ":" indicates gemination). The same template (see Table 2.3) may be used to generate different words from different roots.

Table 2.3: Different roots are combined with the same vowel patterns to form "manner nouns" and "infinitives".

Root		**Having to do with the notion of:**	
1) g.d.l		**killing**	
2) s.b.r		**breaking**	
Root	**Template**	**Word**	**Meaning**
s.b.r	məC1C2əC3	məsbər	Infinitive : "to break"
g.d.l	məC1C2əC3	məgdəl	Infinitive : "to kill"
s.b.r	aC1:əC2aC2əC3	assəbabər	Manner noun : "way of breaking"
g.d.l	aC1:əC2aC2əC3	aggədadəl	Manner noun : "way of killing"

Once the roots are identified, words are composed by adding vowels, prefixes and suffixes to the root consonants. As shown in FIGURE 2.2, "Counting the verb stem as two morphemes (root and pattern), an Amharic verb may consist of as many as ten morphemes" [32].

As per [35], verbal roots in Amharic consist of only three radicals of which only any two could be identical. Such roots serve as input for the derivation of various types of verbal and nominal stems. The process involves reduction and extension of one or two of the three radicals.

Example 2.2.1. Showcasing a verb may consist many morphemes.
“ለማያነቡልንም” /ləmmayanəbbullɨnɨm/ “also for they who do not read to us”
lə-mm-a-y-{n.b.b+aC1əC2C3}-u-ll-n-m. Where ”n.b.b” is the root and ”aC1əC2C3” is the pattern.

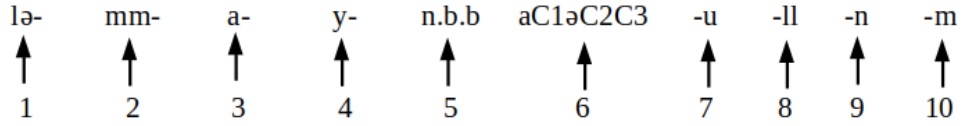

FIGURE 2.2: A verb may consist of up to ten morphemes.

In this thesis, the infix variations of a given root are considered as different atomic stems. For example, both *assəbabər* and *məsbər* have the same root *s.b.r.* However, each of them shall be taken as separate stems without considering the *infix “ba”*. Also, there is no concern about *root-template-extraction* as our focus is only the task of *boundary detection*.

2.2.2 Amharic Orthography

The Amharic script is known as Ge’ez or Ethiopic. The writing system is called Fidäl **(ፊደል)** in Ethiopian Semitic languages. Fidäl means “script”, “alphabet”, “letter”, or “character” or abugida **(አቡጊዳ)**, from the first four symbols. The system allows representing a consonant (C) and a vowel (V) together as a symbol. Each symbol represents a CV syllable. However, a syllable in Amharic may have a CVC structure. Unlike majority of its Semitic scripts, such as Arabic and Hebrew, the Amharic script is written from left to right.

An abugida, from Ge’ez: **(አቡጊዳ)**, sometimes known as alphasyllabary, neosyllabary or pseudo-alphabet, is a segmental writing system in which consonant-vowel sequences are written as units; each unit is based on a consonant letter, and vowel notation is secondary. Formally, an abugida is defined as “a type of writing system whose basic characters denote consonants followed by a particular vowel, and in which diacritics denote other vowels” [36]. Amharic has thirty three basic consonants and seven diacritics representing implicit vowel characters. The seven forms are based on the seven vowels **(ኧ (ä) ኡ (u) ኢ (i) ኣ (a) ኤ (e) እ (ə) ኦ (o))**. The 33 basic characters are **ሀ, ለ, ሐ, መ, ሠ, ረ, ሰ, ሸ, ቀ, በ, ተ, ቸ, ኀ, ነ, ኘ, አ, ከ, ኸ, ወ, ዐ, ዘ, ዠ, የ, ደ, ጀ, ገ, ጠ, ጨ, ጰ, ጸ, ፀ, ፈ, ፐ**. Figure Annex .1 gives the typical Amharic alphabet. Figure Annex .2 depicts the transliteration table that is used in the study.

Each of the 33 consonants have seven “orders” or shapes depending on the vowel with which a given consonant is combined. These are arranged into seven

houses (orders) according to the kind of each vowel that the consonants associate themselves with, i.e., the consonant-vowel (CV) combinations [5]. Consider for example the following individual symbols:

ሀ (bä) ሁ (bu) ሂ (bi) ሃ (ba) ሄ (be) ህ (bə) ሆ (bo)

When a vowel come in contact with a consonant on its left side, the vowel will no longer be considered independently but together with the immediately preceding consonant forming a new unique symbol. As an example of this phenomenon, consider the declension of a noun for number by adding -ኦች(oc) at the end of the noun [37]. As can be seen in the example below the combination of ት and ኦ results in ቶ [38].

ቤት(betə) + -ኦች(ocə) = ቤትኦች(betocə)

Written as: ቤቶች (betocə)

ሰው(säwə) + -ኦች(ocə) = ሰዉኦች(säwuocə)

Written as: ሰዎች (säwocə)

In Amharic, the end of a sentence is marked by a pair of colon (። አራት ነጥብ, ʾäratə nät'əbə). Whereas exclamatory and interrogative sentence are ended by '!' and '?' punctuation marks respectively. The individual words in a sentence are separated by colon : ሁለት ነጥብ, hulätə nät'əbə) (or sometimes with white space character [37]. In Amharic there are 9 homo-phonic characters such as (ሀ, ኀ, ሐ, and ኸ), (ሰ and ሠ), (ጸ and ፀ), (ው and ዉ) and (አ and ዓ), which have the same pronunciation and meaning. Sometimes such characters are used interchangeably in a language and this also makes it challenging.

An orthographic word may function as a phrase (e.g., ከቤትዋ/kəbetwa/ "from her house"), a clause (e.g., የመጣው/jəmmətʻtʻaw/ "the one who came"), or even a sentence (e.g., አልበላችም /ʔalbəllaʧʧim/ "She did not eat.") [5]. In order to expose the hidden vowels, we apply a transliteration before tagging.

2.2.3 Stems versus Roots

For applications such as machine translation, the use of a stem as a morpheme is preferable than the use of a root [15]. Let's take, as an example, the task of an Amharic to English statistical machine translation system. Such a system needs word segmentation for a couple of reasons.

Firstly, creating a parallel corpora demands an accurate word-to-word alignment between the source and the target languages [15]. This is not always doable as there are many situations where the same information can be expressed

using one word in Amharic but many words, or even a sentence in English. For example, the Amharic word: "አልሰበረችላቸውም።" is roughly equivalent with the English sentence: *"She did not break for them."*.

Second, the approach usually demands the storage of a pair of surface words into the lexicon. This, however, is not an easy task when processing morphologically rich languages like Amharic. These languages have high vocabulary growth rate which results in high perplexity and a large number of out-of-vocabulary words [39].

Thus, when developing a translation system, one approach is to break the surface words (e.g., "አልሰበረችላቸውም") into morphemes (e.g., "አል-ሰበረ-ች-ላቸው-ም" /əlɨ-səbərə-tʃɨ-l-aʃəwɨ-m/), and then remove the affixes (e.g., əlɨ-, -tʃɨ, -l, -atʃəwɨ and -m) to store representative stems (e.g., "səbərə"). This way, one may get a mapping as depicted in Figure 2.3 and avoid an out-of-vocabulary words (OOV) problem.

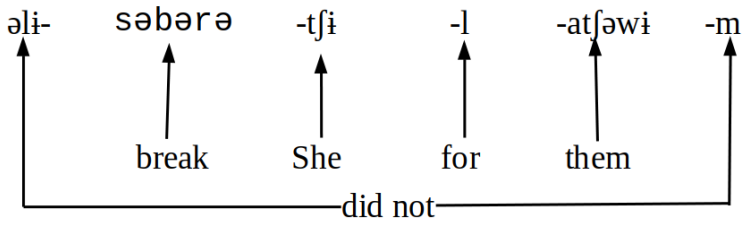

FIGURE 2.3: A mapping between an Amharic word and its equivalent English sentence.

2.2.4 Challenges in Amharic NLP

As Amharic has a complex "root-and-pattern" morphology, the semantic of a word is determined by both the root and the pattern. This complicates the process of interpretation. For languages like English, NLP techniques can benefit from stemming. But for Amharic, finding the root, or stem, of a word is challenging to automate. Moreover, the root of a word often has a very abstract meaning, which is not at an appropriate level for NLP.

The progress of Amharic NLP is also hindered by some aspects like:

- The absence of capitalization, which makes it hard to identify proper nouns, titles, acronyms, and abbreviations.
- the highly inflectional and derivational nature of the language, which makes morphological analysis a very complex task.

2.3 Morphological Segmentation

2.3.1 Boundary Detection

Morphological Segmentation (a.k.a "boundary detection") is the process of breaking down words into sub-components, known as morphemes, which are typically taken to be the meaning-bearing components in languages [1, 2, 3].

For example, using the notation from [4], the Amharic lexical form **/yəbetu/** "of the house", can be segmented as: {**yə**}β{**bet**}β{**u**}, where β denotes a boundary symbol that separates lexical entries indicating that **"yəbetu"** can be segmented into **"yə"**, **"bet"** and **"u"**. The segment **bet** corresponds to the word stem and **yə-** and **-u** are the prepositional prefix and the definite marker suffix respectively.

Thus, in scope, this work does not consider a fully fledged morphological analysis: which includes canonical segmentation (where morph transformation is considered), POS tagging, etc. for given words.

2.3.2 Challenges in Segmentation

There are some inherent challenges associated with an Amharic morphological segmentation. An example challenge is a word-level ambiguity. This is a phenomenon that mostly happen when some character sequences of morphemes innately appear as part of a word. In such a scenario, the characters do not represent grammatical features and accordingly, a better segmentation model should be applied for the disambiguation. Table 2.4 showcases how boundary detection is an ambiguous task by focusing just on two prefixes: **yə** "of" and **kə** "from".

Table 2.4: Ambiguous morph boundaries.

	Word	Segmentation	Legal?
4.1	/kəbet/ "from house"	{kə}β{*bet*}	Yes
4.2	/yəbet/ "of house"	{yə}β{*bet*}	Yes
4.3	/kəyəbet/ "from each house"	{kə}β{yə}β{*bet*}	No
4.4	/yəyəbet/ "every house's"	{yə}β{yə}β{*bet*}	No
4.5	/yəkətəma/ "of city"	{yə}β{kətəma}	Yes
4.6	/yəkətəma/ "of city"	{yə}β{kə}β{təma}	No
4.7	/kəyət/ "from where"	{kə}β{yət}	Yes
4.8	/kəyət/ "from where"	{kə}β{yə}β{*t*}	No

In Table 2.4, if the segmentation is acceptably correct, the legal column becomes "Yes". Just because (**4.1**) and (**4.2**) are legal, it doesn't necessarily hold for (**4.3**) and (**4.4**), as **kəyə** and **yəyə** are single morphemes. (**4.6**) and (**4.8**) are illegal as **kə** and **yə** are innate part of the words **kətəma** and **yət** respectively.

Out-of-vocabulary words (OOV), is another challenge. This problem occurs when one attempts to use the surface forms of words when developing Amharic NLP systems. Unlike less-inflected languages, such as English, one cannot directly store possible word forms in an Amharic lexicon. Instead, as Amharic has a number of inflection possibilities and morpheme structures [24], words are segmented into morphemes to get a stem word. Thus, AMS, can be used to unpack information from Amharic words. Brief, when facing MS challenges, great effort is needed in choosing effective and efficient boundary detection models [40].

2.4 Segmentation Approaches

2.4.1 Rule Based Approaches

Amharic morphological segmentation are mostly found as an embedded part of other rule-based downstream applications such as *Development of Amharic Morphological Analyzer Using Memory Based Learning* [18], *Design and Development of Amharic Grammar Checker* [19] and *Amharic Anaphora Resolution Using Knowledge Poor Approach* [41].

"HornMorpho" [23] is a well known morphological analyzer that is based on finite state transducers (FSTs). It is a fully-fledged morphological analysis tool for Amharic, Tigrinya and Oromo languages.

2.4.2 Neural Network Approaches

Using neural network method, there are mainly two types of sequence segmentation approaches: sequence labeling (see §2.7) and sequence to sequence (seq2seq) (see §2.8.) Recurrent neural networks (see §2.5.1) are effective for both sequence labeling [42, 43, 44] and sequence-to-sequence [28, 45, 46] approaches for segmentation tasks.

Some authors use those approaches (terms) interchangeably. For example, [1] worked on Tigrinya sequence labeling task, but use the term "sequence to sequence" freely in their article. Others, such as [6], use the term seq2seq exclusively for approaches that involve an encoder-decoder frameworks.

LSTM and GRU units, along their bidirectional variants, are widely used RNN networks [47]. We use these units with their variants for both approaches. Concerning performance issues on Amharic morphological segmentation, there is no clear answer as to which of them is better [43].

2.5 Artificial Neural Networks

Artificial Neural Networks (ANN) is a collection of interconnected processing elements known as neurons. Neurons mimic the electrical connectivity in the human brain to produce intelligence [48]. As neuron is to human brain, perceptron is to ANN. Figure 2.4 depicts a perceptron, having four main parts [49]:

1. Input values: are features in numerical format.
2. Weights and bias: The weight is a real number associated with each input node. It indicates the strength of the node. The bias is a constant real number.
3. Weighted sum: $z = \sum_{i=1}^{n}(w_i \times x_i + b) = \boldsymbol{W^T x} + \boldsymbol{b}$
4. Activation function f: defines how the weighted sum of the input is transformed into an output. $y = f(\boldsymbol{W^T x} + \boldsymbol{b})$

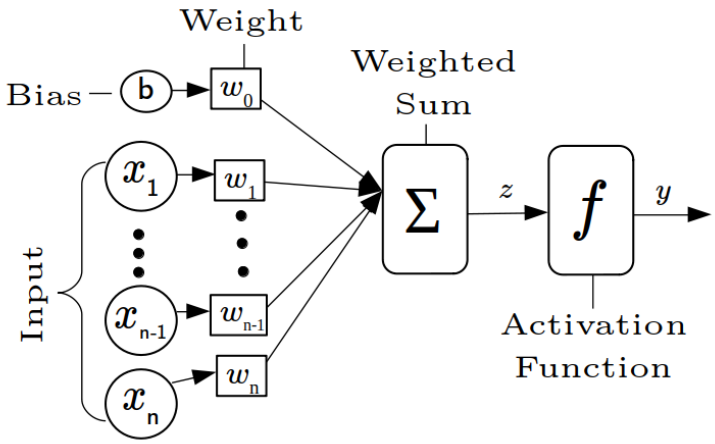

FIGURE 2.4: The perceptron.

2.5.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) is a kind of ANN that uses previous states for making new predictions. Text processing tasks, such as morph classification, are sequence-based. That means, they require knowing what morph came before the one that is being classified at the moment [50]. Regular neural networks

do not consider the input-output dependencies [51]. As such, they do not help to get a better semantic insight from a text data. Processing text data that demand the sequential dependency of words can be done using RNN [52]. As shown in Figure 2.5, RNNs can process sequential data using current timestamp (t) and previous timestamp ($t-1$).

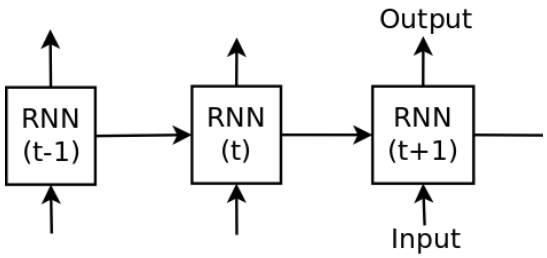

FIGURE 2.5: RNNs capture temporal nature of an input [51].

One characteristic of an RNN is that, the data flows through it in cycles (Figure 2.6). This property makes RNN to be able to memorize inputs. For this reason, they are suitable for sequence labeling and prediction tasks. Machine translation applications such as Google Translate implements RNN for better performance [53].

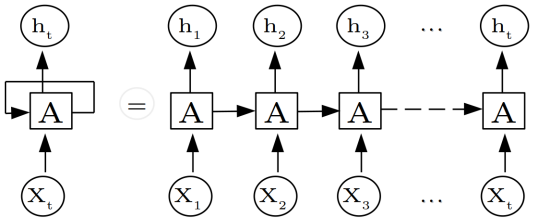

FIGURE 2.6: RNNs do similar calculations for each element in a sequence [51]. X_t, h_t and A denote, respectively, the input, the hidden state and the activation function at time-step t.

Though RNNs are good for sequence processing, they fail to recognize long-term dependencies as they are susceptible to the *vanishing* and *exploding* gradient problems [45]. Such problems arise when the gradient either decays or grows exponentially [54].

Long short term memories (LSTMs) are special type of RNN that overcome the limitations within the existing RNN architecture [51].

This section discusses about the LSTM neural networks and then generalizes the concept to GRU and their bidirectional versions (BiLSTM and BiGRU).

2.5.2 The LSTM Networks

The Long Short Term Memory network architecture consists of a set of recurrently connected memory blocks, known as LSTM memory cells (dotted boxes in Figure 2.7). A block consists of three gates. These gates are used to manage the block's state and output. The gates decide situationally what [55]:

- information to discard from the block (**Forget Gate**).
- values from the input to renew the memory state (**Input Gate**).
- to output based on input and the memory of the block (**Output Gate**).

LSTM are better at finding and exploiting long range dependencies in a data. It has an input layer **X**, hidden layer **h** and output layer **y**. If we input $x = [b, e, t, u]$, for example, into the LSTM, we expect a prediction of a tag set $y = [B, M, E, S]$.

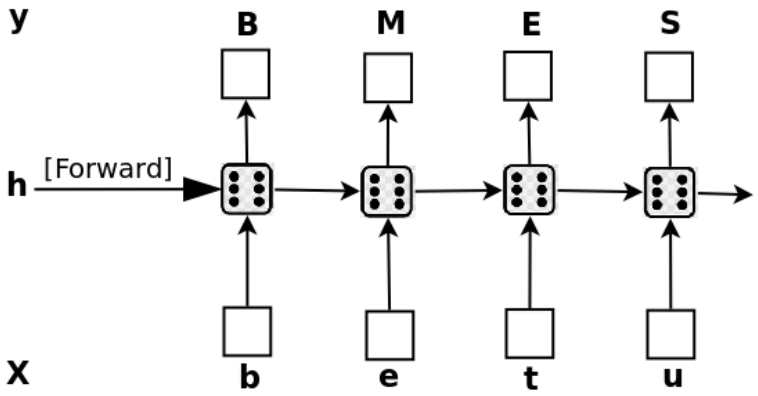

FIGURE 2.7: An LSTM network.

LSTM units

For every memory cell, there is a recurrent unit to overcome the vanishing and exploding gradient problems. Figure 2.8 depicts one variant of an LSTM unit.

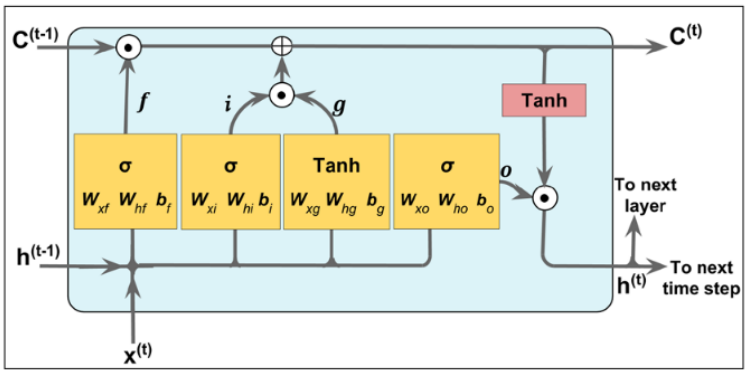

FIGURE 2.8: An LSTM unit.

The equations for a forward pass of an LSTM unit are:

$$
\begin{aligned}
\boldsymbol{f}_t &= \sigma\left(\boldsymbol{W}_{xf}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hf}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_f\right) \\
\boldsymbol{i}_t &= \sigma\left(\boldsymbol{W}_{xi}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hi}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_i\right) \\
\boldsymbol{g}_t &= tanh\left(\boldsymbol{W}_{xg}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{hg}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_g\right) \\
\boldsymbol{o}_t &= \sigma\left(\boldsymbol{W}_{xo}\boldsymbol{x}^{(t)} + \boldsymbol{W}_{ho}\boldsymbol{h}^{(t-1)} + \boldsymbol{b}_o\right) \\
\boldsymbol{C}^{(t)} &= \left(\boldsymbol{C}^{(t-1)} \odot \boldsymbol{f}_t\right) \otimes (\boldsymbol{i}_t \odot \boldsymbol{g}_t) \\
\boldsymbol{h}^{(t)} &= \boldsymbol{o}_t \odot tanh\left(\boldsymbol{C}^{(t)}\right)
\end{aligned}
$$

GRU versus LSTM

The GRU is like an LSTM with a forget gate, but having fewer parameters than LSTM (it lacks an output gate). Concerning performance issues, there is no clear answer as to which (GRU versus LSTM) is better [43].

2.5.3 The Bidirectional Networks

Bidirectional LSTM or GRU (see Figure 2.9), are two hidden LSTM or GRU layers. In sequence tagging task, they enable us to have access to both past and future input features.

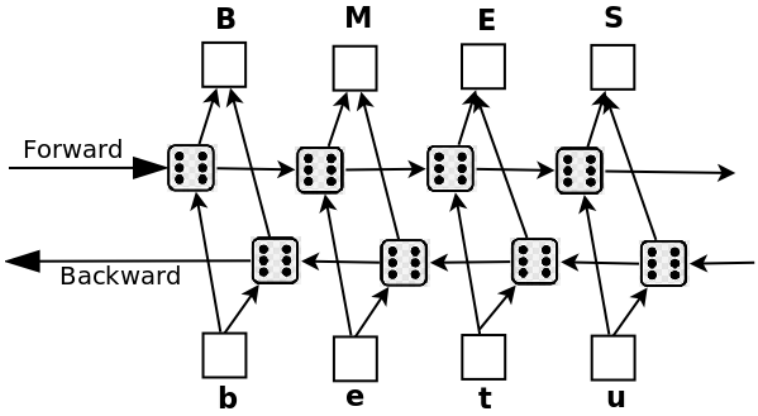

FIGURE 2.9: A BiLSTM or BiGRU network.

2.5.4 Training a Neural Network

Training a neural network involves finding the optimum weights by the commonly used *back-propagation* algorithm [48]. The algorithm usually starts by assigning small random values for the network weights and then iteratively proceeds to find ideal weights. After adjusting every initial random weights, it continues to the next epoch to minimize a cost function or until a maximum epoch point is reached as in Figure 2.10.

The performance of the final model is affected by certain parameters like number of neurons, hidden layers, the architecture, the training settings, and the data splitting scheme [48]. In order to find the "right" combination of parameters, a well defined experimental procedure is needed. The commonly used training procedure includes three phases: data pre-processing, training, and testing as shown in Figure 2.10.

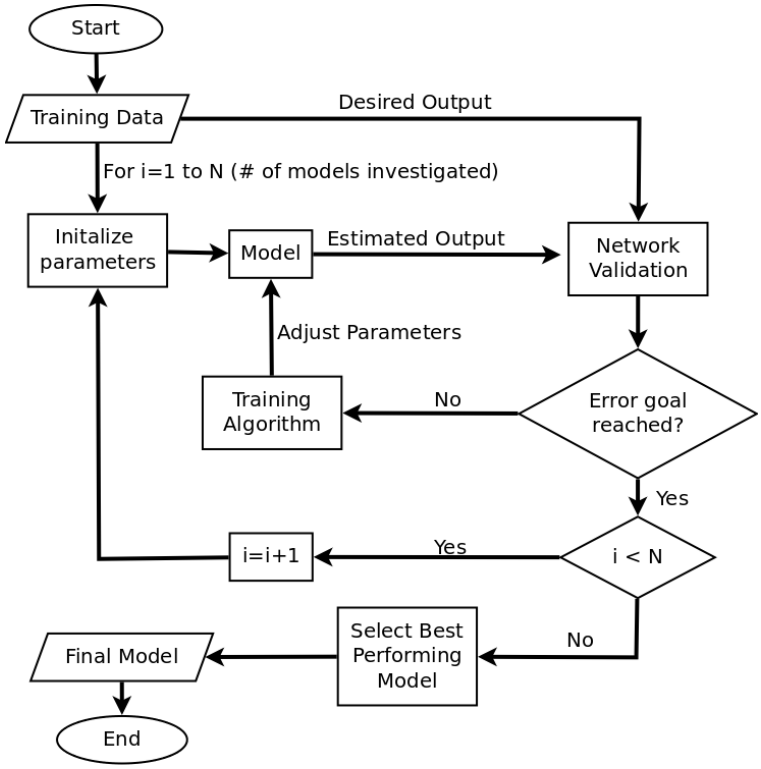

FIGURE 2.10: Workflow of a neural network training [48].

2.5.5 The Back-propagation

A supervised learning method used to determine the weights of an ANN, where the difference between the desired and the model's output is minimized.

2.5.6 The Cross-validation

A re-sampling technique to evaluate the performance of a classifier in a limited data situation [56]. K-fold cross-validation is the basic one. This method consists of partitioning a shuffled dataset into k disjoint subsets (folds). The most commonly used is tenfold cross-validation, when k = 10. In most cases, the training and validation sets must cross over in successive rounds. When the k^{th} fold is used to learn or train a model, the remaining k−1 folds are used to validate the model [57]. Figure 2.11 showcases an example with k = 3.

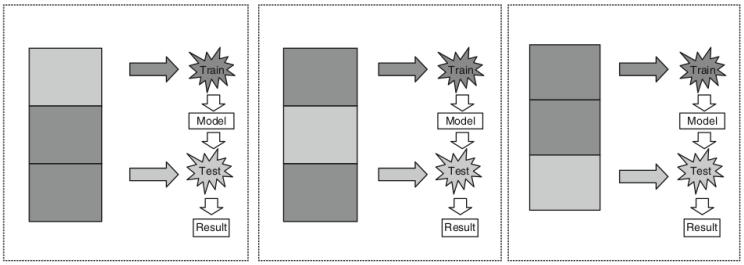

FIGURE 2.11: 3-fold cross-validation procedure. The darker part of the data is used for training. The lighter sections, for validation (figure from [57]).

2.5.7 Inference Layer

Inference refers to the process of converting real-valued scores to a normalized probability distribution. It is usually used as the final layer of a neural network for mutually exclusive multi-class classifications. One of the commonly used inference layer is Softmax. It is a mathematical function that is given by [58]:

$$\sigma(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

Where:	
$\vec{x}$	The input vector.
x_i	Elements of the input vector.
e^{x_j}	The standard exponential function.
$\sum_{j=1}^{N} e^{x_j}$	The normalization term.
N	The number of classes in the multi-class classifier.

2.6 Language Modeling

Language modeling (LM) is the use of different algorithms to find the probabilistic relationships between a given sequence of words. LM can be used to develop a model that generalizes a given corpus of text. For example, LSTM models can be configured to learn the general structural properties of a given corpus, and can generate new sequences that are representative of the original corpus [59].

A language model may work at the word level using individual word forms as atomic units. It can also work at character level, learning from sequences of characters.

2.6.1 Character-level Models

Word level language models handle only generic OOV (using a generic "unknown" vector). Such models are usually less effective than those who use sub-word units. This is because, the number of possible sub-words is a lot smaller than the number of possible words. As such, sub-word unit models usually have fewer number of parameters compared to the word-level models. What is more, the vocabulary in sub-word unit models is finite, which makes it possible to represent any word in the language. For example, English has only 26 letters but more than 171,000 words in a day to day usage [60].

According to [61], character-level models can learn orthographic similarity of affixes and hence, are effective for morphologically rich languages. Also, [62] demonstrated the effectiveness of character-level model to develop a character embedding. Further more, [21] has shown that character embeddings are effective to identify prefixes and suffixes. *An embedding* is a fixed-size real-valued vector to encode and represent an entity such as a character and a word [63].

Figure 2.12 depicts an instance of the architecture of a model generating an embeddings for the word "cats". Given a word $w = c_1, c_2, \ldots c_{|w|}$, where $c_i \in \boldsymbol{C} = \{\text{the set of all characters in the language (or the vocabulary)}\}$, it first represents each character c_i by a one-hot encoding. At each time step $\boldsymbol{i}$, it feeds $\boldsymbol{c_i}$ into a BiLSTM to map it to its continuous representation (*character embedding*), $\boldsymbol{c_i}$.

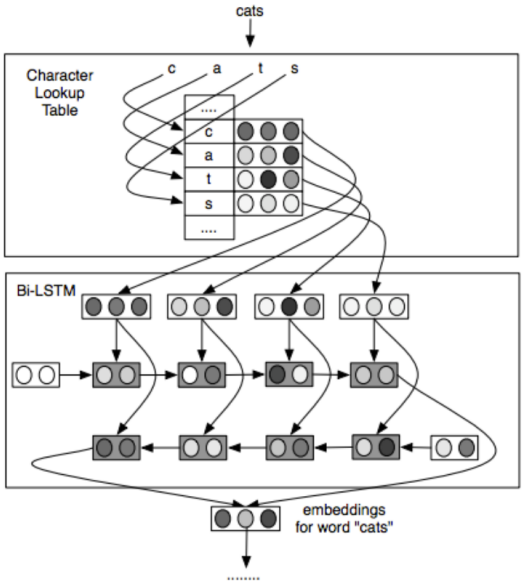

FIGURE 2.12: Character-level BILSTM model [62].

2.6.2 Tagging Scheme

As discussed in, word segmentation can be taken as an organized classification task with encoded classes. An encoding is used to identify the presence of morph boundaries around a target character. An experiment for Japanese word segmentation [21] has reported that the "BIES" scheme contributed for better performance accuracy. Also, as demonstrated by [64], models using fine-grained tagging schemes contribute significantly for performance accuracy. For this reason, we adopt the fine-grained "BMES" encoding scheme by [65] (just a matter of using different letter: the 'I (for Inside)' in the "BIES" corresponds to the 'M' in the "BMES"). This encoding scheme uses four class set {B, M, E, S} to capture information about the sequence of morphs in a given word. Table 2.5 describes the purpose of each label/letter.

Table 2.5: Labeling symbols and their meanings.

Label	Indicates
Begin	The start of a morph.
Middle	The continuity of a morph.
End	The end of a morph.
Single	Single morphs.

Table 2.6 depicts an instance of the "BMES" tag scheme using an example of three words ("bet", "betu", and "betunmko") with their corresponding manual segmentation and tag-set.

Table 2.6: An instance usage of the "BMES" tag scheme.

No.	Input	Segmentation	Labeling
1	[b, e, t]	bet	[B, M, E]
2	[b, e, t, u]	bet-u	[B, M, E, S]
3	[b, e, t, u, n, m, k, o]	bet-u-n-m-ko	[B, M, E, S, S, S, B, E]

2.6.3 One-Hot Encoding

Categorical nominal variables are a finite set of discrete values with no relationship between them. For example, the "BMES" encoding scheme uses four label nominal values {B, M, E, S}. A NN require numerical input and output

variables. For this reason, a *one hot encoding* is used to convert categorical nominal data to integer data.

A one-hot encoding uses a group of bits. A bit represents a single category. Only one bit is "on" at a time [66]. Hence, a categorical variable with n possible categories is encoded as a feature vector of length n. Table 2.7 shows an example for the "BMES" encoding variable. In the table, there are four categories {B, M, E, S} and thus four binary variables are needed. A bit, "$b_i = 1$" for a category and "0" for the rest of the categories.

TABLE 2.7: One-hot encoding of a category of four labels.

	b_1	b_2	b_3	b_4
B	1	0	0	0
M	0	1	0	0
E	0	0	1	0
S	0	0	0	1

2.6.4 Window Based Classification

In sequence classification tasks, an information that co-occurs with a piece of information is required [50]. Without such accompanying information, a word-level ambiguity may arise. This phenomenon results when some character sequences of morphemes innately appear as part of a word. For example, let's consider the Amharic words: "kətəma" (*town*) and "kətəmari" (*from student*). The "kə" in "**kə**təma" is part of the word, and has no grammatical or other meaning. Whereas, the "kə" in "**kə**təmari" is a morph that signifies "from".

In order to disambiguate such situations, a context information of a character should be known. The context information of a character can be obtained from the character's neighbors [67]. *Window based classification method* enables to capture local contexts by considering all successive windows of size w sliding over a given word. The method generates past and future fixed-width character context for a given character. For example, consider an Amharic word *"betu", "the house"* having $n = 4$ characters ($c_{(1)} = b$, $c_{(2)} = e$, $c_{(3)} = t$ and $c_{(4)} = u$). As depicted in Table 2.8, the context characters for each character $c_{(t)}$ ($1 \leq t \leq 4$) is given by $c_{(t-2)}$, $c_{(t-1)}$, $c_{(t)}$, $c_{(t+1)}$, $c_{(t+2)}$ for a window size of $w = 2$ [68]. In other words, two neighboring characters from left and two from right. In order to have a uniform feature size, slots are filled with asterisk "*" where there are no neighboring characters.

TABLE 2.8: Context features of the word "betu" using a sliding window of size=2. The central character is **boldfaced** and it's corresponding label is given based on the "BMES" scheme.

$C_{(t\text{-}2)}$	$C_{(t-1)}$	$C_{(t)}$	$C_{(t+1)}$	$C_{(t+2)}$	Label
*	*	**b**	e	t	B
*	b	**e**	t	u	M
b	e	**t**	u	*	E
e	t	**u**	*	*	S

2.7 Sequence Labeling Models

A sequence is an ordered list of events. For example, a given transliterated (see Figure Annex .2 on page 71) Amharic word *"betu", "the house"* can be represented as a simple time series sequence of characters from time stamp t_1 to t_n as $\langle (t_1, b)(t_2, e)(t_3, t)(t_4, u) \rangle$ [69].

Classification is a type of supervised learning that requires dataset that have already been classified (labeled) as training dataset [69].

Sequence labeling is a classification task that involves the algorithmic assignment of a categorical label to each member of a sequence of observed values [70]. The problem can be formulated as [71]:

> Given an input sequence, represented by an n-dimensional vector $X = (x_1, x_2, ..., x_n)$, each x_i is assumed to have a predefined class label y_i, the goal is to construct a classifier h that can correctly predict a new label sequence $y = h(x) = (y_1, y_2, ..., y_n)$.

NLP applications that are considered as sequence labeling tasks include named entity recognition (NER), part-of-speech tagging (POS), word segmentation (WS), chunking and morphological segmentation (MS). For example, in POS tagging problem [71], one (X, y) pair might consist of:

$$X = \langle \text{do, you, read, books} \rangle \text{ and}$$
$$y = \langle \text{verb, pronoun, verb, noun} \rangle.$$

As words are sequences of morphemes and morphemes, in turn, are sequences of characters, a word segmentation problem can be addressed as a sequence classification task.

2.8 Sequence to Sequence Models

In this section, we present a brief overview of sequence to sequence models including those that use attention mechanism.

2.8.1 Seq2seq Model

Seq2seq is an algorithm that was first developed by Google for use in machine translation [72]. Its applications include language translation, image captioning, text summarization and MS [6].

A seq2seq model, requires two LSTM models. The first LSTM model reads the input sequence to get a fixed-dimensional vector representation. The second LSTM is basically a RNN language model conditioned on the input sequence. Typically, the first network is called an *encoder* and the second, a *decoder* (see Figure 2.13).

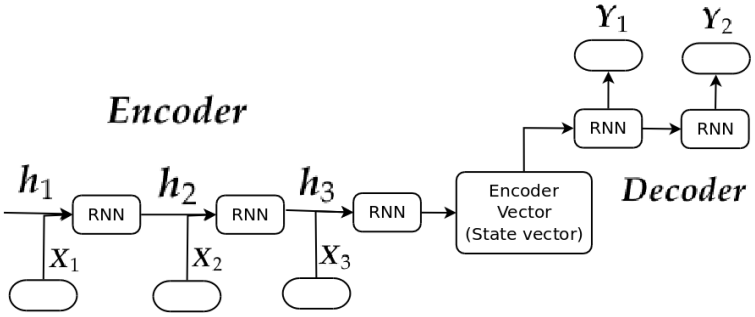

FIGURE 2.13: Encoder-decoder architecture (modified from [73]).

1. **Encoder:** It takes an input sequence and returns an output and the network's internal state. For example, given a sequence of vectors $\boldsymbol{x} = (x_1, \ldots, x_T)$, it converts them into a fixed-length vector $\boldsymbol{c}$. The encoder is an RNN, typically LSTM, such that [73]:

$$h_t = f(x_t, h_{t-1})$$
$$\boldsymbol{c} = g(h_1, \ldots, h_T)$$

where, h_t = hidden state at time t, $\boldsymbol{c}$ = vector generated from the hidden states, and f & g are some nonlinear functions.

2. **Decoder:** It takes the state from the encoder (called *context* or *conditioning* or *thought vector*) as input. It then predicts the target sequence at each time step given the output of the previous time step. For example, it predicts the next target y_t given the context vector $\boldsymbol{c}$ and all the previously predicted targets $(y_1, \ldots, y_{t-1})$, that means, it calculates a conditional probability over the translation $\boldsymbol{y}$ working on the joint probability [73]:

$$
\begin{aligned}
p(\boldsymbol{y}) &= \prod_{i=1}^{x} p\left(y_t | y_1, \ldots, y_{t-1}, c\right) \\
&= \prod_{i=1}^{x} g\left(y_{t-1}, s_t, c\right)
\end{aligned}
$$

where, $\boldsymbol{y} = (y_1, \ldots, y_t)$ and $p\left(y_t | y_1, \ldots, y_{t-1}, c\right) = g\left(y_{t-1}, s_t, c\right)$, g is a nonlinear function that outputs the probability of y_t, s_t is the value of the hidden state of the current position, and $\boldsymbol{c}$ the context vector.

Figure 2.14 demonstrates an example of an encoder-decoder architecture when used for segmentation. In this particular situation, the decoder may learn to generate the "translation" of the encoded characters into a target segmentation.

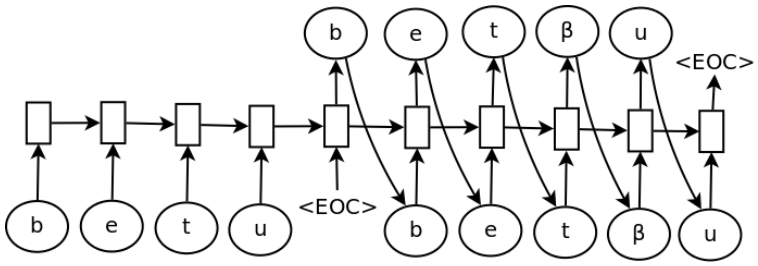

FIGURE 2.14: An instance of an encoder-decoder model for segmentation of the word "bet-u" = "bet" β "u". ⟨EOC⟩ stands for End-Of-Character.

2.8.2 Seq2Seq Model with Attention

Recently, the most commonly used seq2seq modeling architecture is an attention [74] augmented RNN [28, 42].

An architecture that use an attention mechanism does not encode an input sequence into a single fixed size representation. Instead, it learns how to generate an input representation for each output time step.

Attention helps the network to focus on specific inputs. This is achievable by letting the decoder to get the value of the hidden state at each step in the input sequence. Attention mechanism increases the performance accuracy of a machine translation model [60].

Attention Mechanism

In seq2seq modeling with attention, the decoder does some kind of search, that allows it to look at the whole source sequence when it needs to produce an output sequence. This process is called an attention mechanism. Figure 2.15 illustrates the situation graphically.

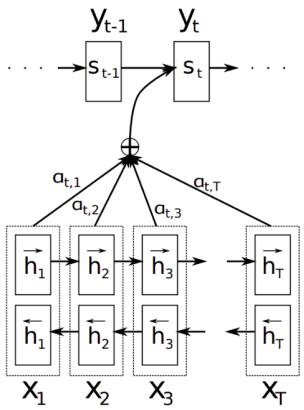

FIGURE 2.15: The attention mechanism. (Taken from [74].)

The aim is to replace the fixed-length context vector by another context vector $\boldsymbol{c}_i$ which is a sum of the hidden states of the input sequence, weighted by alignment scores. That is, the probability of each output target is conditioned on a distinct context vector $\boldsymbol{c}_i$ for each target item $\boldsymbol{y}$ as:

$$p\left(y_t|y_1,\ldots,y_{t-1},c\right) = g\left(y_{t-1},s_i,c\right)$$

where, $\boldsymbol{s}_i = f\left(s_{i-1},y_{i-1},c_i\right)$ (the hidden state for time, i).

Context Vector

The context vector c_i is the sum of the hidden states of the input sequence, weighted by alignment scores:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

The weight α_{ij} of each hidden state h_j is calculated by: $\alpha_{ij} = softmax(e_{ij})$ and $e_{ij} = a(s_{i-1,h_j})$. a is an alignment model which scores how well the inputs around position j and the output at position i match. $a(s_{i-1,h_j}) = \boldsymbol{V}_a tanh(\boldsymbol{W} s_{i-1} + \boldsymbol{U}_a h_j)$. Where, $\boldsymbol{V}_a$ and $\boldsymbol{U}_a$ are weight matrices to be learned in the alignment model. Figures 2.16 and 2.17 respectively show the schematic diagrams of the "seq2seq fixed-context vector" and "seq2seq with attention mechanism" architectures.

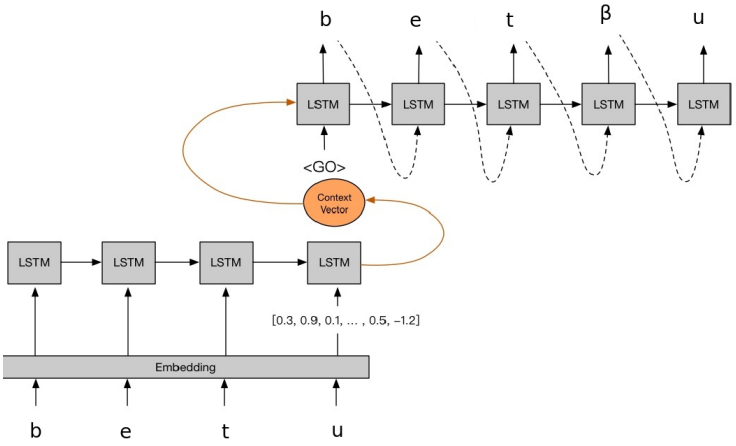

FIGURE 2.16: Encoder-Decoder with fixed-context vector (from [74]).

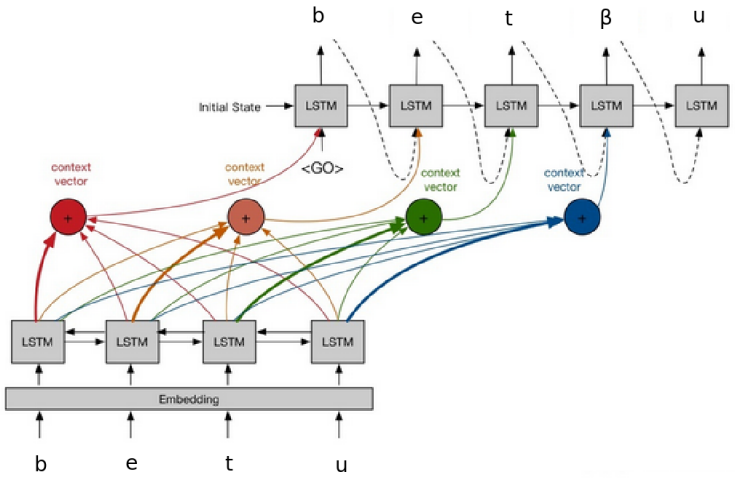

FIGURE 2.17: Encoder-Decoder with attention mechanism (from [74]).

2.9 Evaluation Metrics

One of the essential tasks in building a classifier is to evaluate its performance [75]. Performance evaluation is a core task since it determines the goodness of a classifier [76]. Evaluation enables to estimate the predictive power of a given classifier when applied on new and future data [77]. Performance measure is done by comparing predicted class labels with true class labels. For this, the usual way is to hold out a sample of data that has been labeled with the target from the training dataset [76].

Accuracy, precision and recall are among the various metrics to evaluate the goodness of a classifier [78]. These metrics are based on a "confusion matrix" [79] associated with the classifier. A confusion matrix is a two-dimensional square matrix that depicts the number of occurrences between true and predicted classes. That is, it contains information about actual and predicted classifications as shown in Table 2.9.

TABLE 2.9: Confusion matrix for a binary classifier.

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

2.9.1 Evaluation of a Binary Classifier

This section describes evaluation metrics for a single prediction of a binary classification problem. As depicted in Table 2.9, a single prediction has four possible outcomes: TP, FN, FP and TN. These outcomes are the basis for metrics such as accuracy, precision, recall, and F1 Score [80]. Following is a description list of Table 2.9 and the various metrics.

1. **True Positive** (TP): Positive class correctly identified as positive.
2. **False Negative** (FN): Positive class incorrectly identified as negative.
3. **False positive** (FP): Negative class incorrectly identified as positive.
4. **True Negative** (TN): Negative class correctly identified as negative.
5. **Accuracy** $= \dfrac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$.
   It measures how frequently a classifier makes the correct prediction [75].
6. **Precision** $= \dfrac{\text{TP}}{\text{TP} + \text{FP}}$.
   It answers the question: "what proportion of predicted Positives is truly Positive?" [81].
7. **Recall** $= \dfrac{\text{TP}}{\text{TP} + \text{FN}}$.
   It answers the question: "what proportion of actual Positives is correctly classified?" [81].
8. **F1 Score** $= \dfrac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

   It is the harmonic mean of recall and precision.

2.9.2 Evaluation of a Multiclass Classifier

A multiclass classifier is supposed to categorize each dataset into 1 of K number of classes [81]. In this scenario, a confusion matrix for every class $c_k \in C = 1, \ldots, K$ should be calculated. The k^{th} confusion matrix considers class c_k as the positive class and all other classes c_j with $j \neq k$ as the negative class [82]. Precision and recall are calculated, for each class, in the same fashion as in the binary classification [79].

2.9.3 Percentage Difference and Percentage Change

Both, percentage difference (PD) and percentage change (PC), show a difference between two values as a percentage of one (or both) values [83]. PD is used

when one value is not obviously older or better than the other. PC is about comparing **old** to **new** values. One uses the following formula to calculate PD and PC.

$$\text{Percentage Change} = \frac{\text{New Value} - \text{Old Value}}{|\text{Old Value}|}$$

$$\text{Percentage Difference} = |\frac{\text{First Value} - \text{Second Value}}{(\text{First Value} + \text{Second Value})/2}|$$

3 Related Work

3.1 Introduction

Word segmentation is considered as a fundamental step in most downstream NLP applications. It is also taken as a component of those applications rather than a standalone morphological segmentation tool. We have also observed that *morphological segmentation* is interchangeably used with a fully-fledged task of *morphological analysis*. Our task focuses on the task of boundary detection (a.k.a, morphological segmentation).

Accordingly, this chapter presents those works related to boundary detection for foreign languages and Amharic.

3.2 Segmentation for Foreign Languages

WS is regarded as a first step for almost all Semitic languages [4]. In 2018, the first LSTM based morphological segmentation research for Tigrinya was conducted [1]. The research reported that BiLSTM is the best performing model for Tigrinya MS. In the study, [1] compared three different segmentation models: CRF, LSTM and BiLSTM model families. They used a manually annotated morphologically segmented Tigrinya corpus (around 45,127 words) to get an F1 score of 94.67%.

Recently, MSs with seq2seq NNs, using encoder-decoder attention-based models, have gained success. One such work is a MS task for Russian language [6]. In their work, [6] defined MS as sequence transduction using character embeddings (about 33 Russian characters). They used the architecture and the hyperparameters by [84]. Their encoding follows the attention mechanism as described in [16]. Their encoder-decoder model encodes the input as a sequence of Russian characters and outputs target characters.

Another related work which gained success using seq2seq translation model is a CWS [46]. Like the Russian [6], they used an attention-based encoder-decoder

framework. Their model gets a global information from inputs and directly outputs a segmented sequence. They also claim that their work addresses Chinese spelling correction with CWS jointly in an end-to-end mode as is done by [84].

Last but not least, the work of [30] presented a canonical segmentation for Indonesian and German languages. They used character-based encoder-decoder framework in a low-resource situation. They claimed higher performance results for those languages.

3.3 Segmentation for Amharic Language

Works related to Amharic morphological segmentation are mostly found as a component part of other downstream applications, such as morphological analyzer by [18]. Another example is the works of [19] and [41] who developed respectively *statistical-based Amharic grammar checker* and *Amharic anaphora resolution system*. They both used "HornMorpho" [23] as their morphological analyzer.

The work of [85] introduced genetic algorithms to generate optimal morphological slot models by applying on Amharic verbs. Another work by [86] applied a supervised machine learning approach to morphological analysis of Amharic verbs using Inductive Logic Programming (ILP) to score an accuracy of 86.99%.

The work of [87] achieved a precision of 94% and a recall of 97% by applying a supervised machine learning approach, using generic background knowledge, to incrementally learn and segment affixes.

3.4 Summary

Our work is similar to the work of [18] when it comes to segmenting a given word into prefix, stem, suffix, and putting a boundary marker between each segment. The difference lies only on the way a training is performed. We used neural network approach as opposed to theirs: memory-based learning (i.e., probabilistic and knowledge-based methods). Similarly, the works by [85], [86] and [87] are methodically different from our work.

Other works, such as by [19] and [41] used "HornMorpho" [23] as their morphological analyzer. HornMorpho is based on finite state transducers (FSTs). It is a fully-fledged morphological analysis tool for Amharic, Tigrinya and Oromo

languages. Our approach is different from HornMorpho: we use NN-based sequence modeling approaches as opposed to FSTs. After all, our work is limited to only WS task [as opposed to a fully-fledged morphological analysis tool].

Our work adapts key components from the work of [1]. However, it is uncertain whether the LSTM approach for Tigrinya is also effective for AMS for several reasons. First, there is a difference between the two languages [22]. Second, they never used GRU for their labeling task. GRU is proved to perform better than LSTM for tasks that involve long texts and small datasets [88]. Amharic is an under resourced language, where GRU may be a good alternative. Third, replicating their process may be challenging [64]. Last, they only approached segmentation as a sequence labeling task [as opposed to seq2seq approach].

Our work, also, adapts most of the techniques used by [6]. However, we are not sure if the methods used for Russian language are equally applicable for AMS. A problem may arise from orthographic variations due to language difference.

Ours is also similar to the work of [46] in many aspects. However, it is not sure if these methods are equally effective for AMS (as for CWS). One challenge, maybe, CWS involves breaking a given **sentence** into meaningful ***words*** whereas the AMS breaks a given **word** into sub-words (***morphs***). Another challenge, possibly, is Amharic orthography (Ge'ez) is different from Chinese (Kanji) which may produce orthographic variations and increase the difficulty of word segmentation. Our works are also different with regard to addressing other problems concurrently. We are not interested to jointly address another problem. Our only focus is AMS.

Our work is also inspired by the work of [30]. However, we are different in some aspects. First, our goals are different: ours is *surface segmentation* rather than *canonical segmentation*. Second, Amharic is a different language family [compared to Indonesian and German].

4 Design of Amharic Segmenter

4.1 Introduction

To develop seq2seq modeling approaches, a three step process is followed. First, an unsupervised technique is applied to generate a MS corpus. Then, AMS models are implemented using a supervised method on the enriched corpus. Finally, the models are evaluated for their performances.

4.2 Architecture of the Proposed Solution

4.2.1 Conceptual Design

This section details the general steps to develop an AMS model without focusing on a particular machine learning tool or framework. Figure 4.1 shows the flowchart of the model.

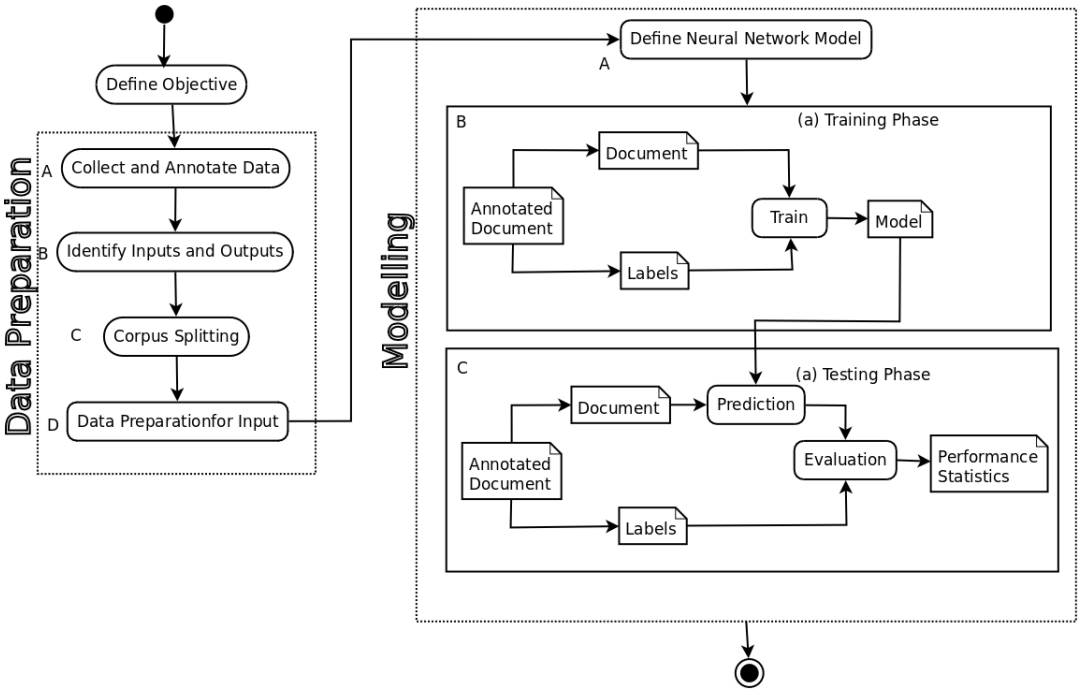

FIGURE 4.1: Flowchart to build an AMS Model.

The flowchart is comprised of three parts: the design objective (§4.2.2), the data preparation (§4.2.3) and the conceptual modeling (§4.2.4).

4.2.2 Define Objective

The objective is *to develop a NN AMS model* that consists of:

1. **Inputs:** Morph-segmented Amharic words.
2. **Process:** Develop a sequence-to-sequence model.
3. **Output:** A predictive model that identifies morph boundaries.

4.2.3 Data Preparation

A) **Collect and annotate data**

The following three steps are taken to manually annotate the raw corpus.

(a) Transliterate the Amharic script.

(b) Morph-break the transliterated word.

- If the approach is a sequence labeling:
  - Label the morph-broken words.
- If the approach is a seq2seq:
  - No need to label.

Figure 4.2 depicts an example of the process using the "BMES" tagging scheme. The sentence has five words, and it is chosen arbitrarily.

No.	Representation	Word-1	Word-2	Word-3	Word-4	Word-5
1	Orthography	ስምምነቱ	የኢትዮጵያን	ስኬት	የሚያሳይ	ነው
2	Transliteration	smmnvtu	yvxityoPyan	sket	yvmiyasay	nvw
3	Morph-break	smmnvt-u	Yv-xityoPya-n	sket	yv-m-iy-as-ay	nvw
4	BMES Tagging	BMMMMES	BEBMMMMMMES	BMME	BESBEBEBE	BME

FIGURE 4.2: An example of a tagging process.

B) **Identify inputs and outputs**

The following steps are followed to identify the inputs and outputs.

(a) Get a vocabulary of labeled words.

- If the approach is a sequence labeling: for every word in the vocabulary separate characters and labels.
- If the approach is a seq2seq: no feature extraction is needed.

C) **Corpus splitting**
Partitioning of data into two datasets:

- Training dataset (90%)
- Testing dataset (10%)

D) **Data preparation for input**
The inputs to a neural network are usually numeric values. So we do the following conversions.

- For the inputs:
  (a) Identify the unique characters in the training dataset.
  (b) Map each unique character to a unique integer.
  (c) Encode the words into encoded integers: an index of each unique character.
- For the outputs: prepare a one-hot-encoding for the class labels.

4.2.4 Modeling

A) **Define neural network model**

(a) Prepare a character embedding;

(b) Define the neural network architectures;

B) **Train the model**
Figure 4.3 illustrates a simple procedure to train a neural network. It all starts from an annotated document. Then, the document is split into the [un annotated] document and it's corresponding labels. The combination shall be given to the chosen training framework. The output is shown to be a trained model.

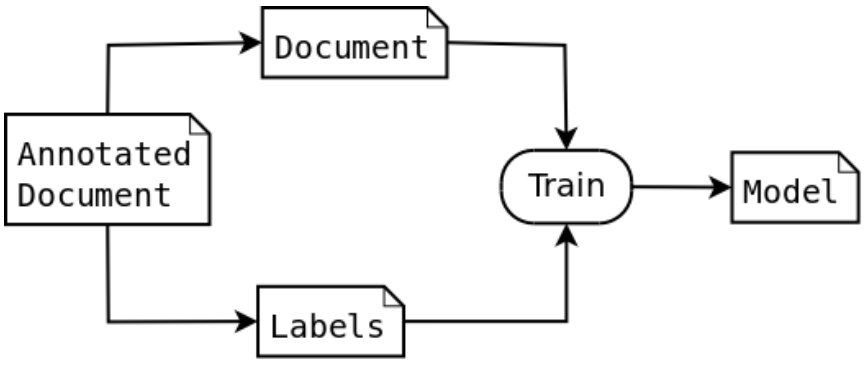

FIGURE 4.3: Training phase.

The detailed training steps are performed as depicted in Figure 2.10 [a flowchart of a procedure to train a neural network]. The procedure needs the following inputs:

(a) Character encoding of the vocabulary;

(b) One-hot-encoded character labels;

(c) The training algorithm;

(d) The validation type and dataset;

(e) The number of iteration;

C) **Test the model**

This step is expected to give answer for the question: "How well is the performance of the model on previously unseen data?".

Figure 4.4 demonstrates the steps to test a neural network. It also starts by splitting an annotated test document. Then, the trained model is applied on test [un annotated] document to predict the expected labels. The expected labels are then compared to the "gold" standard correct labels. Finally, after evaluation, some performance stats shall be displayed.

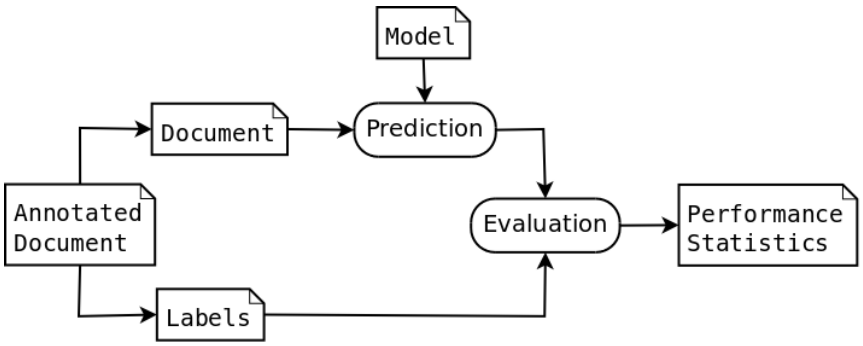

FIGURE 4.4: Testing phase.

4.2.5 Amharic Segmentation Dataset

In a hypothetical scenario, the total population of the segmentation data constitutes all Amharic words. However, as it is not feasible to collect and use all words, sampling technique is used. As a result, some representative stems with a *complete* morph list (136,632 words) is used to prepare a manually segmented Amharic corpus (used for seed dataset) as proposed by [89].

Once the seed dataset is prepared manually (see Figure 4.5 on the following page), it is used to develop a seed segmentation model. Then, the seed model

is applied to generate more dataset by applying unsupervised method on an unlabeled bulk corpus (see Figure 4.6 on the next page). Algorithm 4.2.1 on page 44 depicts the detailed steps to prepare the new dataset.

4.2.6 Preparing a Seed Model

Two kinds of dataset are used to prepare a seed model. The first one is a manually labeled, morphologically segmented corpus (136,632 words), prepared by [89]. As this data has a relatively *complete* morph lists, it is used for two purposes. First, it helped in generating an affixation table. Second, it is used as a part of the training corpus. However, this data has only few representative *stems*. To compensate for the lack of *stem varieties*, another dataset, from Contemporary Amharic Corpus (CACO) by [5], which is a morphological analysis result is used. As the dataset from CACO is not directly applicable for the purpose of this study, it is filtered by applying a regular expression algorithm and the affixation table to get 306,417 words. Finally, the combination of the processed and the filtered data (443,049 words) served as a seed data to train a seed model. Figure 4.5 depicts the complete steps of seed model preparation.

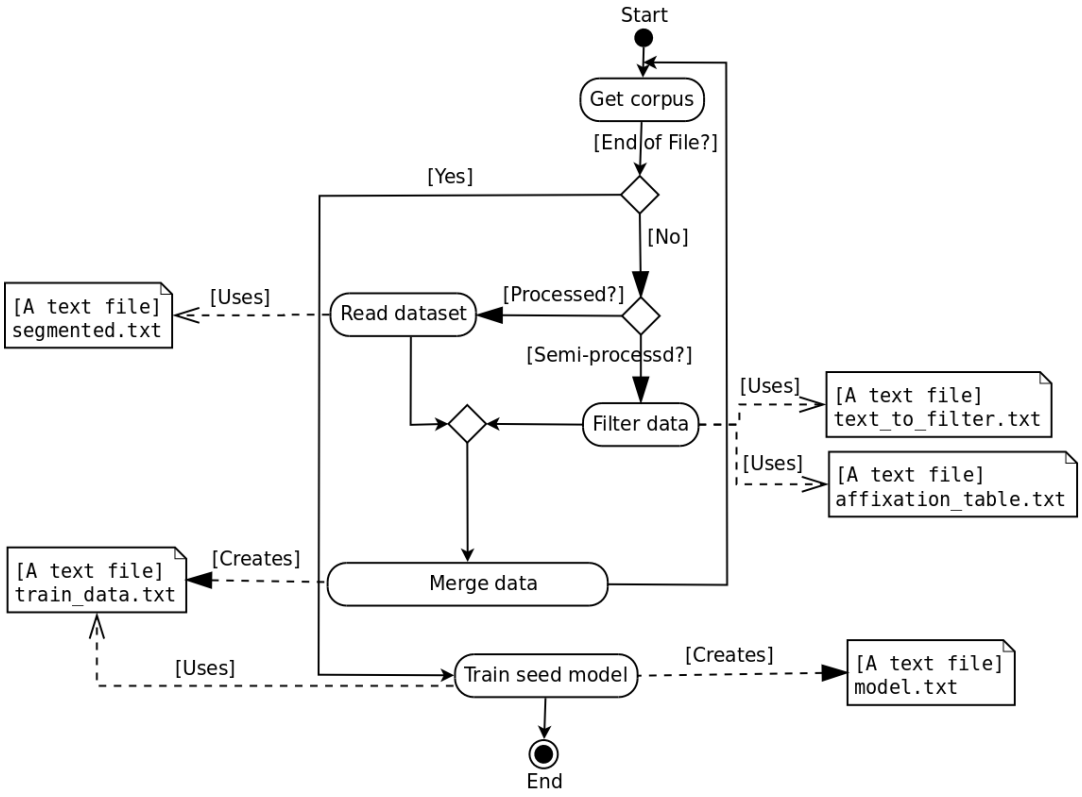

FIGURE 4.5: Seed model preparation.

4.2.7 Data Augmentation

Having the seed model and the *most frequently used Amharic word lists* (147,039 words) from [90] as a bulk raw dataset, some 732,466 segmented words are filtered. Overall, a large corpus (1,175,515 words) that helps to train the seq2seq AMS models is generated. Figure 4.6 and Algorithm 4.2.1 presents the detailed steps toward generating *big-Amharic-morphologically-segmented-dataset*. The linear-chain CRF model of the Wapiti toolkit [91] is used for segmenting and labeling purposes.

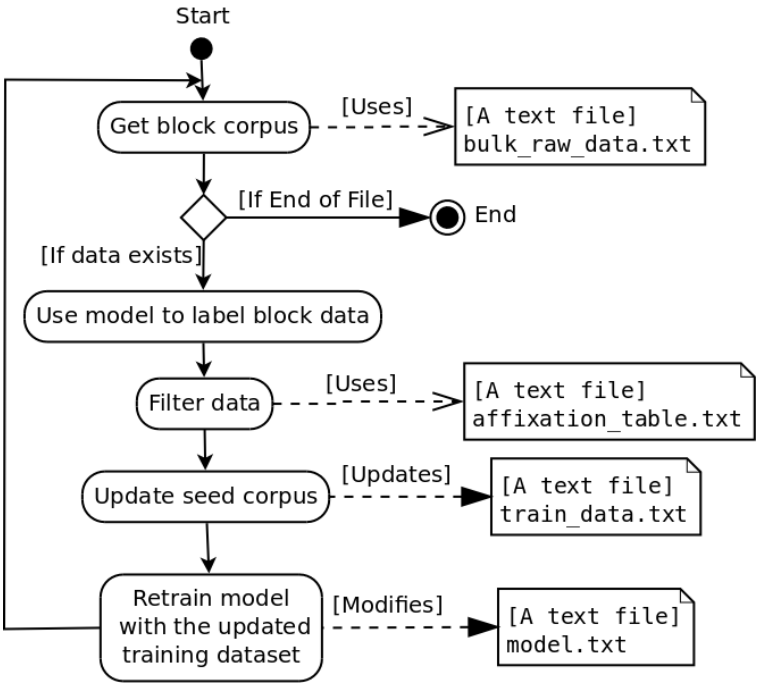

FIGURE 4.6: Data preparation from a raw unsegmented corpus.

Algorithm 4.2.1: GETNEWCORPUS(seedCorpus, bulkCorpus, affixTable)

```
blockSize ← 1000
begin ← 0
end ← blockSize
seedModel ← crfTrain(seedCorpus, 'model.txt')
newCorpus ← seedCorpus
while begin ≤ sizeOf(bulkCorpus)
  do { wordBlock ← getDataBlock(bulkCoupus, begin, end)
       for each word ∈ wordBlock
         do { transWord ← transliterate(word)
              writeOnFile(transWord, 'transWord.txt')
              crfPredict('model.txt', 'transWord.txt', 'result.txt')
              filteredSegments ← filterSegments('result.txt', affixTable)
              newCorpus ← merge(newCorpus, filteredSegments)
              newCorpus ← dropDuplicates(newCorpus)
       newModel ← crfRetrain(newCorpus, 'model.txt')
       begin ← end+1
       end ← end + blockSize
return (newCorpus)
```

4.2.8 AMS as a Sequence Labeling Model

AMS can be treated as a sequence labeling task. This approach expects the input characters to be labeled with their corresponding boundary markers. Example 4.2.1 shows an input-output pair for an Amharic word *"betu", "the house"*.

Example 4.2.1. Expected tag set for the word "betu" using character representation and the "BMES" tagging scheme.

> Given a sequence of characters representing an Amharic word, "betu", "*the house*", as $x_i = \langle$'b' 'e' 't' 'u'$\rangle$, the objective is to *build a classifier* that predicts the sequence of characters that represent the corresponding morph boundaries $y_i = \langle$'B' 'M' 'E' 'S'$\rangle$.

Figure 4.7 further demonstrates the prediction output as: b/B, e/M, t/E, u/S. In other words, the word "betu" is split as $\{bet\}\beta\{u\}$, where "u" is a single morph, definite marker suffix.

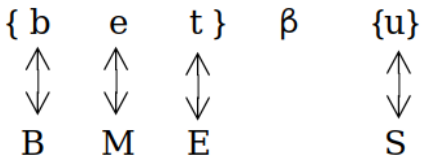

FIGURE 4.7: Segmentation of the word "betu" using the "BMES" tag set.

Table 4.1 presents the four possible experiments. The experiments concentrate on models with a "BMES" tagging scheme.

TABLE 4.1: Possible experiments for sequence labeling model.

No.	MODEL	LABELING
1	GRU	BMES
2	BiGRU	BMES
3	LSTM	BMES
4	BiLSTM	BMES

The architecture of the proposed AMS models is illustrated in Figure 4.8. First, the embedding layer generates embeddings from the character-level features. Next, the embeddings are forwarded to either of the models. Then, a time distributed dense layer processes the output of the network. At the end, a softmax calculates the tag probability distributions over all candidates.

4.2.9 AMS as a Seq2Seq Model

AMS as a seq2seq model gains an overall information from inputs and directly output a segmented sequence without using context features.

Table 4.2 presents a sample input-output pair for a seq2seq AMS model. The input, $\boldsymbol{X}$, is the word "yəbetu", "of the house" having 6 characters. The output is a sequence, $\boldsymbol{y}$, having 8 characters including a boundary marker, β. The final segmentation result is {yə}β{bet}β{u}. Figure 4.9 is a schematic diagram to further illustrate the situation.

TABLE 4.2: Input-output instance for seq2seq AMS model.

	Sequence	**Length**
Input	$\boldsymbol{X}$ = [y, ə, b, e, t, u]	6
Output	$\boldsymbol{y}$ = [y, ə, β, b, e, t, β, u]	8

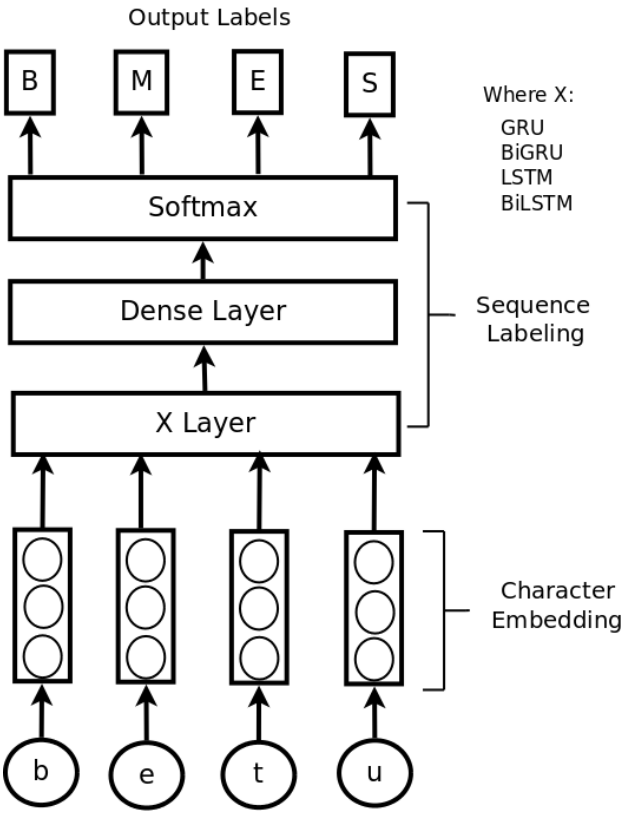

FIGURE 4.8: The general architecture to implement the setups in Table 4.1 (adapted from [1] and [20]).

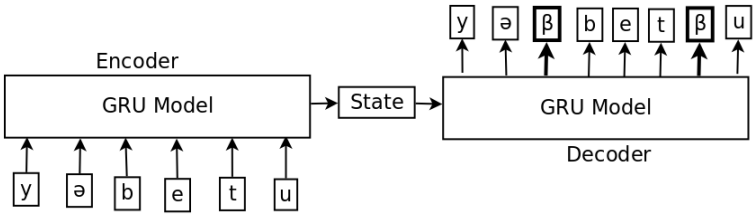

FIGURE 4.9: An example seq2seq architecture of AMS.

The attention-based seq2seq model architecture for AMS is shown in Figure 4.10. The model contains character embedding layer, an Encoder layer, and an attention head and a decoder.

4.3 Evaluation

Two morphological segmentation evaluation approaches are suggested by [92]. The first one is called "direct evaluation", in which the results of a MS model are compared to "gold" standards. The other approach is known as "indirect evaluation", where the MS models are used in some other application such as for speech recognition system.

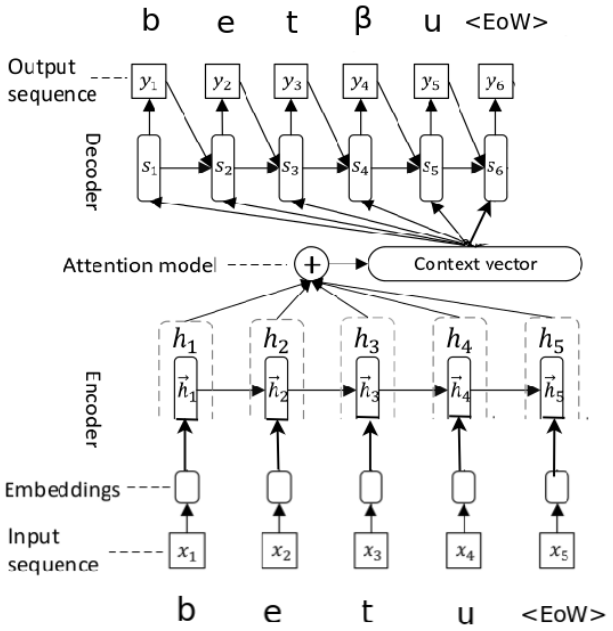

FIGURE 4.10: Architecture of the attention-based seq2seq model. <EoW> stands for End-of-Word. (Adapted from [46])

As recommended by [6], and as the focus of this work is a standalone AMS model, this study uses the direct evaluation technique. So, boundary precision (P), boundary recall (R) and boundary F1-score are reported.

$$\mathrm{P} = \frac{\text{number of correct boundaries found}}{\text{total number of boundaries found}}$$

$$\mathrm{R} = \frac{\text{number of correct boundaries found}}{\text{total number of correct boundaries}}$$

$$\text{F1-score} = \frac{2 \times \mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

Where, "boundary" means the border between morphs. For example, suppose there are two boundaries in the gold standard for the Amharic word "yv-bet-u" (of-the-house). If the AMS model segments this word as "yv-be-t-u", with three boundaries, one can compute precision as 0.67, recall as 1.00 and F1-score as 0.80. The total precision, recall and F1-score are calculated as averages.

5 Experimentation & Evaluation

5.1 Introduction

This section presents the experimental settings for a practical implementation of the AMS models that are described in Chapter 4. In particular, it aims to answer the research questions (stated in RQ1 and RQ2).

RQ1: is meant to examine a sequence labeling model on Amharic morphological segmentation task.

RQ2: is intended to ascertain if a seq2seq model with an attention mechanism perform better than a sequence labeling model on Amharic morphological segmentation task.

5.2 Experimental Procedure

5.2.1 Dataset Collection

The Amharic morphological segmentation corpus is prepared as per the discussion in §4.2.7. The total corpus size is 1,175,515 morphologically segmented Amharic words. This data is divided into two parts: 90% for training and 10% for testing (see Table 5.1). For the sequence labeling models, the training data is prepared (see an excerpt in Table 5.2) as a pair of [input word, target labels]. For the seq2seq model, the targets are morph-broken words (see Table 5.3).

Table 5.1: Data partitioning: training, validation and testing.

Dataset	Proportion	Words
Testing	10%	117,551
Training	90%	1,057,964
		1,175,515

Table 5.2: Sample dataset ready for sequence labeling training as a pair of [input word, target labels].

Input word	Target labels
svbabrona	BMMMMESBE
svbabrana	BMMMMESBE
svbabrvwna	BMMMMESSBE

Table 5.3: Sample dataset ready for seq2seq training as a pair of [input word, target segmented word].

Input word	Segmented word
s v b a b r o n a	s v b a b r - o - n a
s v b a b r a n a	s v b a b r - a - n a
s v b a b r v w n a	s v b a b r - v - w - n a

5.2.2 Tools and Programming Language

The Python programming language is applied for the experimentation. As for the deep learning package, Keras [93] with TensorFlow [94] as a back-end is used. For evaluating the models, the Scikit-learn [95] toolkit is used.

All the experiments are done on a virtual machine (see Table 5.4).

Table 5.4: Virtual machine specification for the experiments.

Operating System	Ubuntu 20.04.2 LTS 64-bit
Processor	Intel(R) Core(TM) i7-6500U CPU
Memory	64 GiB
Storage	1.0 TB

5.3 Experimental Scenarios (ES)

Two experimental scenarios ES1, concerning RQ1, and ES2, focusing on RQ2, are considered.

ES1 is used to examine the sequence labeling task. The models GRU, BiGRU, LSTM and BiLSTM (see Table 4.1) with similar configuration as the settings by [1] are trained (see Figure 4.8).

ES2 is to investigate a sequence-to-sequence task with an attention mechanism. It presents the implementation of the seq2seq models.

Baseline: For a fair comparison, the BiLSTM model with the same hyperparameters settings to that of [1] is considered as a baseline.

5.3.1 ES1 (Sequence Labeling)

- Hyperparameters (ES1)
  Table 5.5 presents the parameters used in the study (from [1]).

Table 5.5: The hyperparameters of the study.

Hyperparameter	Value
Character embeddings	50
Hidden layer units	100
Label set	{B, M, E, S}
Learning rate	0.001
Dropout rate	0.8
Batch size	64
Optimizer	Adam
Epochs	50

- Training (ES1)
  - After the input layer, strings of characters are converted to vectors in the embedding layer (see Figure 4.8). The embedding layer is a vector size of dimension 50. After embedding the characters, a Dropout layer is applied to avoid over-fitting. The dropout rate was set to 0.8. Next, the data moves into the model $\in$ {GRU, BiGRU, LSTM, BiLSTM} layer. A time distributed dense layer is then applied before output. A time distributed dense layer is used because there is one output label $\in$ {B, M, E, S} per time-step or character. Figure 5.1, exported from Keras, presents this procedure schematically.

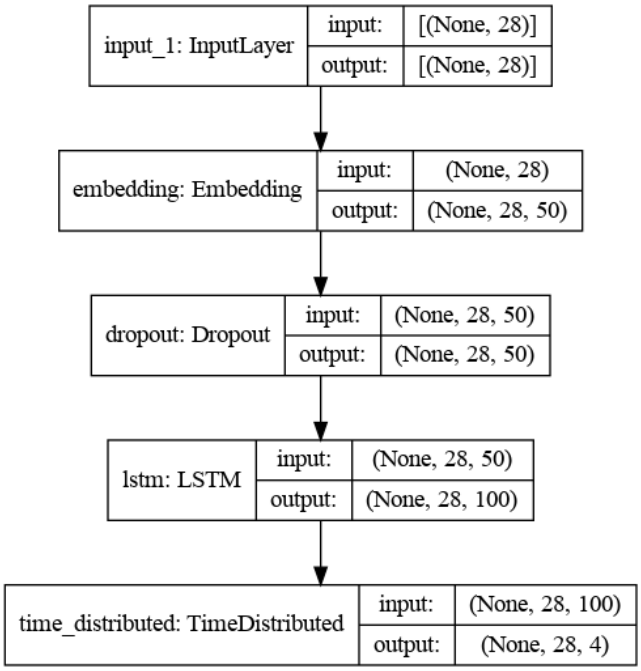

FIGURE 5.1: A schematic diagram of the training procedure for visualization of the LSTM model. This plotting is a representative of all the models.

5.3.2 ES2 (Seq2Seq)

- Hyperparameters (ES2)
  - Inspired by previous works [84, 44], many parameter combinations are explored in the preliminary experiments. The complete list of parameters is shown in Table 5.6. "Hidden layer size" stands for the number of BiLSTM layers or hidden state dimension and "Embeddings dimension" stands for dimensionality of the embedding layer.

TABLE 5.6: Hyperparameters of the Seq2seq training.

HYPERPARAMETER	VALUE
Encoder and decoder	
Number of epochs	10
Number of units	1024
Batch size	64
Character embeddings	256
Optimizer	Adam
Attention	
Attention type	Bahdanau's

- Training (ES2)
  - The training involves three distinct models (an encoder, an attention head and a decoder, in that order) acting as a single end-to-end model.

    Encoder
    It takes a list of tokens. It converts those tokens into vectors by an embedding layer. Then, a model ∈ {GRU, BiGRU, LSTM, BiLSTM} layer processes the vectors sequentially. It outputs the processed sequence (for the attention head) and the internal state (useful to initialize the decoder).

    Bahdanau's additive attention [16]
    It computes the attention weights and the context vectors.

    Decoder
    After accepting the output from the encoder, it converts the tokens into a vector using an embedding layer. The decoder keeps track of what has been generated so far using a similar layer as in the encoder. Finally, it produces context vectors and do logit predictions for the next token.

5.4 Experimental Results

5.4.1 Experimental Results (ES1)

The results of the first scenario are summarized in Table 5.8 with Precision, Recall and F1-score. A sample prediction, using three Amharic words, is given below. Though the words are different, they have identical tag: BMEB-MEBMESSBE (from the "gold" standard).

1. **ስለቤታችንንምኮ**/slv**b**etacnnmko/
2. **ስለሴታችንንምኮ**/slv**s**etacnnmko/
3. **ስለቄባችንንምኮ**/slv**q**e**b**acnnmko/

Based on the sample prediction (using the three words), one can calculate precision, recall and f1-score as shown in Table 5.7.

Table 5.7: A prediction using the LSTM model for ES1.

Word	Predicted Tags	P(%)	R(%)	F1(%)
1	BMEBMEBMESSBE	100	100	100
2	BMEBMMMMMMEBE	100	40	57
3	BMEBMEBMMMSBE	100	60	75

Below (Table 5.8) are the macro averages: precision, recall and f1-score for ES1.

Table 5.8: Results of ES1 using the BMES tagging scheme.

Model	Precision	Recall	F1-score
GRU	90.81	91.34	91.07
LSTM	91.33	91.88	91.60
BiGRU	93.45	92.93	93.19
BiLSTM	94.53	95.12	94.82

5.4.2 Experimental Results (ES2)

The results of the second scenario are summarized in Table 5.9 with Precision, Recall and F1-score. Figure 5.2 presents the attention weights for the input characters: “slvbetacnnmko”. The output is the “morph broken characters” which satisfy the given “gold” standard.

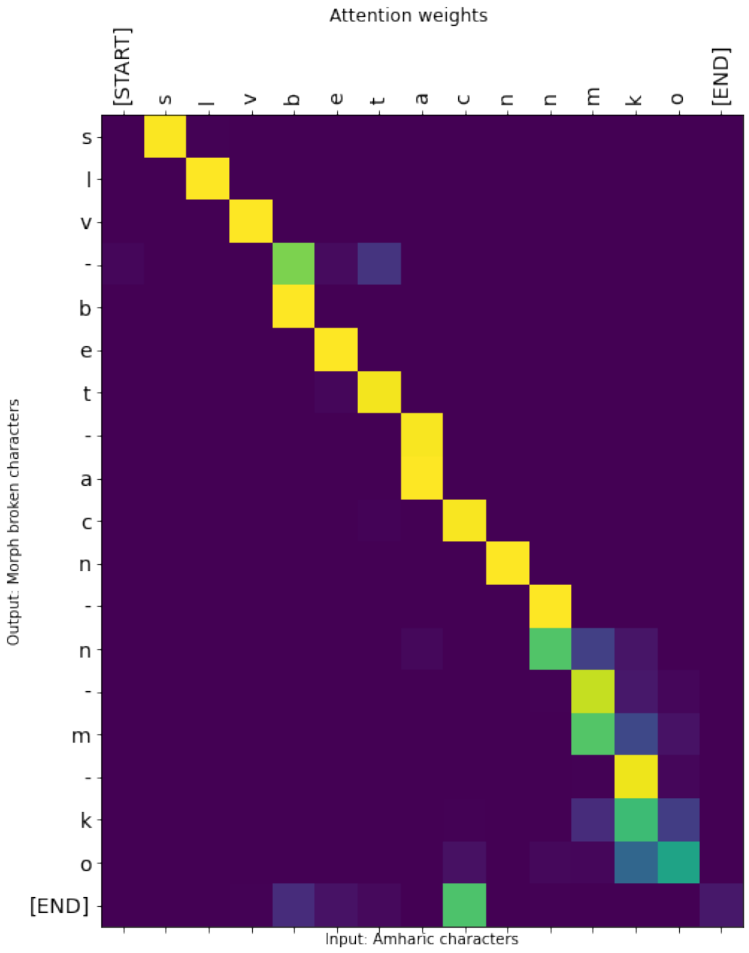

FIGURE 5.2: Attention weights for ES2 with inputs (Amharic characters) and outputs (morph broken characters) of the inputs.

TABLE 5.9: Results of ES2 (seq2seq).

MODEL	PRECISION	RECALL	F1-SCORE
GRU	91.73	92.56	92.14
LSTM	92.47	93.36	92.91
BiGRU	95.58	95.95	95.76
BiLSTM	97.47	97.84	97.65

5.5 Discussion

In the literature, the term "sequence-to-sequence" can refer the task of *sequence labeling* and *seq2seq*. This exchangeable usage arises from the fact that both

inputs and outputs of these tasks are sequences.

In this paper, the outputs for the task of *sequence labeling* are sequenced combinations of labels. However, the outputs of our seq2seq task are identical set of characters for the word under consideration, with some boundary marks.

Based on these differences, we conducted two experimental scenarios. For each scenario, we used four recurrent neural network models (GRU, LSTM, BiGRU and BiLSTM). Overall, the sequence-to-sequence approach outperformed the sequence-labeling method for all the models involved.

5.5.1 Comparison of the Models in ES1

Figure 5.3 shows a scatter plot of a comparison between the F1-scores for the results of the first experimental scenario (ES1).

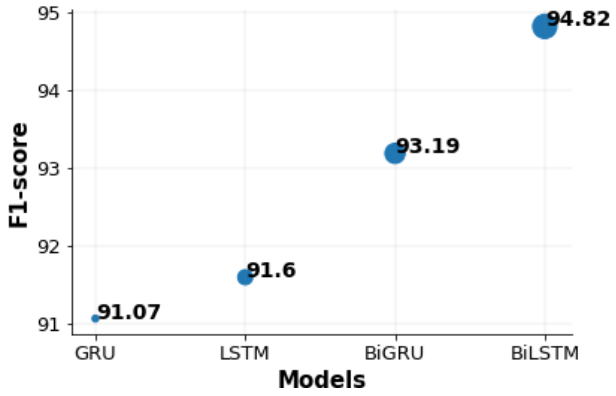

FIGURE 5.3: A scatter plot comparing F1-scores of ES1.

The graph depicts that, the BiLSTM model has the highest F1-score. On the other extreme, the GRU model has the lowest F1-score. In general, the bidirectional models (BiGRU and BiLSTM) perform better than the unidirectional models (GRU and LSTM).

This general trend is inline with the results obtained by [1] when taking LSTM and BiLSTM models. However, our F1-score for the BiLSTM model has shown a slightly higher percentage change, of 0.16% than theirs.

Table 5.10 displays the percentage differences (see §2.9.3) between the various models. Looking at the first row, the percentage difference between the GRU and the LSTM models is insignificant. Because of this, we do not conclude that one is better than the other, as declared by [88]. Nevertheless, one may use

the GRU, for some performance thresholds, instead of the LSTM, if network complexity is an issue.

TABLE 5.10: The percentage difference between the F-score values of the models under the columns *Reference* and *Compared with* for the sequence labeling modeling approaches.

Reference	Compared with	%age Difference
GRU	LSTM	0.58
	BiGRU	2.30
	BiLSTM	4.03
LSTM	BiGRU	1.72
	BiLSTM	3.45
BiGRU	BiLSTM	1.73

5.5.2 Comparison of the Models in ES2

Figure 5.4 shows a scatter plot of a comparison between the F1-scores for the results of the second experimental scenario (ES2).

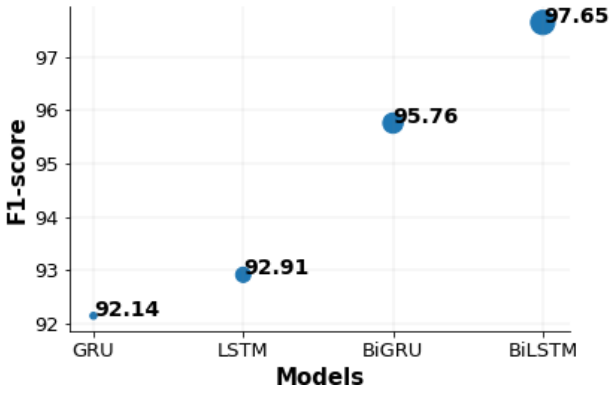

FIGURE 5.4: A scatter plot comparing F1-scores of ES2.

The graph depicts that, the BiLSTM model has the highest F1-score. The GRU model has the least F1-score. Overall, the bidirectional models (BiGRU and BiLSTM) perform better than the unidirectional models (GRU and LSTM).

If one considers the BiLSTM model of [1] as a baseline, our F1-score for the BiLSTM model for the seq2seq approach has a 3.15% higher percentage change.

Table 5.11 presents the details of the percentage differences between the different models. The least difference is scored between the GRU and the LSTM (0.83%).

TABLE 5.11: The percentage difference between the F-score values of the models under the columns *Reference* and *Compare with* for the seq2seq modeling approaches.

Reference	Compare with	%age Difference
GRU	LSTM	0.83
	BiGRU	3.85
	BiLSTM	5.81
LSTM	BiGRU	3.02
	BiLSTM	4.97
BiGRU	BiLSTM	1.95

5.5.3 Seq2seq vs Sequence Labeling

Figure 5.5 displays F1-score comparisons by overlaying the scatter plots of both experimental scenarios (ES1 and ES2).

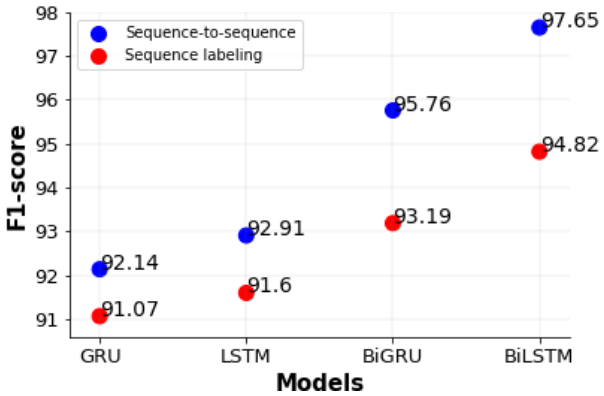

FIGURE 5.5: An overlay of the scatter plots of ES1 and ES2.

The graph depicts that, using similar dataset, the sequence-to-sequence approach is better than the sequence-labeling method for all the models involved. However, the two approaches appear to perform approximately equally for the GRU and LSTM models. The small difference can probably be explained as chance variation, rather than being a real difference in performance between the models. For this reason, one can not draw a conclusion that the sequence-to-sequence method is better overall.

6 Conclusion

6.1 Overview

Morphological segmentation is one of the basic components in Amharic NLP applications. Amharic is an under-resourced, under-studied and morphologically complex language. It has a much greater number of words than morphemes. The rich inflectional and derivational patterns in Amharic create sizable word-forms that worsen data sparsity. As a result, most Amharic NLP applications, that use words as a *minimal unit*, face the challenge of handling rare and unknown words.

To improve these issues, we examined MS as a supervised neural sequence-to-sequence and sequence labeling approaches using character embeddings. To realize this, we constructed a morphologically segmented corpus of 1,175,515 words.

Generally speaking, the sequence-to-sequence approach outperformed the sequence-labeling method when comparing corresponding models (i.e., GRU vs GRU, LSTM vs LSTM, etc.). We achieved an F1-score of 97.65% (state-of-the-art) using the BiLSTM sequence-to-sequence method. Considering the BiLSTM model of [1] as a baseline, our F1-score for the BiLSTM model for the seq2seq approach has a 3.15% higher percentage change.

Our expectation is that, this research motivates the understanding of the properties and the processing challenges of Amharic. We expect to inspire further research by releasing the resources (the corpus, the algorithm and the Python code) of this research to the public.

6.2 Contribution of the Study

- Morphological segmentation helps us to obtain *basic vocabulary units* for downstream language technologies. Among these technologies are included part of speech taggers, stemmers, machine translators, morphological analyzers, spelling and grammar checkers, speech and text understanding, statistical language modeling, and information retrieval.

- Morphological segmentation *improves out-of-vocabulary words* (OOV) challenge.
- We think that our work *inspires further researches* in terms of, say, using our dataset and methods.
- Through our work, the research communities and academicians will understand *the properties and language processing challenges* of Amharic.

6.3 Future Work and Recommendation

This study can motivate future works in several ways. First, to provide a more comprehensive comparison, it would be necessary to measure the performance of the models on additional datasets. Second, one may use different inference layers, which have an impact on performance of a morphological segmentation task. For example, models with a CRF inference layer are observed to outperform models with softmax layer on NER and chunking tasks [64, 20]. Third, using state-of-the-art technology, such as the *transformer architecture* may further improves the accuracy of a MS. Furthermore, one may relax the scope of this study from surface segmentation to canonical segmentation, which is said to be more useful for fully-fledged morphological analyzers [96]. Finally, one may examine the impact of integrating a morphological segmenter into downstream applications, such as part-of-speech tagging and machine translation [7].

References

[1] Yemane Tedla and Kazuhide Yamamoto. "Morphological Segmentation with LSTM Neural Networks for Tigrinya". In: *International Journal on Natural Language Computing* 7 (Apr. 2018), pp. 29–44.

[2] Mathias Creutz and Krista Lagus. "Unsupervised Models for Morpheme Segmentation and Morphology Learning". In: *ACM Trans. Speech Lang. Process.* 4.1 (Feb. 2007), 3:1–3:34. ISSN: 1550-4875.

[3] Ryan Cotterell, Arun Kumar, and Hinrich Schütze. "Morphological Segmentation Inside-Out". In: Jan. 2016, pp. 2325–2330. DOI: 10.18653/v1/D16-1256.

[4] Markus Walther. "Computational Nonlinear Morphology with Emphasis on Semitic Languages". In: *Computational Linguistics* 28 (Dec. 2002). George Anton Kiraz (Beth Mardutho: The Syriac Institute) Cambridge: Cambridge University Press (Studies in natural language processing, edited by Branimir Boguraev and Steven Bird), pp. 576–581.

[5] Andargachew Mekonnen, Michael Gasser, Andreas Nürnberger, and Binyam Seyoum. "Contemporary Amharic Corpus: Automatically Morpho-Syntactically Tagged Amharic Corpus". In: (Oct. 2018).

[6] N. V. AREFYEV, T. Y. GRATSIANOVA, and K. P. POPOV. "Morphological segmentation with sequence to sequence neural network". In: *Komp'juternaja Lingvistika i Intellektual'nye Tehnologii*. 2018, pp. 85–95.

[7] Yemane Tedla and Kazuhide Yamamoto. "Morphological Segmentation for English-to-Tigrinya Statistical MachineTranslation". In: *Int. J. Asian Lang. Process* 27.2 (2017), pp. 95–110.

[8] L. Wang, Z. Cao, Y. Xia, and Gerard de Melo. "Morphological Segmentation with Window LSTM Neural Networks". In: *AAAI*. 2016.

[9] Hassan Al-Haj and Alon Lavie. "The impact of Arabic morphological segmentation on broad-coverage English-to-Arabic statistical machine translation". In: *Machine translation* 26.1-2 (2012), pp. 3–24.

[10] Roy Bar-Haim, Khalil Sima'an, and Yoad Winter. "Choosing an optimal architecture for segmentation and POS-tagging of Modern Hebrew". In: *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*. Association for Computational Linguistics. 2005, pp. 39–46.

[11] Mequanent Argaw. “Amharic Parts-of-Speech Tagger using Neural Word Embeddings as Features”. M.S. thesis. Addis Ababa, Ethiopia: Dept. Electrical and Computer. Eng., Addis Ababa Univ., 2019.

[12] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining Concepts and Techniques*. Apress, 2012. ISBN: 978-0-12-381479-1.

[13] Nikola Ljubešić. “Comparing CRF and LSTM performance on the task of morphosyntactic tagging of non-standard varieties of South Slavic languages”. In: *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. 2018, pp. 156–163.

[14] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. “Natural Language Processing (Almost) from Scratch”. In: *Journal of Machine Learning Research* 12.76 (2011), pp. 2493–2537. URL: http://jmlr.org/papers/v12/collobert11a.html.

[15] Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan. “Language Model Based Arabic Word Segmentation.” In: 2003, pp. 399–406. DOI: 10.3115/1075096.1075147.

[16] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. 2014. arXiv: 1406.1078 [cs.CL].

[17] Burcu Can and Suresh Manandhar. “Unsupervised Learning of Morphology by using Syntactic Categories.” In:

[18] Mesfin Abate and Yaregal Assabie. “Development of Amharic Morphological Analyzer Using Memory-Based Learning”. In: *Proceedings of the 9th International Conference on Natural Language Processing (PolTAL2014)*. Vol. 8686. Springer Lecture Notes in Artificial Intelligence (LNAI), 2014, pp. 1–13.

[19] Aynadis Temesgen Gebru and Yaregal Assabie. “Development of Amharic Grammar Checker Using Morphological Features of Words and N-Gram Based Probabilistic Methods”. In: *Proceedings of the The 13th International Conference on Parsing Technologies (IWPT2013)*. 2013, pp. 106–112.

[20] Zhiheng Huang, Wei Xu, and Kai Yu. *Bidirectional LSTM-CRF Models for Sequence Tagging*. cite arxiv:1508.01991. 2015. URL: http://arxiv.org/abs/1508.01991.

[21] Yoshiaki Kitagawa and Mamoru Komachi. “Long Short-Term Memory for Japanese Word Segmentation”. In: (Sept. 2017).

[22] Tekabe Legesse Feleke. “The Similarity and Mutual Intelligibility between Amharic and Tigrigna Varieties”. In: Jan. 2017, pp. 47–54.

[23] Michael Gasser. *HORNMORPHO 2.5 User’s Guide*. 2012. URL: gasser@cs.indiana.edu.

[24] Sebastian Reiner Spiegler. “Machine learning for the analysis of morphologically complex languages”. PhD thesis. University of Bristol, 2011.

[25] Stig-Arne Grönroos, Sami Virpioja, and Mikko Kurimo. “North Sámi morphological segmentation with low-resource semi-supervised sequence labeling”. In: *Proceedings of the Fifth International Workshop on Computational Linguistics for Uralic Languages*. Tartu, Estonia: Association for Computational Linguistics, Jan. 2019, pp. 15–26. DOI: 10.18653/v1/W19-0302. URL: https://www.aclweb.org/anthology/W19-0302.

[26] Marek Rei, Gamal K. O. Crichton, and Sampo Pyysalo. “Attending to Characters in Neural Sequence Labeling Models”. In: *CoRR* abs/1611.04361 (2016). arXiv: 1611.04361. URL: http://arxiv.org/abs/1611.04361.

[27] Abdulrahman Almuhareb, Waleed Alsanie, and Abdulmohsen Al-Thubaity. “Arabic Word Segmentation With Long Short-Term Memory Neural Networks and Word Embedding”. In: *IEEE Access* 7 (2019), pp. 12879–12887. DOI: 10.1109/access.2019.2893460. URL: https://doi.org/10.1109%2Faccess.2019.2893460.

[28] Rohit Prabhavalkar, Kanishka Rao, Tara Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly. “A Comparison of Sequence-to-Sequence Models for Speech Recognition”. In: 2017. URL: http://www.isca-speech.org/archive/Interspeech_2017/pdfs/0233.PDF.

[29] Hana Yousuf, Michael Gaid, Said Salloum, and Khaled Shaalan. “A Systematic Review on Sequence to Sequence Neural Network and its Models”. In: *International Journal of Electrical and Computer Engineering* 11 (Oct. 2020).

[30] Tatyana Ruzsics and Tanja Samardžić. “Neural Sequence-to-sequence Learning of Internal Word Structure”. In: *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, pp. 184–194. DOI: 10.18653/v1/K17-1020. URL: https://aclanthology.org/K17-1020.

[31] Mikko Kurimo, Mathias Creutz, Matti Varjokallio, Ebru Arisoy, and Murat Saraçlar. “Unsupervised segmentation of words into morphemes - Challenge 2005, An Introduction and Evaluation Report”. In: 2006.

[32] Imed Zitouni. *Natural Language Processing of Semitic Languages*. Springer Publishing Company, Incorporated, 2014. ISBN: 3642453570, 9783642453571.

[33] Alejandro Gutman and Beatriz Avanzati. *Amharic Morphology*. 2013. URL: http://www.languagesgulper.com/eng/Amharic.html (visited on 01/20/2021).

[34] S. Weninger, G. Khan, M.P. Streck, and J.C.E. Watson. *The Semitic Languages: An International Handbook*. Handbücher zur Sprach- und Kommunikationswissenschaft / Handbooks of Linguistics and Communication Science [HSK]. De Gruyter, 2011. ISBN: 9783110251586. URL: https://books.google.com.et/books?id=SMzgBLT87MkC.

[35] Baye Yimam. "Root Reductions and Extensions in Amharic". In: 1999.

[36] Peter T. Daniels and William Bright, eds. *The World's Writing Systems*. New York: Oxford University Press, 1996.

[37] Getahun Amare. "Review of Yamariñña Säwasäw". In: *Journal of Ethiopian Studies* 28.2 (1995), pp. 55–60. ISSN: 03042243. URL: http://www.jstor.org/stable/41966049.

[38] Tewodros Abebe Abebawu Eshetu Getenesh Teshome. "Learning Word and Sub-word Vectors for Amharic (Less Resourced Language)". In: *International Journal of Advanced Engineering Research and Science* 7.8 (2020), pp. 358–366. ISSN: 2349-6495(P). DOI: 10.22161/ijaers.78.39. URL: https://ijaers.com/detail/learning-word-and-sub-word-vectors-for-amharic-less-resourced-language/.

[39] Martha Yifiru Tachbelie. "Morphology-based language modeling for Amharic". In: (2010).

[40] John Goldsmith, Jackson Lee, and Aris Xanthos. "Computational Learning of Morphology". In: *Annual Review of Linguistics* 3 (Feb. 2017). DOI: 10.1146/annurev-linguistics-011516-034017.

[41] Temesgen Dawit and Yaregal Assabie. "Amharic Anaphora Resolution Using Knowledge-Poor Approach". In: *Proceedings of the 9th International Conference on Natural Language Processing (PolTAL2014)*. Vol. 8686. Springer Lecture Notes in Artificial Intelligence (LNAI), 2014, pp. 278–289.

[42] Andrej Karpathy. *The Unreasonable Effectiveness of Recurrent Neural Networks*. 2015. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (visited on 05/17/2021).

[43] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[44] Ebrahim Ansari, Zdeněk Žabokrtský, Mohammad Mahmoudi, Hamid Haghdoost, and Jonáš Vidra. "Supervised Morphological Segmentation Using Rich Annotated Lexicon". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. Varna, Bulgaria: INCOMA Ltd., Sept. 2019, pp. 52–61. DOI: 10.26615/978-954-452-056-4_007. URL: https://www.aclweb.org/anthology/R19-1007.

[45] Zachary C Lipton, John Berkowitz, and Charles Elkan. "A critical review of recurrent neural networks for sequence learning". In: *arXiv preprint arXiv:1506.00019* (2015).

[46] Xuewen Shi, Heyan Huang, Ping Jian, Yuhang Guo, Xiaochi Wei, and Yi-Kun Tang. "Neural Chinese Word Segmentation as Sequence to Sequence Translation". In: *Communications in Computer and Information Science*. Springer Singapore, 2017, pp. 91–103. DOI: 10.1007/978-981-10-6805-8_8. URL: https://doi.org/10.1007%2F978-981-10-6805-8_8.

[47] D. Jurafsky and J.H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall series in artificial intelligence. Pearson Prentice Hall, 2009. ISBN: 9780131873216. URL: https://books.google.com.et/books?id=fZmj5UNK8AQC.

[48] Trevor Bihl, William Young, and Gary Weckman. "Artificial Neural Networks for Business Analytics". In: *Encyclopedia of Business Analytics and Optimization*. 2014, pp. 193–208. DOI: 10.4018/978-1-4666-5202-6.ch019.

[49] DeepAI. *What is a Perceptron?* 2021. URL: https://deepai.org/machine-learning-glossary-and-terms/perceptron (visited on 05/08/2021).

[50] Leonardo De Marchi and Laura Mitchell. *Hands-On Neural Networks*. Packt Publishing Ltd., 2019. ISBN: 978-1-78899-259-6.

[51] Manpreet Singh Ghotra and Rajdeep Dua. *Neural Network Programming with TensorFlow*. Packt Publishing Ltd., 2017. ISBN: 978-1-78839-039-2.

[52] Charu C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer, 2018. ISBN: 978-3-319-94463-0.

[53] JJojo Moolayil. *Learn Keras for Deep Neural Networks*. Morgan Kaufmann Publishers, 2019. ISBN: 978-1-4842-4240-7.

[54] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. "LSTM neural networks for language modeling". In: *Thirteenth annual conference of the international speech communication association*. 2012.

[55] Jason Brownlee. *Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras*. 2020. URL: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ (visited on 04/08/2021).

[56] J. Brownlee. *Statistical Methods for Machine Learning: Discover how to Transform Data into Knowledge with Python*. Machine Learning Mastery, 2018. URL: https://books.google.com.et/books?id=386nDwAAQBAJ.

[57] Lei Tang Payam Refaeilzadeh and Huan Liu. "Cross-Validation". In: *Encyclopedia of Database Systems*. Springer Science+Business Media, LLC, 2018, pp. 677–682. ISBN: 978-1-4614-8265-9. DOI: https://doi.org/10.1007/978-1-4614-8265-9.

[58] DeepAI. *Softmax Function*. 2021. URL: https://deepai.org/machine-learning-glossary-and-terms/softmax-layer (visited on 05/16/2021).

[59] J. Brownlee. *Long Short-Term Memory Networks With Python: Develop Sequence Prediction Models with Deep Learning*. Machine Learning Mastery, 2017. URL: https://books.google.com.et/books?id=m7SoDwAAQBAJ.

[60] Mike Bernico. *Deep Learning Quick Reference*. Packt Publishing, 2018. ISBN: 9781788837996. URL: https://www.oreilly.com/library/view/deep-learning-quick/9781788837996/.

[61] Clara Vania. "On understanding character-level models for representing morphology". PhD thesis. The University of Edinburgh, 2020.

[62] Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W. Black, and Isabel Trancoso. "Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation". In: *CoRR* abs/1508.02096 (2015). arXiv: 1508.02096. URL: http://arxiv.org/abs/1508.02096.

[63] O. Campesato. *Artificial Intelligence, Machine Learning, and Deep Learning*. Mercury Learning & Information, 2020. ISBN: 9781683924661. URL: https://books.google.com.et/books?id=pqnNDwAAQBAJ.

[64] Jie Yang, Shuailong Liang, and Yue Zhang. "Design Challenges and Misconceptions in Neural Sequence Labeling". In: *CoRR* abs/1806.04470 (2018). arXiv: 1806.04470. URL: http://arxiv.org/abs/1806.04470.

[65] Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. "Supervised Morphological Segmentation in a Low-Resource Learning Setting using Conditional Random Fields". In: *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 29–37. URL: https://www.aclweb.org/anthology/W13-3504.

[66] A. Zheng and A. Casari. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media, 2018. ISBN: 9781491953198. URL: https://books.google.com.et/books?id=sthSDwAAQBAJ.

[67] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. "Deep Learning for Chinese Word Segmentation and POS Tagging". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 647–657. URL: https://www.aclweb.org/anthology/D13-1061.

[68] Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuan-Jing Huang. "Long short-term memory neural networks for chinese word segmentation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 1197–1206.

[69] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. "A brief survey on sequence classification". In: *ACM Sigkdd Explorations Newsletter* 12.1 (2010), pp. 40–48.

[70] *Sequence labeling. Wikipedia*. Accessed Jan. 25, 2021 [Online]. URL: https://en.wikipedia.org/wiki/Sequence_labeling.

[71] Charu C. Aggarwal. *Discrete Sequence Classification, Data Classification Algorithms and Applications*. CRC Press, 2015. ISBN: 978-1-4665-8675-8.

[72] *Seq2seq. Wikipedia*. Accessed May 26, 2021 [Online]. URL: https://en.wikipedia.org/wiki/Seq2seq.

[73] David S. Batista. *The Attention Mechanism in Natural Language Processing - seq2seq*. 2020. URL: http://www.davidsbatista.net/blog/2020/01/25/Attention-seq2seq/ (visited on 05/29/2021).

[74] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2016. arXiv: 1409.0473 [cs.CL].

[75] Alice Zheng. *Evaluating Machine Learning Models*. O'Reilly Media, Inc., 2015. ISBN: 978-1-491-93246-9.

[76] Amazon Web Services. *Amazon Machine Learning: Developer Guide*. Version Latest. 2021.

[77] Paweł Cichosz. "Assessing the quality of classification models: Performance measures and evaluation procedures". In: *Central European Journal of Engineering* 1.2 (2011), pp. 132–158.

[78] Guozhu Dong and Jian Pei. *Sequence Data Mining*. Springer, 2007. ISBN: 978-0-387-69937-0.

[79] Margherita Grandini, Enrico Bagli, and Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. arXiv: 2008.05756 [stat.ML].

[80] Data Science. *What is Confusion Matrix and Advanced Classification Metrics?* 2019. URL: https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html (visited on 02/27/2021).

[81] Boaz Shmueli. *Multi-Class Metrics Made Simple, Part I: Precision and Recall*. 2019. URL: https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2 (visited on 03/03/2021).

[82] Matthias Döring. *Performance Measures for Multi-Class Problems*. 2018. URL: https://www.datascienceblog.net/post/machine-learning/performance-measures-multi-class-problems/ (visited on 03/03/2021).

[83] MathsIsFun.com. *Percentage Difference, Percentage Error, Percentage Change*. 2021. URL: https://www.mathsisfun.com/data/percentage-difference-vs-error.html (visited on 12/17/2021).

[84] Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. "Massive exploration of neural machine translation architectures". In: *arXiv preprint arXiv:1703.03906* (2017).

[85] Wondwossen Gewe, Michael Gasser, and Baye Yimam. "Automatic Morpheme Slot Identification using Genetic Algorithm". In: Dec. 2013. DOI: 10.13140/2.1.3123.2006.

[86] Wondwossen Mulugeta and Michael Gasser. "Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming". In: 2012.

[87] Wondwossen Gewe, Michael Gasser, and Baye Yimam. "Incremental Learning of Affix Segmentation". In: Dec. 2012. DOI: 10.13140/2.1.1026.0486.

[88] Shudong Yang, Xueying Yu, and Ying Zhou. "LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example". In: June 2020, pp. 98–101. DOI: 10.1109/IWECAI50956.2020.00027.

[89] Derib Ado. "Amharic Morph Order and Concatenation Rule". unpublished. 2021.

[90] Pavel Rychlý and Vít Suchomel. "Annotated Amharic Corpora". In: vol. 9924. Sept. 2016, pp. 295–302. ISBN: 978-3-319-45509-9. DOI: 10.1007/978-3-319-45510-5_34.

[91] Thomas Lavergne, Olivier Cappé, and François Yvon. “Practical Very Large Scale CRFs”. In: *Proceedings the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*. Uppsala, Sweden: Association for Computational Linguistics, 2010, pp. 504–513. URL: http://www.aclweb.org/anthology/P10-1052.

[92] Sami Virpioja, Ville Turunen, Sebastian Spiegler, Oskar Kohonen, and Mikko Kurimo. “Empirical Comparison of Evaluation Methods for Unsupervised Learning of Morphology”. In: *Traitement Automatique des Langues* 52 (Jan. 2011), pp. 45–90.

[93] *Keras: The Python Deep Learning library*. https://keras.io/. Accessed: 2020-01-18.

[94] *Tensorflow: An end-to-end open source machine learning platform*. https://www.tensorflow.org/. Accessed: 2020-01-18.

[95] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[96] Justin Mott, Ann Bies, Stephanie Strassel, Jordan Kodner, Caitlin Richter, Hongzhi Xu, and Mitchell Marcus. “Morphological Segmentation for Low Resource Languages”. English. In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, May 2020, pp. 3996–4002. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.493.

Annexes

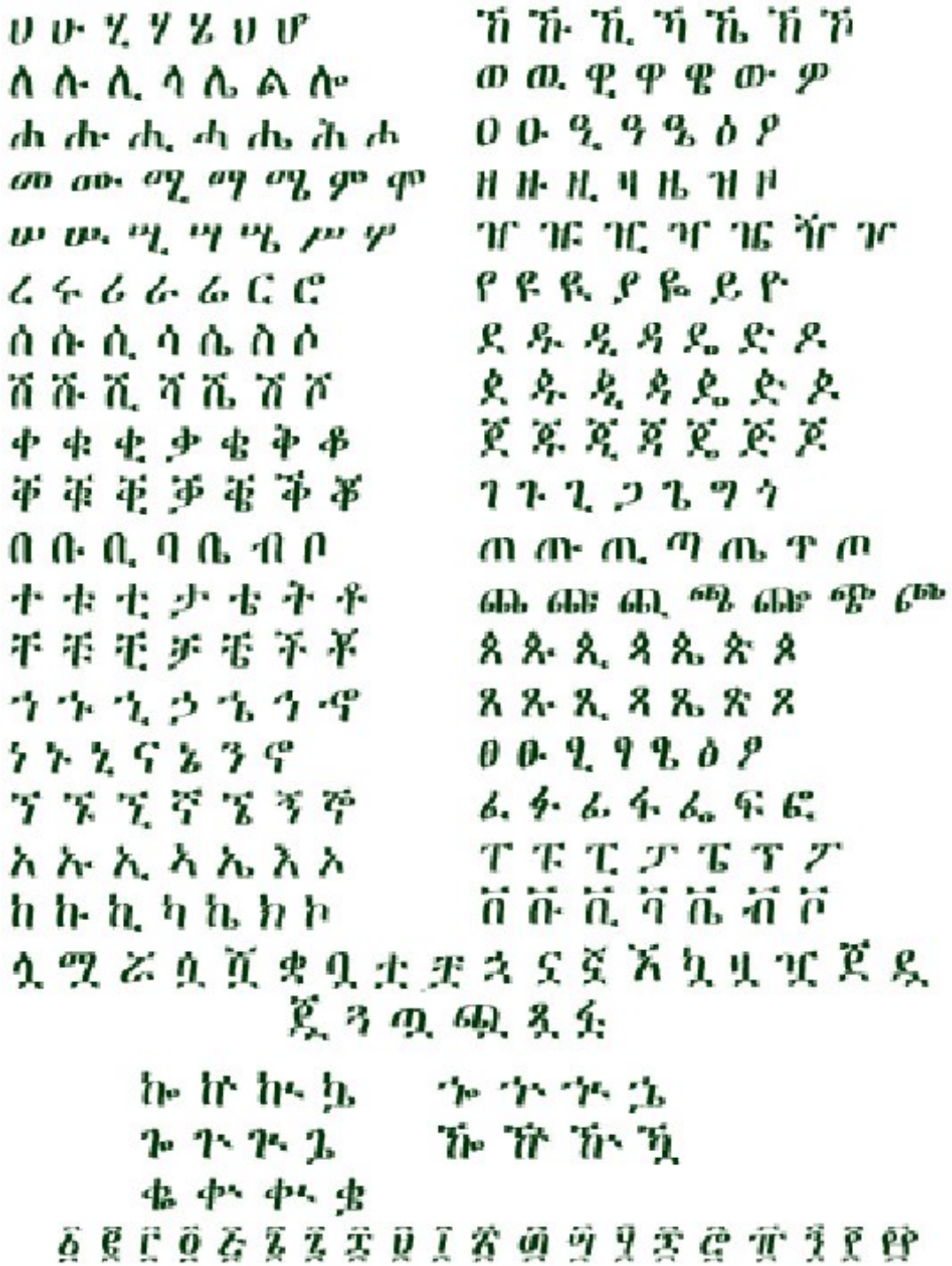

FIGURE ANNEX .1: The Abugida writing system of Amharic

ሀ hv	ሚ mua	ሾ Jo	ተ tv	ነ ni	ኰ kU	ዝ z	ጀ je	ጨ Co	ፍ f	፲ 3	
ሁ hu	ሠ Sv	ሿ Jua	ቱ tu	ና na	ኩ kI	ዙ zo	ጅ j	ጫ Cua	ፎ fo	፳ 4	
ሂ hi	ሡ Su	ቀ qv	ቲ ti	ኔ ne	ኪ kE	ዟ zua	ጆ jo	ፉ Pv	ፏ fua	፴ 5	
ሃ ha	ሢ Si	ቁ qu	ታ ta	ን n	ኸ Kv	ዠ Zv	ጇ jua	ፊ Pu	ፐ pv	፵ 6	
ሄ he	ሣ Sa	ቂ qi	ቴ te	ኖ no	ኹ Ku	ዡ Zu	ገ gv	ፋ Pi	ፑ pu	፶ 7	
ህ h	ሤ Se	ቃ qa	ት t	ኗ nua	ኺ Ki	ዢ Zi	ጉ gu	ፌ Pa	ፒ pi	፷ 8	
ሆ ho	ሥ S	ቄ qe	ቶ to	ኘ Nv	ኻ Ka	ዣ Za	ጊ gi	ፍ Pe	ፓ pa	፸ 9	
ለ lv	ሦ So	ቅ q	ቷ tua	ኙ Nu	ኼ Ke	ዤ Ze	ጋ ga	ፎ P	ፔ pe	፺ 10	
ሉ lu	ረ rv	ቆ qo	ቸ cv	ኚ Ni	ኽ K	ዥ Z	ጌ ge	ፆ Po	ፕ p	፹ 20	
ሊ li	ሩ ru	ቈ qua	ቹ cu	ኛ Na	ኾ Ko	ዦ Zo	ግ g	ፏ Pua	ፖ po	፻ 30	
ላ la	ሪ ri	ቌ qO	ቺ ci	ኜ Ne	ዀ Kua	ዧ Zua	ጎ go	ፈ Fv	ቨ Bv	፼ 40	
ሌ le	ራ ra	ቍ qU	ቻ ca	ኝ N	ወ wv	የ yv	ጓ gua	ፉ Fu	ቩ Bu	፽ 50	
ል l	ሬ re	቎ ql	ቼ ce	ኞ No	ዉ wu	ዩ yu	ጐ gO	ፊ Fi	ቪ Bi	፾ 60	
ሎ lo	ር r	቏ qE	ች c	ኟ Nua	ዊ wi	ዪ yi	ጕ gU	ፋ Fa	ቫ Ba	፿ 70	
ሏ lua	ሮ ro	ቐ Qv	ኾ co	አ xv	ዋ wa	ያ ya	ጒ gI	ፌ Fe	ቬ Be	ᎀ 80	
ሐ Hv	ሯ rua	ቒ Qu	ቿ cua	ኡ xu	ዌ we	ዬ ye	ጔ gE	ፍ F	ብ B	ᎁ 90	
ሑ Hu	ሰ sv	ቒ Qi	ኀ Lv	ኢ xi	ው w	ይ y	ጠ Tv	ፉ Fo	ቦ Bo	ᎂ 100	
ሒ Hi	ሱ su	ቓ Qa	ኁ Lu	ኣ xa	ዎ wo	዗ yo	ጡ Tu	ፏ Fua	ቧ Rv	፲፻ 1000	
ሓ Ha	ሲ si	ቔ Qe	ኂ Li	ኤ xe	ዐ Xv	ደ dv	ጢ Ti	ፀ Dv	0 0	፼ 10000	
ሔ He	ሳ sa	ቕ Q	ኃ La	እ x	ዑ Xu	ዱ du	ጣ Ta	ፁ Du	1 1	. .	
ሕ H	ሴ se	ቖ Qo	ኄ Le	ኦ xo	ዒ Xi	ዲ di	ጤ Te	ፂ Di	2 2	- -	
ሖ Ho	ስ s	ቘ Qua	ኅ L	ከ kv	ዓ Xa	ዳ da	ጥ T	ፃ Da	3 3	' x	
ሗ Hua	ሶ so	በ bv	ኆ Lo	ኩ ku	ዔ Xe	ዴ de	ጦ To	ፄ De	4 4		
መ mv	ሷ sua	ቡ bu	ኇ Lua	ኪ ki	ዕ X	ድ d	ጧ Tua	ፅ D	5 5		
ሙ mu	ሸ Jv	ቢ bi	ኈ Lo	ካ ka	ዖ Xo	ዶ do	ጨ Cv	ፆ Do	6 6		
ሚ mi	ሹ Ju	ባ ba	኉ LU	ኬ ke	ዘ zv	ዷ dua	ጩ Cu	ፈ fv	7 7		
ማ ma	ሺ Ji	ቤ be	ኊ Ll	ክ k	ዙ zu	ጀ jv	ጪ Ci	ፉ fu	8 8		
ሜ me	ሻ Ja	ብ b	ኋ LE	ኮ ko	ዚ zi	ጁ ju	ጫ Ca	ፊ fi	9 9		
ም m	ሼ Je	ቦ bo	ኌ nv	ኳ kua	ዛ za	ጂ ji	ጬ Ce	ፋ fa	፩ 1		
ሞ mo	ሽ J	ቧ bua	ኍ nu	ኰ kO	ዜ ze	ጃ ja	ጭ C	ፌ fe	፪ 2		

Figure Annex .2: Transliteration Table.

Declaration

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

Declared by student:

Name:	Terefe Feyisa Mamo
Signature:	_____
Date:	March 9, 2022

Confirmed by advisor:

Name:	Demeke Asres Ayele (PhD)
Signature:	_____
Date:	March 9, 2022