CONTEXT-BASED SPELL CHECKER FOR SIDAAMU AFOO

MASTER'S PROGRAM (MSc)

MEKONEN MOKE WAGARA

ADVISOR

MICHAEL MELESE (PhD)

HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATICS, DEPARTMENT OF COMPUTER SCIENCE

HAWASSA, ETHIOPIA

November, 2022

CONTEXT-BASED SPELL CHECKER FOR SIDAAMU AFOO

MEKONEN MOKE WAGARA

ADVISOR

MICHAEL MELESE (PhD)

THESIS SUBMITTED TO

HAWASSA UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

INSTITUTE OF TECHNOLOGY, SCHOOL OF

GRADUATE STUDIES, HAWASSA UNIVERSITY,

HAWASSA, ETHIOPIA

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE

DEGREE OF

MASTERS OF SCIENCE IN COMPUTER SCIENCE

November, 2022

Declaration

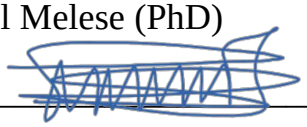
I hereby declare that this MSc Specialty or equivalent thesis is my original work and has not been presented for a degree in any other university, and all sources of material used for this thesis/dissertation have been duly acknowledged.

Name: - Mekonen Moke Wagara

Signature: _____

This MSc Specialty or equivalent thesis has been submitted for examination with my approval as thesis advisor.

Name: - Michael Melese (PhD)

Signature: 

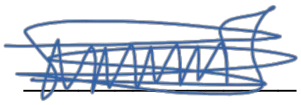
Place and Date of Submission: December 22, 2022

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY

ADVISORS' APPROVAL SHEET

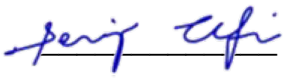
This is to certify that the thesis entitled "*Context-based spell checker for Sidaamu Afoo*" was submitted in partial fulfillment of the requirements for the degree of Master's Degree with specialization in COMPUTER SCIENCE for, the Graduate Program of the Department COMPUTER SCIENCE, and has been carried out by **Mekonen Moke** (ID NO. GPCoScR/0011/12), is conducted under my supervision. Therefore I recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the Department.

Michael Melese (PhD)		December 22, 2022
Name of major advisor	Signature	Date
Andargachew Mekonnen (PhD Candidate)	_____	_____
Name of co-advisor	Signature	Date

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINERS' APPROVAL SHEET

We, the undersigned, members of the Board of Examiners of the final open defense by **Mekonen Moke Wagara** have read and evaluated his/her thesis entitled "*Context-based spell checker for Sidaamu Afoo*", and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the degree.

Michael Melese (PhD)	_____	_____
Name of Major Advisor	Signature	Date
Andargachew Mekonnen (PhD candidate)	_____	_____
Name of Co- Advisor	Signature	Date
Tesfaye Bayu (PhD)	_____	_____
Name of Internal Examiner - I	Signature	Date
Girmay Tekle	_____	_____
Name of Internal Examiner - II	Signature	Date
Dereje Teferi (PhD)		Dec 22, 2022
Name of External Examiner	Signature	Date
_____	_____	_____
SGS Approval	Signature	Date

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

Stamp of SGS Date: _____

Acknowledgement

First and foremost, praise and thanks to the Almighty God, who comforts me when I get worried, teaches me when I get confused, and strengthens me when I get weak throughout my research work and helps me to complete my research successfully.

Next, I would like to express my sincere gratitude to my advisor, Dr. Michael Melese, for his invaluable guidance from title modification to research completion. I would like to thank him for his friendship, sincerity, and motivation. Also, I would like to thank my co-advisor, Andargachew Mekonnen, for his deep comments and encouragement to finish my research and for providing the necessary data resources for my work.

I am extremely grateful to my parents for their love, prayers, caring and sacrifices for educating and preparing me for my future. I am especially thankful to my mom, Bekelech Bada, for her love, understanding, prayers, and continues support in any condition. Also, I want to express my thanks to my little brother, Getu Moke, for his service during my research work.

Finally, my special thanks goes to my friends Kebede Solomon, Andu Alemayehu, Kitaw Ayele, Tariku Workineh, and others whom I did not mention for their direct and indirect support.

Abstract

A spell checker is one of the applications of natural language processing that is used to detect and correct spelling errors in written text. Spelling errors that occur in the written text can be non-word errors or real-word errors. A non-word error is a misspelled word that is not found in the language and has no meaning whereas a real-word error, that is, the word is a valid word in the language but it does not fit contextually in the sentence. We designed and implemented a spell checker for Sidaamu Afoo that can detect and correct both non-word and real-word errors. Sidaamu Afoo is one of the languages spoken in the Sidaama region in the south-central part of Ethiopia. It is an official working language and is used as a medium of instruction in primary schools of the Sidaama national regional state in Ethiopia. To address the issue of spelling errors in the Sidaamu Afoo text, a spell checker is required.

In this study, the dictionary look-up approach with a hashing algorithm is used to detect non-word errors, and the character-based encoder-decoder model is used to correct the non-word errors. The LSTM model with attention mechanism and edit distance is used to detect and correct the context-based spelling error. To conduct the experiment, 55440 sentences were used, of which 90% were for training (i.e., 49,896) and 10% were for testing (i.e., 5544). According to the experimental results, for an isolated spell checker, dictionary lookup with hashing achieved an accuracy of 93.05%, a recall of correct words of 91.51%, and a precision of incorrect words of 72.37% for detection. The encoder-decoder model achieved a recall of 91.76% for corrections. For a context-sensitive spell checker, the LSTM model with attention and edit distance achieved an accuracy of 88.8%, recall of the correct word of 86.18%, and precision of the incorrect word of 62.84% for detection. It achieved a recall of 74.28% for the correction. The results of the experiment show that the model used to detect and correct both non-word and real-word spelling errors in Sidaamu Afoo's written text performed well. Finally, to improve the performance of the model, we recommend using additional data set and a state-of-the-art transformer model.

Keywords— *Spell checker, isolated word spell checker, context-based spell checker, encoder-decoder model, LSTM, Sidaamu Afoo*

Contents

List of Tables

List of Figures

List of abbreviations

AI Artificial Intelligence.

ANN Artificial Neural Network.

BPTT Backpropagate Through Time.

CNN Convolutional Neural Network.

CPU Central Processing Unit.

CV Consonant Vowel.

DL Deep Learning.

DNN Deep Neural Network.

FCBH Faith comes by hearing.

FDRE Federal Democratic Republic of Ethiopia.

FSA Finite State Automata.

GLU Gated Linear Unit.

GPU Graphical Processing Unit.

GRU Gated Recurrent Unit.

IBM International Business Machines Information Interchange.

LM Language Modeling.

LSTM Long Short Term Memory.

MLP Multilayer Perceptron.

MS Micro soft.

NL Natural Language.

NLM Neural Language Modeling.

NLP Natural language processing.

NMT Neural Machine Translation.

OCR Optical Character Recognition.

RLM Recurrent Language Model.

RNN Recurrent Neural Network.

SNRS Sidaama National Regional State.

SOV Subject-Object-Verb.

SVO Subject-Verb-Object.

Chapter One

Introduction

1.1 Background of the study

Language is one of the fundamental aspects of human behavior and is a crucial component of our lives [1]. Human being use language as a means of communication. People communicate in many different ways:- such as spoken form and written forms. The spoken form is our main tool for organizing daily activities with other people when communicating. The written form acts as a permanent record of information passed from one generation to the next generation. Currently, where access to computers and the internet grow more and more widespread, the written form continues to be one of the most popular formats for publishing information and knowledge, either in an electronic or non-electronic form [2]. Huge amounts of text are created every day and the variety of writing styles, formatting, structures, and publishing types are almost unlimited. Due to the fact that language is normally written by humans, it might be the case that different kinds of spelling errors occur within those written texts. The human brain is quite good and has the ability to auto-correct spelling errors. For instance, assume when someone texting or typing on a phone or computer, typing *"cna"* instead of *"can"* and the brain usually fixes it without having to do anything [1]. In the study [3], Sakaguchi et al. realize that if only the beginning and last characters are correct, humans can read words if internal characters are transposed. For instance, *Cmabrigde Uinervtisy - Cambridge University*. However, this is difficult for computers if it is not trained with a vast number of data sets. The writer, on the other hand, makes spelling mistakes, either unintentionally or due to a lack of orthographic knowledge of the language. Today, natural language processing (NLP) is concerned with giving computers the ability to under-

[1] Does Your Brain AutoCorrect?

stand natural language in much the same way humans can [4]. Having such types of systems enables one to communicate with machines as though one is communicating with a human. Spell checker is one of the most essential NLP applications related to text processing [5]. It is the application of NLP that detects and corrects spelling errors in the text. It can be either a stand-alone or embedded with other text editing applications, such as a word processor, search engines, etc. In broadly, spelling errors are divided into two types: non-word and real-word errors [6, 7]. Non-word errors are spelling errors that result in non-existent words in the vocabulary, whereas real-word errors result in an existing word in the vocabulary that is not correct in the context of words. Of these, a non-word error has been widely studied and algorithms to detect and suggest the correct word for the error have been proposed. An isolated-word spell checker is used to handle non-word spelling errors, whereas a context-based spell checker is used to handle both non-word and real-word errors. Identifying real-word errors is more difficult than identifying non-word errors because a real-word error requires the whole sentence to capture the context of the word [8, 9, 10]. For instance, the sentences *the son rises in the east* are written instead of the *The sun rises at the east*. Here, the words *son* and *sun* are correct words in the English language. An isolated-word spell checker cannot identify these kinds of errors.

Spell checker performs two main tasks such as spelling detection and spelling correction. Spelling detection identifies whether the input word is correct or misspelled in the language, whereas spelling correction auto-corrects or suggests one or more alternative correct words for a misspelled word in ranked order [11]. According to Kukich [7], dictionary look-up and n-gram analysis are two methods for spelling detection for isolated word spell corrector. The dictionary lookup method is the most popular way for error detection. It compares the input word against the dictionary. However, as human languages are complex and contain countless words and terms, as well as domain-specific idioms, proper names, technical terminologies, and special jargon, regular dictionaries are insufficient to cover all words in the vocabulary of the language [12]. The isolated error correction phase of the spell checker is contain suggesting the candidates and ranking them in an appropriate order. Suggesting candidates have listed the alternative word for the misspelled words from the dictionary by calculating the similarity between the misspelled words with the dictionary words. Several algorithms for finding candidate corrections have been explored [13]. The minimum edit distance, similarity key techniques, rule-based techniques, n-gram-based techniques, and

probabilistic techniques and neural networks. The most popular method for a year is computing the minimum edit distance between the detected word and a dictionary entry. The minimum edit distance has been defined as the minimum number of editing operations (i.e. insertions, deletions, and substitutions) that are required for transforming one string into another. Lastly, rank the candidates which filter and reorder a large number of possibilities so that the right correction appears at the top or very near the top of the list. Choosing the correct word from candidates is done by either human or by computer auto-correcting.

Currently, the neural network approach has become the state of the art for different NLP tasks. Mainly, a recurrent neural network for sequential data has achieved good results in various NLP tasks, such as machine translation, speech recognition, name entity recognition, etc [14]. Ghosh and Kristensson [15] are the first among the researchers who used the neural network model for error correction. They proposed a correction model for English language text and achieved good results for spelling correction using the encoder-decoder model. Currently, It has been applied to spelling correction for different languages using different neural network approaches. The Bangla [16], Turkish [17], Hindi [5], Chinese [18, 19, 20], Indonesian [21], Azerbaijani [22], Arabic [23, 24, 25] are some languages which conducted the spelling correction using deep learning approach. In this study, we used the encoder-decoder model to correct the non-word spelling error and the neural language model to correct the contextual spelling error in the Sidaamu Afoo text.

Context-based spell-checking for misspelled words is a new approach that solves both non-word and real-word. Context-based error detection and correction help to improve the performance of spell checkers, since it focuses not only on non-word errors but also on real-word errors [26]. Research into context-based spell checking is going on to develop a spell checker and corrector based on the context of the text rather than detecting and correcting a spelling error of a word without taking into account the context of the word [27]. Spelling error detection and correction now focus on the development of spell-checking algorithms that make use of context. Nowadays many applications were developed contextually to correct spelling errors. For instance, Microsoft word 2007 provided context-based spellchecker to solve both non-word and real-word errors during typing on the Microsoft word interface [26]. Recent years have seen the advance of context-aware spell checkers such as Google Suggest, offering reasonable corrections of search queries using *Did you mean?* [28]. For a long period of time, the n-gram language model has been state-of-art to correct context-sensitive spell

checkers. These spell checks typically employ noisy-channel or winnow-based techniques. However, the n_gram has its own limitation when predicting the word. These are the sparsity of the word, limitation of semantics, long-term dependencies, and storage. The elevate the problem of the n_gram language model, the recurrent neural network has come as state-of-art to different sequence-based data. Recurrent neural networks (RNNs) are capable of learning features and long term dependencies from sequential and time-series data [29]. However, the techniques used for training RNNs might significantly restrict the memory generated from the recurrent connections. As a result, the network has not been able to learn long-term sequential dependencies in the data for any of the models up to this point due to exploding or vanishing gradients throughout the training phase. It has also the vanishing gradient problem which is it cannot capture the long-term relationship of the word in the sentences. To tackle the problem, the most commonly designed technique used for different studies is the long short memory (LSTM) [30]. LSTM used different memory cells and their input and output are controlled by gates. These gates control the flow of the information to hidden neurons and preserve extracted features from previous timestamps [30, 31].

The aim of this research is to develop an isolated and context-based spell checker for Sidaamu Afoo. Sidaamu Afoo is one of the spoken and working languages in Sidaama national regional state in Ethiopia. The language is spoken in southern Ethiopia by the Sidama people. Next to Afan Oromo and Somali, Sidaamu afoo is the biggest language under the Cushitic category [32]. It uses the Latin script for the writing purpose. Sidaamu Afoo has around five million native speakers settled in the south-central part of Ethiopia [2].

Generally, the aim of the research is to develop an isolated and context-based spell checker for the Sidaamu Afoo writing system. In the Sidaamu Afoo writing system, there is no spellchecker used to correct misspelled words yet. Obviously, misspelled words hinder the idea from the author's world and the reader's world. To transfer knowledge correct spelling has a great role in order to facilitate knowledge sharing. The spell checker fills the gaps of the spelling hindrance between the author's and the reader's world. An isolated and context-based spell checker for Sidaamu Afoo specifically needed to correct the misspelled words in the writing system.

[2] Link: https://joshuaproject.net/languages/sid

1.2 Motivation of the study

The first reason that motivates us to conduct research on spell checkers for Sidaamu Afoo is that there has been no spell checker developed for Sidaamu Afoo. However, a huge amount of electronic data is processed every day since the language is used in schools, offices, and other media. When processing electronic data, a spelling error has the potential to change the writer's message. Spell checkers can detect and correct spelling errors that occur during document processing. Due to the absence of a spell checker for Sidaamu Afoo, the texts kept containing spelling errors, and that has ability to hinder the message of the writer. The researcher is one of the community members of this language speaker and needs to contribute something to the language to make it more valuable in his discipline. The second reason that motivates us to conduct this study is that the spell checker paves the way for anyone interested in working on the NLP application for Sidaaamu Afoo. NLP applications like grammar checkers, machine translation, text-to-speech synthesis, information retrieval, dialogue systems, question-answering, etc. can incorporate spell checkers to improve performance.

1.3 Statement of the problem

The writing system has a great role in sharing information between the authors and the reader. Poor spelling (misspelled words) has the ability to hinder communication between authors and readers [33, 34]. As a result, the authors did not transfer their knowledge clearly to the readers. Sidaamu Afoo is the working language of the Sidaama national regional state. In addition, Sidaamu Afoo is used as a medium of instruction from grade one to grade four, as a subject course from grade one to grade twelve in Sidaama national regional state, and given as a department at Hawassa University. However, seeing the spelling error is common in different Sidaamu Afoo digital texts. To confirm this, we conducted a sample survey on the Sidaamu Afoo text. Accordingly, we have selected three famous personal Facebook pages that have many followers, and the top 50 comments were collected from each post. We analyzed the comments and found that 67 (55.3%) out of 150 users do not spell correctly. Furthermore, we collect and analyze banners, letters, cards, etc., and spelling error appears even in publicly available banners. Some of them are depicted in Appendix B.4. This implies that an efficient and accurate spell checker is needed for Sidaamu Afoo.

In the Sidaamu Afoo writing system, consonants can be geminate or simplex, and vowels can be short or long in the word. Sometimes the short and long vowels, as well as simplex and geminated consonants in the word, cause real-word spelling errors [32, 35]. For instance ***/qara/*** meaning in English *'main'* and ***/qarra/*** meaning in English *'challenge'* or ***/gowa/*** meaning in English **'sewing'** ***/gowwa/*** meaning *'fool'* and ***/goowa/*** meaning in English *'neck'* has completely different meaning. In sentences, those words can change the message completely to another meaning. For instance,

1. ***Sidaamu dagoomu qoqqowi qaru quchumi Hawaasaati.*** *-'The capital city of Sidaama National Regional State is Hawaasa.'*

2. ***Sidaamu dagoomu qoqqowi qarru quchumi hawaasaati.*** *- 'The challenge city of Sidaama National Regional State is Hawaasa.'.*

The message of the first and second sentences is completely different due to the distorted word ***qarru****-'challenge'*. Usually, such errors distort the syntax and semantics of the whole sentence, which requires human beings to detect them. Even for human beings, non-native Sidaamu Afoo writers cannot correct those spelling errors correctly since they are not familiar with the languages and do not know the semantics of the sentences. This implies that an efficient and accurate context-based spell checker is needed for Sidaamu Afoo.

The research to develop efficient spell-checkers for different languages is going on since the 1960s [36]. In this regard, many spell-checkers have been developed for some local languages like Amharic [37, 38], Afaan Oromo [39, 40], Tigrigna [41] and other foreign languages like English [7, 10, 3], French [42], Bangla [43], Azerbaijani [22], Turkish [17], but research for the regional languages is still in its infancy stage. This is because there are many regional languages and each one of them is under-resourced and lacks well-prepared computational resources. Sidaamu Afoo is one of the under-resourced languages and there has been no spell checker tried yet. This research proposes a context-independent or isolated spell checker that detects and corrects non-word spelling errors without considering the context of the word and a context-based spell checker that detects and corrects both non-word and real-word spelling errors by considering the context of the word for Sidaamu Afoo. As a result, developing a Sidaamu Afoo spell checker can help language users and learners, even native Sidaamu Afoo writers, correct misspellings without spending time and effort doing so during writing.

To this end, the following research questions are explored and answered in this study.

- Which approach is better for the isolated word and context-based spell checker for Sidaamu Afoo words?
- To what extent spelling errors in Sidaamu Afoo can be detected and corrected for isolated word and context-based spell checkers?

1.4 Objectives of the study

1.4.1 General objectives

The general objective of the study is to design and implement an isolated and context-based spell checker for Sidaamu Afoo.

1.4.2 Specific objective

To achieve the general objective, the following specific objectives is performed: -

- To review related literature to understand state-of-the-art of deep learning and spell checkers.
- To collect and prepare Sidaamu Afoo corpus.
- To design an architecture of spell checker model.
- To develop a model that detects the misspelled word and suggests a closer correction.
- To evaluate the performance of the model with test data.
- To draw a useful conclusion and forward recommendations for further work.

1.5 Scope and limitation of the study

The scope of the study is to design and implement a spell checker model for Sidaamu Afoo which can detect and correct both non-word errors without considering the context of the misspelled word and contextual error(both non-word and real-word) by taking the context

of the word. The study attempts to solve spelling errors as a result of typographical errors, which occur due to the deletion, insertion, substitution, or transposition of characters. The numbers, punctuation, and compound words are not considered.

The main limitation of the computational language application for low-resourced languages is the lack of enough data for its application. Sidaamu Afoo is also one of the low-resource languages that have limited resources for computational language applications. The limitation of the proposed model is it does not correct more than one distance error between the misspelled word and the correct word. Also, it does not correct compound words, numbers, or punctuation except the single (‘) and double (”) apostrophes. Those punctuation are used as glottal sounds in the word.

1.6 Methodology of the study

To achieve the specific objectives, the researcher would use different methods and techniques that are related to a spelling error detector and corrector. We use the following methods:

1.6.1 Literature Review

To get a deeper understanding of the state-of-the-art approaches for spell-checker we reviewed different works of literature.

1.6.2 Research design

To study the research, we follow an experimental research design. Experimental research is conducted using dependent and independent variables with a scientific approach [44]. The reason that we select the experimental research design is to investigate the possible cause and effect by observing the effect of some variables on other variables easily. The main steps followed in conducting experimental research include data set preparation, implementation, and evaluation.

1.6.3 Data collection and preparation

Sidaamu Afoo does not have publicly available corpus text for spell checkers. However, for training and testing purpose Sidaamu afoo corpus were important. In addition, to mea-

sure the performance of the spell checker during detecting and correcting both non-word and real-word errors the corpus is necessary. So, We collected the corpus from publicly available websites like Faith-comes-by-hearing (FCBH) and Ethio press agency. The Sidaamu Afoo Holly Bible and Bakkalcho newspaper crawled from the FBCH and Ethio press agency websites respectively. In addition, the Sidaama National Regional State constitution is also used for corpus.

Text preprocessing is an essential task for corpus preparation and to clear unnecessary errors. Since the corpus was collected from various sources, it can contain spelling errors. We manually cleared the corpus from any kind of unnecessary errors that were made valid to represent Sidaamu afoo vocabulary. The cleaned corpora were used for training and testing purposes for the spell-checker model.

1.6.4 Evaluation

The performance of the system is measured using evaluation metrics. Three frequently used measures for evaluating a spell-checking application are precision, recall and accuracy. In general, precision denotes a system's exactness, recall indicates a system's coverage and accuracy indicates the overall performance of the spell checker.

1.7 Significant of the study

This study has significance for the Sidaamu Afoo users to minimize information gaps and seamlessly transfer their knowledge and information from one to another. Because the spell checker in Sidaamu Afoo detects and provides suggestions for misspelled words, it facilitates good communication between the writer and readers.

The output of this research can be used as inputs for other Sidaamu Afoo NLP applications like grammar checkers, search engines, text summarizing, sentiment analysis, machine translation, and information retrieval, which require proper spell checkers [45]. For instance, spell checks are important in search engines since user-generated text frequently contains spelling errors. Many websites provide a function that automatically suggests the right answers to users' misspelled questions in the form of *"Did you mean?"* suggestions or automated corrections. Providing suggestions makes it convenient for users to accept a proposed correction

without retyping or correcting the query manually. Also, machine translation requires a powerful spell checker to translate from one language to another. So, this study contributes a lot to researchers who are interested in working in the field of natural language processing for Sidaamu Afoo.

1.8 Organization of the study

The overall research work is organized into six chapters. The first chapter describes the introduction of the study, which includes the background, motivation, objective, methodology, scope and limitations and significance of the study. The second chapter is all about literature reviews and related work. It introduces the concepts of spell checking, the approaches to detecting and correcting both isolated and context-sensitive spell checkers, the details of the state-of-the-art deep learning approach, the evaluation metrics of the spell checker, and related works. In the third chapter, the characteristics of the Sidaamu Afoo writing system that are related to the research area are discussed.

Chapter four describes the general architecture and design of the Sidaamu Afoo spell-checker model. It explains in detail the components of the designed architecture and the methods of detecting and correcting Sidaamu Afoo spelling errors. The fifth chapter presents the experimental results and evaluation of the spell-checking model. Finally, a conclusion and future work are presented in chapter six.

Chapter Two

Literature review

2.1 Overview

This chapter discusses the approaches and techniques for detecting and correcting spelling errors. Literature reviews are necessary to analyze and understand the previous researches that have been done in spelling checking and correcting techniques. The literature can be explained different aspects of information on spell correction techniques and algorithms from various resources that have been done before the current spelling checker. In section 2.2, we discussed types of spelling errors that can occur during processing a text, the two common error types, and the pattern of spelling errors. Next Section 2.3 and 2.4 discuss techniques that are applied to spell checkers for spell error detection and correction respectively. In addition, the background of the state-of-art that we used is discussed. The evaluation method of a spell checker is discussed in section 2.5. In Section 2.6 the related work on error detection, correction, and the methods of spelling error generation is discussed. Lastly, section 2.7 summarized the related works and informs the techniques used in this study.

2.2 Spell checker

In the current research, spelling errors are defined as human-generated writing errors. The term spelling error sometimes refers to both spelling errors and typing errors. According to Kukich [7], spelling error is classified into non-word and real-word spelling errors. A non-word error is defined as a given word that is not found in a dictionary or does not give any meaning to the language [46, 47, 6]. Real-word error is defined as a given word that is valid and meaningful in language but does not contextually fit in the sentence, thus making the

sentence syntactically or semantically ill-formed or incorrect. The goal of the spell checker was to correct those kinds of errors in the text.

A spell checker is a software that finds and fixes misspelled words in written text [7]. It recognizes acceptable words in some languages and typos in the language. It may be stand-alone or integrated with other applications, such as a search engine, email client, electronic dictionary, or word processor. The most common way that spell checkers are used is as a component of a more extensive application, like a word processor, email client, electronic dictionary, or search engine [13]. The history of the spell checker is not new, it dates back 1960s. The first spell checker developed for the mainframe computer that supported six languages for the IBM company only displays the error word in the text. In the 1980s IBM company introduce a new spell checker for the personal computer which is advanced of the first mainframe computer spell checker [40]. Moreover, for personal computers spell checkers were first introduced in 1981 by the IBM Company. By the mid-1980s many popular word processing applications (such as WordStar and WordPerfect) had integrated spell checkers [40].

Fundamentally, it is made up of two components including an error detector and an error corrector. The error detector detects whether the word is correct or not. If the word is misspelled, the spell checker will be prompted to correct the misspelled word in the text. While the error corrector provides candidate suggestions for misspelled words to represent error words with correct words and ranking candidate suggestions. Some spellcheckers use the same technique to provide all two functionalities in one step, while others use different techniques for each functionality. For instance, the rule-based and statistical approaches to spelling correction rely on error detection first before offering correction suggestions. However, many neural approaches to spelling checking normally correct errors directly over an entire input sentence. Presenting an entire sentence to the network or decoder for correction involves the risk of modifying words that are correct in the context and should not be changed.

Spell checking involves non-word error detection and spelling correction involves isolated-word error correction [7]. Isolated-word error correction refers to spell correcting without taking into account any textual or linguistic information in which the misspelling occurs; therefore the corrections are based only on the misspelled word itself. Whereas context-dependent word correction would correct errors involving textual or linguistic context. The

context-based word correction approach is useful for correcting both non-word and real-word errors. Unlike non-Word errors, for real-word errors, detection is a much harder problem compared to correction. Because the error is not an actual spelling error, (e.g. *form - from*, *the - he*) rather it is some sentence disagreement.

Spelling error correction phases are classified as fully automatic spelling error correction systems and interactive spelling error correction systems [12]. In a fully automatic spelling error correction system, the application finds candidate words and chooses the most likely one automatically without user intervention. The interactive spelling error correction system finds candidate words, ranks them, and the most probable words are then suggested to the user and the user can choose the word that was intended.

We conducted experiments in this study for both isolated and context-dependent word correction approaches. The isolated word correction approaches were conducted to detect and correct only non-word spelling errors without considering the context of the input word. For the context-dependent word correction approaches, we tried to use the context of the given word to identify spelling errors and to provide candidate corrections that can detect and correct both non-word and real-word spelling errors. In the isolated word correction approach, first, detect the non-word spelling and then provide the candidate correction for the detected word to the user. whereas, for the context-dependent word correction approach, the suggestion is generated for the input word implicitly. Then candidates are provided for the user if and only if the input word is not found in the generated candidates.

2.2.1 Spelling error pattern

An error can occur due to different factors. One of the most commonly known factors is the human factor, that human does not write correctly when typing. Another is the OCR factor, when OCR systems convert handwritten text to digital text, due to a lack of pattern recognition the spelling error can occur in the text [48]. It substitutes the wrong characters for similar-looking characters (e.g., *c - e, m - nn, o - 0, t - f*, etc). The current research considers only human-generated writing errors. Kuckich [7] generalizes the type of errors to cognitive, phonetic, and typographic errors.

1. **Cognitive error:** Cognitive errors occur when there is a misconception or a lack of knowledge of the correct spelling of a word. For example *beleive* instead of *believe*.

This kind of error happens in Sidaamu Afoo due to the writer's lack of knowledge of Sidaamu Afoo orthography. For instance, *quchchuma* instead of *quchuma*.

2. **Phonetic error**: This is a special kind of cognitive error. It occurs when users frequently attempt to replace a wrong spelling with the most likely one that roughly matches the actual word. In this kind of error, sometimes homophone characters are used instead (e.g., c for k). For example *kat* instead of *cat*. The phonetic error occurs in Sidaamu due to some assimilated letters when reading specific words. This rarely happens in Sidaamu Afoo. For example, *Anbooma* instead of *Ambooma - Hyena*. Here the letter *n* and *m* are assimilated in *b*.

3. **Typographic error:** This kind of spelling error is due to pressing the wrong key when typing or missing the key. The typographical errors fall into one of the four categories.

   (a) **Insertion:** While writing a word, if a character is pressed twice or some other character is pressed along with the original character, the undesired character is considered to be inserted. e.g. *them* instead of *the*.

   (b) **Substitution:** When another character substitute for the desired character in a word, the desired character is changed by the wrong character. e.g. *tee* instead of *the*.

   (c) **Deletion:** While writing a word if any or more characters are missing, then it is considered to be deleted and the error is considered to be a deletion error. e.g. *te* instead of *the*.

   (d) **Transposition:** Transposition mistakes, as the name implies, happen when characters are swapped, or move locations. e.g. *teh* instead of *the*.

These typographic errors can be single errors or multiple errors. A single error is an error created by one character modified in the word. Whereas, multiple errors happen with more than one character modification. According to [46, 36], 80% of the typographical errors are under the single errors that happen in the written text. The typographic error also occurs in Sidaamu Afoo's text as well.

2.3 Spelling error detection approaches

The first action taken in a spell checker is error detection. It is determining whether an input word is misspelled or not in the language. To detect non-word spelling, there are two commonly known techniques to correct spelling errors: dictionary lookup and n-gram analysis [49, 50].

2.3.1 Dictionary lookup

It is a straightforward and the most widely used method. In the dictionary lookup method, the dictionary has two functions [51]. First, a dictionary is used to identify whether an input word is valid or not by comparing and locating the input word in a dictionary. It used exact string matching techniques [46]. If the input word does not exist in a dictionary, it is detected as invalid otherwise it is a valid word. For example, if a user types the word *'teh'* instead *'the'* search the word *'teh'* within a dictionary. The word *'teh'* is incorrectly spelled if it would not be found in a dictionary otherwise the word *'teh'* is correct. Second, If an erroneous word is detected, then the dictionary is used to generate possible correction by calculating the similarity between the dictionary entry and the erroneous word. However, the size of the dictionary can have an impact on the spelling tasks. A large-sized dictionary uses more space and may take a longer time to search. The small-sized dictionary does not contain many correct words in the language. It provides the user with an excessive number of false rejections of valid words [47]. The most significant dictionary lookup techniques are hash tables, binary search trees, and finite-state automata to access large-size dictionaries [12].

1. **Hashing:** Hashing is a well-known and efficient technique to increase dictionary access time. The fundamental benefit of hash tables is their random access, which eliminates the numerous comparisons required to search a dictionary [52]. The basic idea of hashing relies on some efficient computation carried out on an input word to detect where a matching entry can be found [53]. More specifically, hashing is a technique used for searching an input word in a pre-compiled hash table via a key or a hash address associated with the word and retrieving the word stored at that particular address. A hash table uses a hash function to map identifying values, known as keys, to their associated values.

In a pre-built hash table, a word is saved at a certain address, which must be determined to look up an input word in a spell-check program. An input word is misspelled if the word stored at the hash address differs from the input word.

2. **Binary searching algorithm**: Binary search trees, particularly median split searching, have been used for dictionary lookup and, subsequently, for spell checking. The dictionary words are sorted in alphabetical order and split into medians. As a result, to find the input word and identify its spelling, the collections would be separated into left and right windows (LRW), and the middle value calculated [54]. Once the middle value is known and it is not the input word looking for, the algorithm looks for the word in one of the two windows, depending on the given criteria. As a result, only one window would be active at any given time. It continues till it found the input word in the dictionary. If the input word is found in the dictionary, the word is a valid word otherwise it is detected as a misspelled word.

3. **Finite state automata (FSA)**: A finite-state automaton (FSA) is used to represent a language, which is defined to be a set of strings, each string being a sequence of symbols from some alphabet. Finite-state machines consist of four main elements: states which define behavior and may produce actions. State transitions, which are movements from one state to another, rules or conditions that must be met to allow a state to transition. Input events may cause rules to be triggered and state transitions when they are generated either externally or internally. A finite state machine needs to have a starting initial state and a current state that keeps track of the outcome of the previous state transition. Received input events act as triggers, which cause an evaluation of some kind of the rules that govern the transitions from the current state to other states.

2.3.2 N-gram analysis

N-gram is a statistical approach that is to estimating the probability of a word or a sentence. To compute the probability sentence, it is required to compute the probability of a word, given the sequence of words preceding it. The detail in the statistical language model in the equation is described in section 2.4.4. It can be word-based or character-based n-grams. In

spelling error detection literature uses the character n-gram which is a set of n consecutive characters extracted from a word. N-grams are sub-strings of length N. N can be 1,2,3, ... If n = 1, 2, 3, they are referred to as uni-grams, bi-grams, and trig-rams, respectively [55].
Generally in spelling detection, each examined n-gram in the input word is looked up in a precompiled table of n-grams to find its frequency. Strings with non-existent or rare n-grams are identified as potentially incorrect words [51]. Character n-gram techniques require a large lexicon or corpus in order to compile the table of n-gram frequencies. It is mainly used for detecting and correcting errors made by optical character recognition (OCR) devices [13].

2.4 Spelling error correction approaches

Error correction is another task that is performed after the input text is detected to be misspelled. It is a way of finding out the suggested word for a misspelled word from a dictionary or corpus to replace it. The spelling correction method includes searching a dictionary for those words that are detected as erroneous words.
Different approaches are proposed to correct spelling errors. Some of the spell correction methods are edit distance, similarity key, rule-based, n-gram analysis, probabilistic and neural networks [46].

2.4.1 Minimum Edit distance

Minimum edit distance is the minimum editing operation(insertions, deletions, substitutions or transposition) between words [13]. In spell correction, the minimum edit distance computes the distance between the misspelled word and the dictionary entry. After computing, the words with minimum edit distance are chosen for suggestion as the alternative correction. It has different algorithms like the Levenshtein algorithm, Hamming, and Longest Common Sub-sequence.
The Levenshtein algorithm is widely used in spelling correction tasks to measure string similarity between misspelled words and dictionary entries [48, 6]. Levenshtein distance is a mathematical measure for calculating the difference between two strings. Mathematically, the Levenshtein distance between two strings a, b (of length $|a|$ and $|b|$ respectively) is given by $lev_{a,b}(|a|, |b|)$ equation below. Levenshtein distance algorithm is an algorithm editor that

utilizes a dynamic programming string for operation [56]. For example, the Levenshtein edit distance between *"dog"* and *"cat"* is three (substituting *d* by *c*, *o* by *a*, *g* by *t*). The Hamming algorithm measures the distance between two strings of equal length. For example, the hamming distance between *"then"* and *"them"* is 1 (changing *n* to *o*). The Longest Common Subsequence algorithm is a popular technique to find out the difference between two words. The longest common subsequence of two strings is the mutual subsequence.

$$lev_{a,b}(i,j) = \begin{cases} max(i,j), & \text{if } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1 \end{cases} & , \quad \text{otherwise} \end{cases}$$

2.4.2 Similarity key techniques

The basic concept of similarity key techniques is the mapping of every word into a key such that similarly spelled words will have similar keys [46]. The suggested word is chosen based on a closely matching key. This method can manage keyboard errors and is fast because only words with comparable key combinations need to be processed. It is not essential in this case to compare each word in the dictionary directly to the misspelled word. As a result, dictionary entries with similar spellings will be referenced by computing keys for misspelled words. A very early, often cited similarity key techniques are the SOUNDEX system and The SPEEDCOP System [13].

The SOUNDEX system was devised to solve the problem of phonetic errors by partitioning the set of letters into seven disjoint sets, assuming that the letters in the same set have a similar sound [57]. Each of these sets is given a unique key, except for the set containing the vowels and the letters h, w, and y, which is considered to be silent and is not considered during encoding. For instance, the word *book* will be transformed to *B002* which will become *B2* and the word bush will be transformed to *B020* which will also become *B2*. Although book *an*d bush are neither equivalent nor similar, they are mapped to the same key, *B2* [46]. SPEEDCOP system finds the similarity key by rearranging the letter of the words.

Table 2.1: Defined rule of the Soundex algorithm

Code	Letters
0	A, E, I, O, U, H, W, Y
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

2.4.3 Rule-based approach

Rule-based techniques involve algorithms that attempt to represent knowledge of common spelling error patterns, for transforming misspelled words into correct ones. The knowledge is presented as rules. The candidate suggestions are generated by applying all applicable rules to a misspelled word and retaining every valid word in the dictionary that results [46, 12, 48].

2.4.4 N-gram technique

An n-gram language model is widely used in natural language processing and state-of-art for a long year. Numerous NLP applications, including machine translation [58], information retrieval [59], document classification [60], spelling correction [36] has been used. It is a special kind of statistical language model approach.

A language model can assign a probability to an entire sequence of words, such as a sentence. To illustrate the concept, [61] gave two example sentences and stated that a language model could estimate that sentence (1) has a higher probability to be in a text than a sentence (2) [62].

(1) *All of a sudden I notice three guys standing on the sidewalk*

(2) *On guys all I of notice sidewalk three a sudden standing the*

A statistical model of language can be represented by the conditional probability of the next word given all the previous ones. It is to estimate the likelihood (probability) of a word or

a sentence [63, 64]. Any form of language processor might produce such an estimate. to make an estimate based on a set of rules during early history. However, it is handwritten, so it is impossible to cover all grammatically correct rules. In the 1980s, the statistical language model was introduced and showed good performance.

Statistical Language Models (SLM) use training data to calculate statistical estimations, in contrast to rule-based methods [65]. It essentially relies on the frequency of n consecutive words occurring in the training data, to give a probability to the next word [66, 10]. Estimation of probability is done by decomposing sentence probability into a product of conditional probability using chain rule as follows:

$$P(W) = P(w_1, w_2..., w_N) = \prod_{i=1}^{N} P(w_i|w_1, w_2, ..., w_i - 1) \tag{2.1}$$

Typically, equation 2.1 is used for different length sequences. For larger i values, it is difficult to estimate it from a corpus since it requires multiple occurrences of any $(w_1...w_i)$. Therefore, limiting the previous context is necessary. This technique is called N-gram, and it is one of the most widely used SLM methods. N-grams are sequences of n-words that form a sentence. N-gram of preceding words to n results in equation2.2.

$$P(W) = P(w_1, ..., p(w_N) \approx \prod_{i=1}^{N} P(w_i|w_1w_2...w_i - 1) \tag{2.2}$$

Their performance is highly dependent on both the quantity and quality of the training data. The last decades have made it possible to access larger amounts of text. Therefore, state-of-art results were improved dramatically.

Simplest N-gram is the unigram. In the unigram model, the probability of a word (w_i) only depends on the probability of (w_i) being in the dataset. It does not take the preceding context into the calculation. For example, In a trigram model, Language Models compute the probability of a sentence, as shown in equation 2.3

$$P(W) = P(w_1, ..., p(w_N) \approx \prod_{i=1}^{N} P(w_i|w_2w_1) \tag{2.3}$$

Maximum likelihood estimation (MLE) is a commonly used technique to compute N-gram probabilities. Considering the trigram example, the probability of word w_i can be calculated

by counting how many times w_i appeared with w_{i-2}, w_{i-1} and normalizing by all occurrences of w_{i-2}, w_{i-1} as shown in equation 2.4

$$P(w_i|w_{i-2}w_{i-1}) = \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})} \tag{2.4}$$

Considering how many word combinations can be made using natural language, there is a great possibility that an n-gram never occurred in the dataset[67]. As a result, the probability is assigned to 0, which will prevent the model to predict any words. To solve this problem, different smoothing extensions are introduced. Alternatives such as interpolation and back-off are described and evaluated in [66].

However, n-gram language models have a sparsity problem, in which we do not observe enough data in a corpus to model language accurately. In addition, there is no semantic relationship between the words [63].

2.4.5 Neural Network techniques

Neural networks have emerged in recent years as an alternative to estimating and storing categorical n-gram features language models [67]. This approach solves the data sparsity problem by representing words as vectors (word embeddings) and using them as inputs to a neural language model [68]. It can be considered a state-of-art method for different NLP applications. The concept of the neural network is applied in the field of natural language processing to achieve better results than the statistical and rule-based methods. We used this method to correct the spelling error of the Sidaamu afoo text. so, we described it in detail as follows.

A neural network is an artificial intelligence technique that is a network of interconnected nodes with input and output layers as well as one or more hidden layers that are modeled after simplified versions of brain neurons. Every node is associated with a weight for a neuron. Applying an activation function to the weighted neurons will produce the output. The diagram representation of a simple neural network is Figure 2.1. The mathematical representation of the neural network in equation 2.5

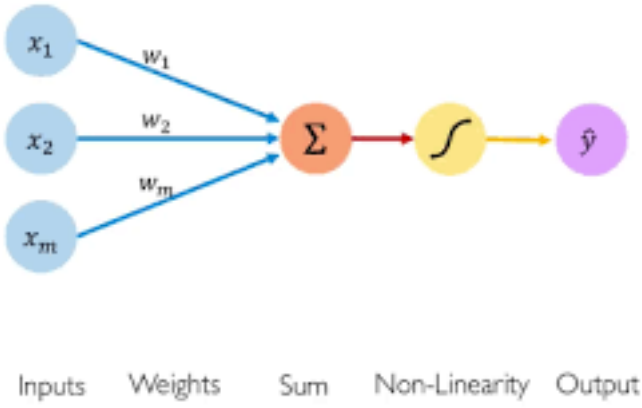


Figure 2.1: Simple Neural network
Source: [69]

$$\widehat{y} = F(\sum_{i=1}^{m} x_1.w_i + w_0.b) \tag{2.5}$$

In equation 2.5 is for the simple neural network with no hidden layers. An artificial neural network can either be shallow or deep. When an ANN has more than one hidden layer in its architecture, they are called a Deep Neural Network, otherwise, it is a shallow or Feed Forward Neural Network.

A deep neural network contains more than one hidden layer, in addition to input and output layers. It can handle non-linearity, which is one of the challenges of the n-gram language [67]. The structure of the deep neural network is inspired by the structure of the human brain. Based on the input received, the neurons transmit the signal to other neurons. It depends on the input signal whether it passes the output or is ignored. Each neuron in a layer has a function known as the activation function. They act as a gateway, transmitting the signal to the next connected neuron. The weight affects the input, the next neuron's output, and finally the final output layer. The weights are initially assigned at random, but when the network is iteratively trained, the weights are optimized to ensure that the network makes the right prediction. Figure 2.2 shows a simple deep learning structure. There are three types of Neural Networks. Those are Artificial Neural network(ANN), Convolutional Neural Networks(CNN) and Recurrent Neural Networks(RNN).

All other neural networks are an extended version of these three neural networks. As mentioned above, NNs are used for language modeling and in this research addresses the issue

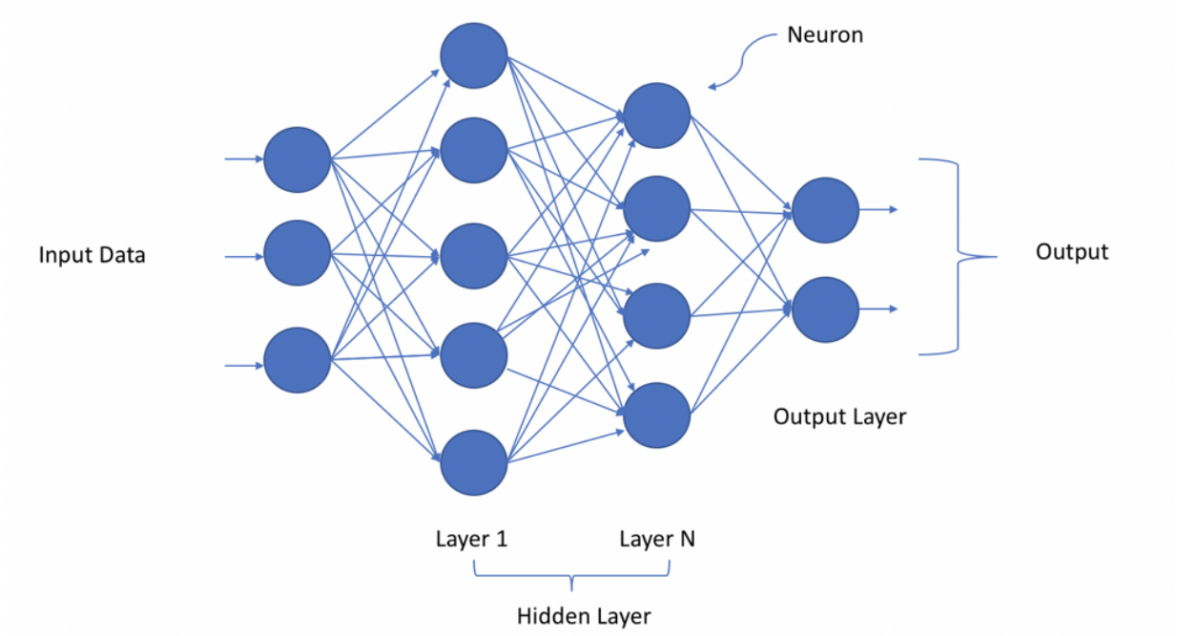


Figure 2.2: Deep neural network

related to language modeling. This thesis covers the theory related to Recurrent Neural Networks.

2.4.6 Recurrent neural Network(RNN)

RNNs are specialized neural-based approaches that are effective at processing sequential information. An RNN model is made to identify the sequential properties of data and then use the patterns to forecast future events. It would feed the data from previous iterations into nodes of the network at the current iteration. It has some sort of memory that captures information or gives the ability network to memorize the sequences in the internal memory. RNNs are used with sequential data to generate outputs for a single step at one time and give the prediction for the upcoming sequences. Unlike feed-forward neural networks (FFNNs), it can have iterative connections. Normal feed-forward networks (FFN) directly map input into output, whereas RNNs employ both the previous outputs and the current input, adding a third hidden layer on top of the input and output layers. These sequences are commonly represented by a fixed-size vector of tokens that are fed sequentially (one by one) to a recurrent unit. For different time series applications for classification and predictions, the RNNs have achieved high accuracy and have gained in popularity [22, 63]. The figure below illustrates a simple RNN framework below 2.3. Figure 2.3 depicts the overall structure of a memory

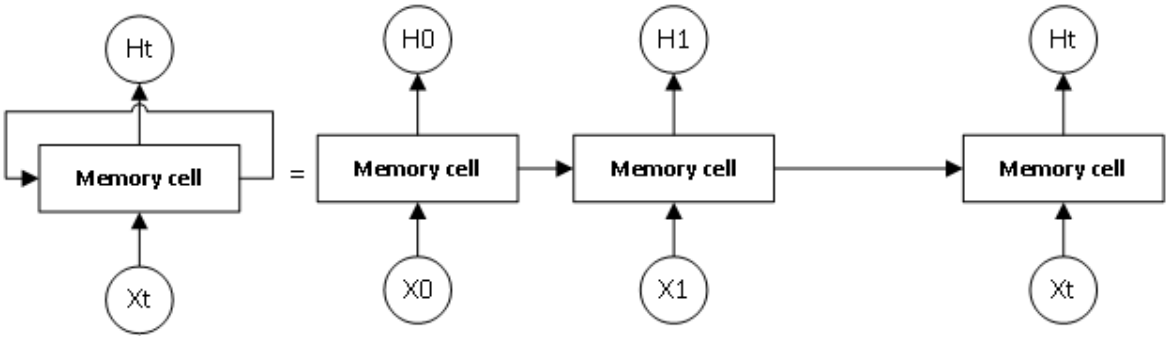


Figure 2.3: Recurrent Neural Network

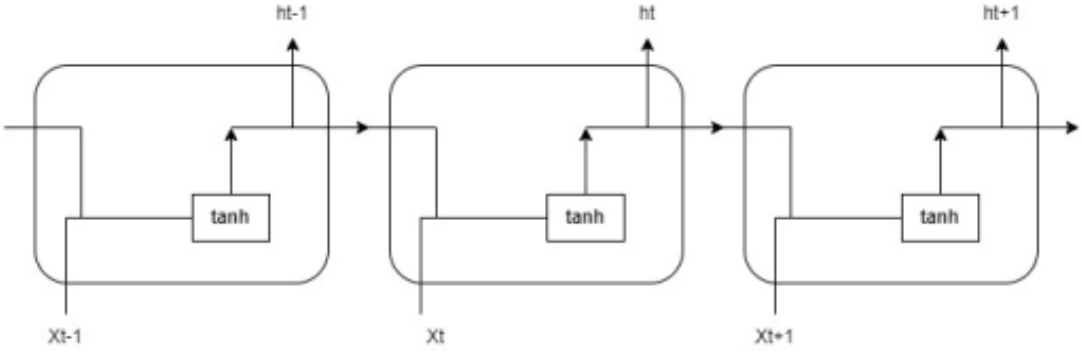


Figure 2.4: The repeating module in a standard RNN contains a single layer

cell, whereas Figure 2.4 illustrates its internal processes.

The main strength of an RNN is the capacity to memorize the results of previous computations and use that information in the current computation. This makes RNN models suitable to model context dependencies in inputs of arbitrary length so as to create a proper composition of the input. RNNs have been used to study various NLP tasks such as machine translation [70], image captioning, and language modeling, among others.

Here X=$(x_1, x_2, ..., x_n)$ is the fixed size input vector. H = $(H_0, H_1, H_2, ..., H_m)$ is the hidden layer that stores the history of the previous symbol e.g. at any step t there will be a vector H_{t-1} representing the history of the previous symbols. Y = $(Y_0, Y_1, Y_2, ..., Y_k)$ are the output symbols. The network's input at any time t is taken to be X_t, its hidden layer is taken to be H_t, and its output symbol is taken to be Y_t. The output of the hidden layer H_{t_1} at time t1 is combined with the vector V_t that represents the current word symbol to produce the input vector X_t.

$$X_t = V_t + H_{t-1} \tag{2.6}$$

$$H_t = tanh(WX_t + UH_{t-1} + b) \tag{2.7}$$

where W is the weight matrix of the input vector, U is the recurrent weight matrix and b is the bias.

$$Y_t = SoftMax(VH_t) \tag{2.8}$$

In equation 2.8, V is the weight of the Hidden layer. The SoftMax function converts the processed fixed-sized vector back into a probable output word.

When training the RNN uses a backpropagation algorithm. Common parameters are shared by all time steps in the network. The gradient and each output depend not on only the calculation of the current time steps but also on previous time steps. For example, in order to compute the gradient at time step T=5, we would need to backpropagate four steps and then sum up all the gradients known as backpropagation through time(BPTT). The fact that RNN would be BPT has difficulties training long-term dependencies i.e. dependencies between the steps that are far apart. This is due to the vanishing gradient problem. The vanishing gradient makes the network only remember recent events and forgets more distant past iterations. Encountering the vanishing gradient problem leads to extremely long training times and may lead to poor performance and low accuracy during the inference of the model [67]. Other variants, long short-term memory (LSTM) networks and gated-recurrent networks (GRU) were later introduced to overcome this limitation.

Long short term memory(LSTM

LSTMs are an advanced form of recurrent neural networks which are capable of handling long-term dependencies and retaining long-term information in the network. LSTMs were first introduced by Hochreiter and Schmidhuber in 1997 [30]. The gated units in LSTM differ from those in RNN but have the same chain architecture. These units carry out operations that allow for the retention or removal of information from the network. Figure 2.5 illustrates the additional components that distinguish LSTM from ordinary RNN.

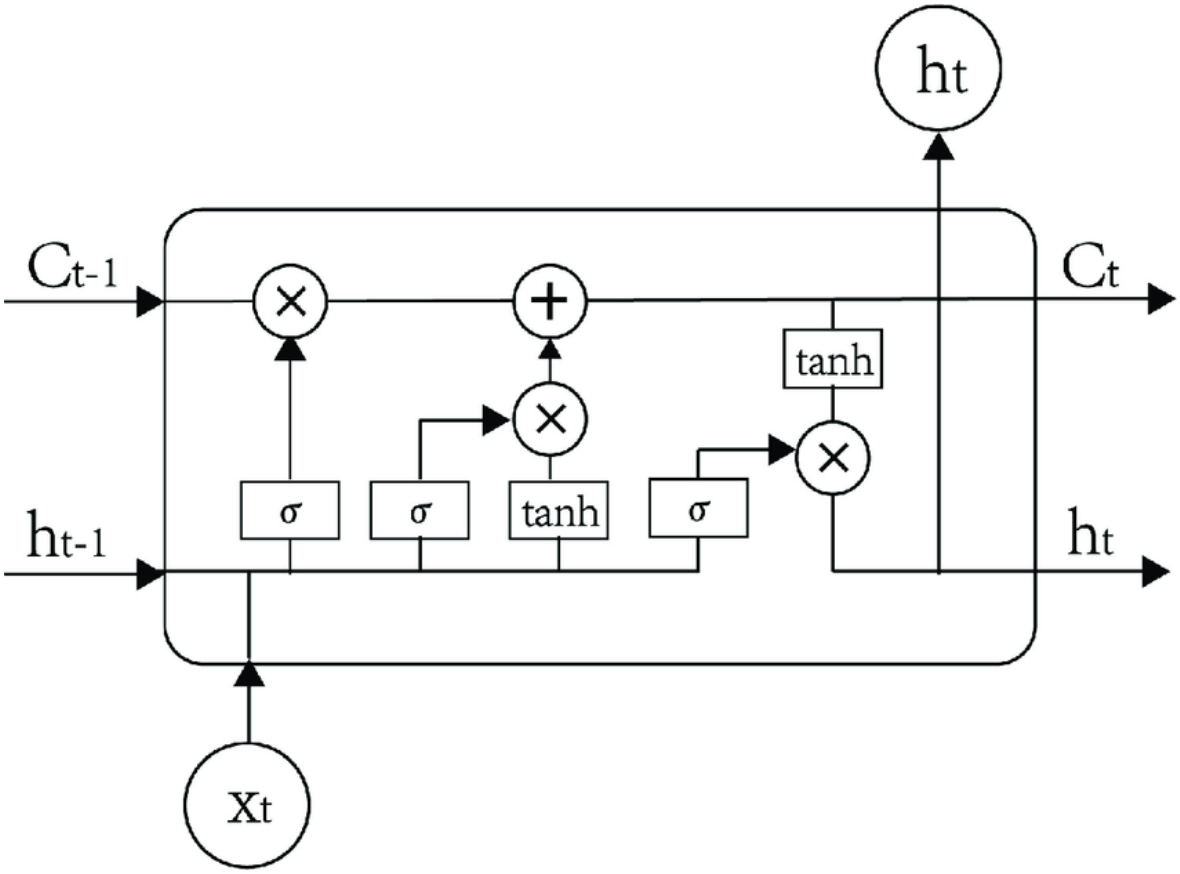


Figure 2.5: The repeating module in an LSTM contains one component
Source: [71]

An LSTM consists of three gates which are in charge of keeping and discarding past data in accordance with model inputs. These three gates are: forget gate, input gate and output gate. All of the gates listed above have sigmoid functions that are identical to tanh activation. They restrict the values to the range of zero to one, which helps the model's ability to remember or retain the data. When the sigmoid or tanh function is used, the value will be kept in the model if it is less than one and above zero, the value will be discarded if its output is zero.

Forget gate: The first block represented in the LSTM architecture is the forget gate (f_t). The sigmoid activation function passes the data from the current input (X_t) and the previous hidden state (h_t). If the output value is closer to 0 means forget, and closer to 1 means retain.

Input Gate: It works as an input to the cell state. It consists of two parts; first, we pass the previous hidden state (ht) and current input (Xt) into a sigmoid function to decide which values will be updated. The network will then be controlled by feeding the same two inputs into the tanh activation. The final step is to multiply the sigmoid output (it) by the tanh output (C't) to determine which information is crucial for updating the cell state.

Output gate: The hidden state contains information on previous inputs and is used for prediction. The output gate regulates the present hidden state (ht). The previous hidden state (h_{t-1}) and current input (x_t) are passed to the sigmoid function. This output is multiplied by the output of the tanh function to obtain the present hidden state. The current state (Ct) and present hidden state (ht) are the final outputs from a classic LSTM unit.

Cell state: The input from the previous cell state (C_{t-1}) is point wise multiplied with the forget gate output. If the forget output is 0, then it will discard the previous cell output (C_{t-1}). This output is point-wise added with the input gate output to update the new cell state (C_t). The present cell state will become the input to the next LSTM unit.

These are the mathematical operations for each stage.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{2.9}$$

where σ is the sigmoid function, f_t is the forget gate's activation vector, W is the weight matrix, x_t is the input vector for the LSTM cell, $h_t - 1$ is the hidden state vector and b_f is the bias.

Which data will be kept and which needs to be removed from the network will be decided in equation 2.9. sigmoid (σ) function will output a number in the range of 0 to 1. The values 0 and 1 indicate whether the network will keep and discard them from the network respectively. The input gate will decide which future information will be stored in the cell state in the following phase. It is divided into two parts. The network's input gates will first choose which values will be updated, then the cell state will choose which new values to add. The vectors representing the new values that will be stored in the cell state are created using the tanh activation function, and the cell state values are updated using the sigmod function.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2.10}$$

$$\widetilde{C}_t = tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{2.11}$$

where σ is the sigmoid function, it is the input gate's activation function. W, $x_t, h_t - 1, b_i$ and $\widetilde{C}_t$ are the weight matrix, input vector, hidden vector, bias and cell input activation vector, respectively.

In the next step, the cell state will be updated to a new state and then it will be used in the

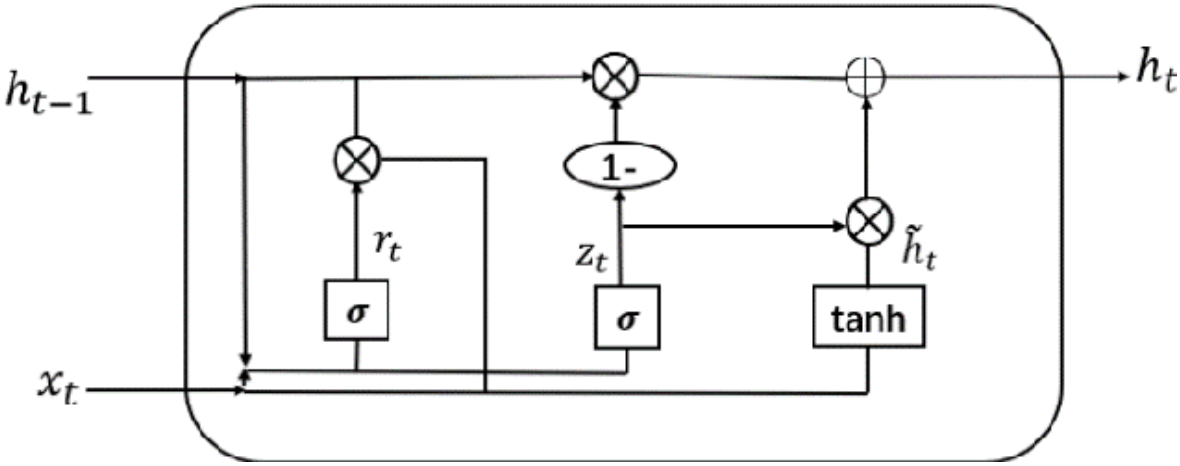


Figure 2.6: The repeating module in a GRU contains one component
Source: [73]

output gate.

$$C_t = f_t * C_{t-1} + i_t * \widehat{C}_t \tag{2.12}$$

In this step, the output will be generated by using the output gate. In output generation, the cell state plays an important role. First, the sigmoid function will be used to filter the values from the cell state and then a tanh layer will be used to generate the output.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{2.13}$$

$$h_t = O_t * tanh(C_t) \tag{2.14}$$

where o_t is the output at time t, W_o and b_o are the weight matrix and bias of the output gate labeled, respectively.

Gated Recurrent neural network(GRU)

Gated recurrent units (GRUs) another approach to solving the problem of short-term memory, was introduced in 2014 by Kyunghyun Cho et al [72]. It is the newer generation of RNNs and very similar to LSTMs. GRU and LSTM differ in that GRU do not have three gates like LSTM and instead uses a hidden state to transmit information from one gate to another. It has only two gates through which it passes the information. Figure 2.6 represents the single module of the GRU.

Reset Gate: It is used for the short-term information of the network. It determines how

the new input should be merged with the prior knowledge. Additionally, it has a sigmoid function that will convert the numbers to a range from 0 to 1. It multiplies the current input by the previous hidden state using the weights from the previous hidden state. The reset gate output and previous hidden state are then multiplied point-wise, and the resulting values are then added together before being handed on to the sigmoid.

Update Gate: The gate is in charge of determining which important details should be kept in the model and which should be discarded. It manages the term memory for the network. This gate is similar to the LSTM's input and forget gate arrangement.

$$z_t = \sigma(W_z.[h_{t-1}, x_t]) \tag{2.15}$$

$$r_t = \sigma(W_r.[h_{t-1}, x_t]) \tag{2.16}$$

$$\widetilde{h}_t = tanh(W.[r_t * h_{t-1}, x_t]) \tag{2.17}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widetilde{h}_t \tag{2.18}$$

2.4.7 Sequence to sequence model(seq2seq)

A sequence-to-sequence (Seq2Seq) model aims to map a fixed-length input with a fixed-length output where the length of the input and output may differ. Sequence-to-sequence neural networks consist of two parts, the encoder and the decoder.

The encoder is a recurrent neural network that gets a sequence of vectors $x = (x_1, x_2, ..., x_T)$ as input and generates hidden states h_1 to h_T with a recursive function:

$$h_t = f(x_t, h_{t-1}) \tag{2.19}$$

In the encoder-decoder approach, the last hidden state of the encoder is passed to the decoder as context vector c:

$$c = h_T \tag{2.20}$$

The decoder is another recurrent neural network that estimates the probability distribution of the next word in the output sequence, given the context vector and the previous words:

$$p(y_t) = f(s_{t-1}, s_t, c) \tag{2.21}$$

where s_{t-1} is the hidden state of the recurrent neural network and the context vector is used as the initial hidden state:

$$s_0 = c \tag{2.22}$$

$$s_t = f(y_{t-1}, s_{t-1}) \tag{2.23}$$

The input-output mapping tasks, such as machine translation, where this model has shown significant potential in the result. An input-side encoder records the representations in the data, and a decoder receives the representation from the encoder together with the input and produces a mapping to the target language that is appropriate.

2.5 Spell checker performance evaluation methods

The designed prototype must be evaluated in order to measure the effectiveness of the model. Evaluation metrics help us to measure the actual performance of spell checkers for both non-word and real-word spelling errors. A study by Kukich [7] provided the following criteria for evaluating a spell checker tool: vocabulary size, test set size, correction accuracy for single and multi-error misspellings, and type of errors. In the literature, several methods for evaluating a spelling error detection and correction system have been proposed. The most frequently used methods for measuring this performance are precision, recall and accuracy. In research described by Ahmadzade and Malekadeh [22], the accuracy of the model is evaluated using the edit distance between the predicted word and the actual word. Here a distance of zero means the model predicts correctly. A distance greater than 0 means that the model predicted is somehow correct but with some edit distance. Some neural spell checkers have been evaluated using accuracy [3, 15, 45, 24].

2.5.1 Recall

The recall is a measure of the completeness of the spellchecker, it tells you how much of the language the spellchecker covers. It measures the completeness of the correct and incorrect words. The completeness of the correct words is calculated by recall correct and the completeness of incorrect words is calculated using recall incorrect. **Recall correct**(R_c) is defined as the number of valid words in the text that are recognized by the spelling checker

(i.e. true positives), in relation to the total number of correct words in the text (i.e. the sum of all true positives and false negatives). It calculated using the equation 2.24

$$R_c = \frac{TP}{TP + FN} \tag{2.24}$$

Recall incorrect(R_i) is the number of invalid words in the text that are flagged by the spelling checker (i.e. true negatives), in relation to the total number of incorrect words in the text (i.e. the sum of all true negatives and false positives). It calculated using the equation 2.25:

$$R_i = \frac{TN}{TN + FP} \tag{2.25}$$

The ideal for any spelling checker would be, of course, to recognize all valid words as valid, and all invalid words as invalid, scoring 100% on both R_c and R_i.

2.5.2 Precision

Precision measures what percentage of words the model label as valid is actually valid. Precision is related to the accuracy of the system in flagging words. That is, it measures the accuracy of the system in identifying only correct words as valid and only incorrect words as invalid. It computes both correct and incorrect precision. **Precision correct**(P_c) is calculated by dividing all actual correct words identified by the model (i.e. true positives) by the total number of correct words flagged by the model (i.e. true positives plus false positives). It calculated using the equation 2.26:

$$P_c = \frac{TP}{TP + FP} \tag{2.26}$$

Precision incorrect(P_i) is the number of actual incorrect flagged by the model (i.e. true negatives) in relation to the total number of incorrect words identified by the model(i.e. true negatives plus false negatives).It calculated using the equation 2.27:

$$P_i = \frac{TN}{TN + FN} \tag{2.27}$$

2.5.3 Accuracy(A)

Accuracy is calculated from both precision and recall which measures the general quality of the spell checker. It measures the overall performance of the spell checker that has been computed.

$$Accuracy(A) = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.28}$$

2.6 Related work

Related works cover mainly the spell-checking and correction procedures, methodologies, and techniques researchers used. To the best of our knowledge, there has been no research conducted on the Sidaamu Afoo language on the Spell Checker. While this study is the first, different literature in local and foreign languages is reviewed.

We review related work by categorizing it into two approaches: the traditional-based approach and the deep learning-based approach. The spell checker's main tasks are the detection and correction of spelling errors. Currently, many researchers use deep learning approaches to improve the performance of detection and correction tasks. The literature employing deep learning approaches is reviewed under the category of deep learning-based approaches; otherwise, it is reviewed under the category of traditional-based approaches.

2.6.1 Traditional based approach

Many types of research were conducted for isolated and context-based word correction for local and foreign languages using this approach. To detect non-word spelling errors, the dictionary lookup, and n_gram approaches were used in different studies [7]. The dictionary lookup technique simply looks at an input string and checks if it appears in a dictionary. In contrast, n-gram error detection examines each n-gram in an input string and looks at the pre-compiled table of n-gram statistics. If the number of n-grams in a string is less frequent or nonexistent, such words are considered likely errors. Since these two techniques have to process large dictionaries, techniques such as hash tables, binary search and finite automata have been used to speed up the spell detection process. For Amharic Nigusu [48], Afaan Oromo Tirate [40] used the dictionary lookup with hashing approach to detect the non-word spelling error. They have tokenized the given sentences into words and the word is searched

in the list of stored words or dictionary.

However, in morphologically rich languages that can be inflected in different forms, the different rules were applied with the error detection techniques. For Afaan Oromo Gaddisa olani [39] using a morphological rule-based to detect and correct spelling errors. To fix the problem, the Hunspell [1] were used as a Knowledge base. To detect spelling errors, the roots and suffixes are searched in the dictionary. if root or suffix or both root and suffixes are not found in the dictionary, it detects as a misspelled word. Mariawit Shimelis [74] conducted a morphology-based spell checker for the Amharic language. She integrated the rule of character bi-gram and tri-gram with the dictionary lookup approach. Sankaran [75] analyzed the morphology and the difficulty of creating a dictionary with enough coverage for highly inflected languages due to numerous words that can be generated from a single word stem. Since Sidaamu Afoo is also a highly inflected language, this drawback affects Sidaamu Afoo as well.

The next step of the spell checker is error correction. Error correction is replacing or suggesting corrections for identified incorrect words in a given text. Traditional spelling error correctors first generate a list of candidate corrections, then rank these candidates and finally select the appropriate set of correction suggestions above a threshold value. Techniques used for isolated word correction (correcting one word at a time, without considering the context) include minimum edit distance-based methods [76]. For Amharic [48, 74] and Afaan Oromo [40, 39], rule-based methods [77], n-gram based techniques [78], and probabilistic methods (noisy channel model) [79]. For context-sensitive spell correction statistical methods such as Bayesian networks [80], Latent Semantic Analysis [81], and Statistical Machine Translation (SMT) [82] have been commonly applied.

2.6.2 Deep learning based approach

As in many other NLP applications, Deep Learning techniques have been used to implement spell correctors as well. The main advantage of using Deep Learning techniques over traditional methods is that they can capture non-linear correlations between text [83]. Therefore, Deep Learning based approaches can capture features in the text that cannot be expressed in a fixed set of manually created rules. Deep learning-based models for spell correction can be

[1] Hunspell is an open-source spell checker. It has been designed especially for languages with rich morphology and complex system of word compounding, originally for Hungarian. Link: http://hunspell.github.io/

implemented either as a sequence-to-sequence task, language modeling task or a sequence labeling task. It summarized in the the table 2.2

Sequence-to-sequence task

A sequence-to-sequence task is implemented using an encoder and a decoder, which are jointly trained using the text with spelling errors as the input, and its spell-corrected counterpart as the output. Thus this is similar to a Neural Machine Translation (NMT) task. Ghosh and Kristensson [15] proposed a method of Correction and Completion of spelling using an Encoder-Decoder (CCEAD) with an Attention mechanism. They are the first researchers to apply deep learning techniques to the English language. In the model, the encoder and the CNN are provided the character sequence as the input, while the decoder is provided the word sequence as the output. Malekzadeh and Ahmadzade [22] is proposed the same technique with the aforementioned study, which character-based encoder-decoder model for the Azerbaijani language. This model also employs the attention mechanism, which provided superior performance for spelling correction. The encoder uses a LSTM cell to convert the input sentences into feature vectors whereas the decoder also uses the LSTM to predict the next word through the softmax function. Etoori et al. [45] also used the encoder-decoder architecture to Indic languages Hindi and Telugu.
Zaky and Romadhony [21] also used a similar model, however in order to capture context they used the word Part of Speech (POS) tag and word embedding of context words as additional features. Büyük [17] also used individual words, however, they used surrounding words as additional features to capture context for the Turkish language.

Language modelling task

Sakaguchi et al.[3] proposed a word recognition model based on a semi-character level recurrent neural network (scRNN). They were motivated by the *cambridge effect* that the swapping position of the letter does not affect the spelling of the human being if only the first and last letters are correct. *Cambridge effect* demonstrates that human reading is resilient letter transposition if the first and last letters are correct. The following example is demonstrated in the paper:

"Aoccdrnig to arscheearch at Cmabrigde Uinervtisy, it deosn't mttaer in waht

oredr the ltteers in a word are, the olny iprmoetnt tihng is taht the frist and lsat ltteer be at the rghit pclae. The rset can be a toatl mses and you can sitll raed it wouthit porbelm. Tihs is bcuseae the huamn mnid deos not raed ervey lteter by istlef, but the wrod as a wlohe."

They used the semi-character input(scRNN) instead of using a whole word or character sequence. Each word is represented in three vectors: the first vector is the one-hot-encoded first character of the word, the second vector represents the middle character and the last vector is the one-hot-encoded last character of the word. The model is trained with the misspelled word as input and the corresponding correct word is predicted as the output via a softmax layer. Moslem et al. [24] used a language model with words as the input. This model was capable of generating only a possible set of predictions, the final selection was done by applying edit distance on the predicted words.

sequence labelling task

In a sequence labeling task, the labels may mark each word as having correct spellings or not. Such a task is capable only of spelling error detection [84]. When a sequence labeling task is used as a spelling corrector, the word itself becomes the label - a correct word has itself as the corresponding label, while an incorrect word has its corrected version as the corresponding label. A sequence labeling task is implemented using a single encoder, which needs text with spelling errors as the input. Li et al. [19] are the first to use the sequence labeling task for neural spell correction. The input to their model is a character sequence. The output of the character-level recurrent layer is fed to a word-level recurrent layer, which predicts the correct word as the label (thus the model name CharRNN-WordRNN). Han et al. [20] proposed a similar model for Chinese spell correction. Instead of recurrent models, Li et al. [85] used a Transformer based word encoder to extract global context information and a character encoder to encode spelling information. Jayanthi et al. [86] used a pre-trained transformer network. They averaged the sub-word representations to obtain the word representations, which were fed to a classifier to predict its correction.

Hybrid task

Wang et al. [87] combined a character-level sequence-to-sequence model and a word level language model. Li et al. [19] used a subword-based encoder, which was fine-tuned on a pre-trained language model. Sonnadara et al. [88] conducted the novel benchmark with the neural spell correction for the Sinhala language which is used in Sri Lanka. They used a hybrid of scRNN, LSTM-CharCNN [3] and CharRNN WordRNN [89] neural models for implementation.

2.6.3 Data preparation

To train all the deep learning approaches discussed above, a training corpus containing spelling errors, along with their corrections is needed. Different techniques have been employed to generate this training corpus.

Ghosh and Kristensson [15] Obtain a corpus written by humans and manually correct it. Jayanthi et al. [86], Li et al. [85] get a spell-corrected corpus and automatically replace a correct word with a miss-spelled word obtained from a list of miss-spelled words written by humans. Zhou et al. [90] Use a record of consecutive corrections by a user on a piece of text. Li et al. [19] generate synthetic errors by considering phonetic similarity. Pruthi et al. [91], Zaky and Romadhony [21], Sakaguchi et al. [3], Büyük [17] generate spelling errors by random operations. These include jumbling, deleting, inserting, repeating and substituting characters in individual words, splitting sentences at random positions, changing word order, and dropping words. Etoori et al. [45], Ahmadzade and Malekzadeh [22] are generate synthetic spelling errors considering errors humans usually make.

2.7 Summary

The aim of this study was to design the isolated-word and context-based spell checkers for Sidaamu Afoo. Different spell-checking techniques are applied for different languages. So, we reviewed different papers to achieve the goals of this study. Even though there were no spell-checking techniques tried yet for Sidaamu Afoo, we selected the techniques that are used for different languages. we have done the experiments for both isolated and context-based spell checkers. For the isolated word spell checker, we selected the dictionary lookup

method for error detection that was used for many different languages. To correct the isolated word correction method we selected two different methods like minimum edit distance and encoder-decoder model that shows good performance for the tradition-based approach and deep learning-based approach respectively.

For the context-based spell corrector, the experiment was conducted using n_gram language [48, 40], neural language model[24]. In addition, we apply the attention-based neural language model for the context-based spell checker for Sidaamu Afoo. The error is generated based on [17, 3] that generates one of the typographic error operations like insertion, deletion, substitution and transposition in the corrected data. The model is evaluated using different spell checker evaluation metrics like recall, precision and accuracy for each experiment.

Table 2.2: Summary of related works

Title (Year)	Architecture	Input	Language	Data	Finding
Neural networks for text correction and completion in keyboard decoding (2017)	Encoder decoder with attention + CNN	Encoder-character, decoder -word	English	2M sentences	98.1 % word and 68.9 % sequence accuracy.
Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network (2017)	Single encoder language model	semi character	English	39,832 sentences	Accuracy of 98.96 %- permuted internal letters, 96.70 % -single inserted letter and 85.74 %- deleted letters.
Spelling error correction using nested RNN and pseudo training (2018)	Single encoder sequence labeling	character	English	One Billion Words	It outperforms the PyEnchant spell checker and character convolutional neural network
Automatic spelling correction for resource-scarce languages using deep learning (2018)	Encoder-decoder with attention	Character	Telugu, Hindi	108587 words	85.4% accuracy for Hindi and 89.3% for the Telugu language
Spell correction for azerbaijani language using deep neural networks (2021)	Encoder decoder with attention	character	Azerbijani	12k sentences	The model predicted 75% distance zero i.e. model predicts correctly, 90% in distance one, 96% distance two and 98% distance three.
Arabisc: Context-sensitive neural spelling checker (2020)	Single encoder language model	word	Arabic	554,622 sentences	The model detects and exactly suggest for all 20 sentences of non-word spelling, whereas detected all 20 real-word spelling and 19 exactly suggested for real-word spelling.
Context-dependent sequence to sequence Turkish spelling correction (2020)	Encoder-Decoder with attention	Character	Turkish	4M sentences	Correction accuracy 94% on the synthetic dataset.
An lstm-based spell checker for indonesian text (2019)	Encoder-decoder	character	Indonesia	31,812 words	83.76% accuracy.

Chapter Three

Sidaamu Afoo Writing system

3.1 Overview

This chapter discusses the linguistic characteristics of the Sidaamu Afoo. Section 3.2 gives a brief description of the language, mainly the writing system and the way spelling error occurs in the writing system. Section 3.3 discusses the phonology of the language. Some of the basic morphophenetics that can cause for the spelling error in Sidaamu Afoo is discussed in section 3.4. Lastly, section 3.5 discussed the major part of the speech in Sidaamu Afoo

3.2 Sidaamu Afoo

Sidaamu Afoo is a Highland East Cushitic language spoken in the south-central part of the Federal Democratic Republic of Ethiopia. It belongs to the High Land East Cushitic language and is closely related to Kambaatisa (62%), Halaaba (Alaba) (64%), Hadiyyisa (53%), Kabeena (64%), Gedeuffa (40%), and Burji (41%)[92]. The language is spoken in the Sidaama National Regional State (SNRS) in Ethiopia. As Tafesse G/Mariam [35] stated, Sidaamu Afoo is witnessing rapid standardization and development in the last two decades starting from the late 1990s. Since 1991, its own official writing system has been the Latin alphabet system. It serves as a medium of instruction in the primary and secondary schools in SNRS. It has a department at Hawassa university and Hawassa teacher training college. Sidaamu Afoo, with some modifications, shares many features with the English writing system, as does the language's writing alphabet. Thus, letters in English or Latin alphabets are also found in Sidaamu Afoo, except for the ways they are combined in phonetic alphabets and the styles in which they are uttered. However, the linguistic structure is completely

different. In Sidaamu Afoo, the construction of sentences is Subject-Object-Verb. But in English, Subject-Verb-Object is the arrangement of the sentences. The language syllables structure obeys the (C)V(C)(C) general template for the formation of words, but the most common one is CV. This does not mean all words start with a CV, but some words begin with a VC syllable structure [35]. Where C is the consonant ad V is the vowel.

In Sidaamu Afoo, words are constructed from vowels and consonant combinations. Vowels are sound makers and can sound by themselves, while consonants are sound receivers and cannot sound by themselves. Except sh, both consonants and vowels in the word can be shortened and lengthened. The maximum number of consonants or vowels that can occur successively in Sidaamu Afoo is two. Otherwise, it is misspelled. Sometimes it seems to take more than two similar vowels with different utterances. At that time, it must take a double glottal stop or a single glottal stop to segment these vowels. For instance, *'"kuu"u" - those* Similarly, it cannot have more than two consecutive consonants, one after the other. A single consonant and two serial consonants in a word will produce two different meanings of that word.

3.3 Phonology of Sidaamu Afoo

3.3.1 Consonant phonemes

Sidaamu Afoo has 24 language-specific [‘, B, C, CH, D, DH, F, G, H, J, K, L, M, N, NY, PH, Q, R, S, SH, T, W, X, Y’] and five borrowed [P, TS, V, Z, ZH] consonants in total 29 consonant phonemes and are displayed according to their place and manner of articulation in Table 3.1. By the borrowed consonant there are no native words in Sidaamu Afoo that are formed from these characters. They are called **borrowed (Argete Fidalla)"**. For example, the consonant **z** is found in only in Amharic loanwords such as **/Zufaane/** ’throne’(Amh. zufan) and **Muuze** ’banana’(Amh. muz)[35].

Table 3.1: Sidaamu Afoo consonant phonemes

	Bilabial	Labio dental	Dental	Alveolar	Palatal	Velar	Glottal
Stop Plosive Ejective Implosive	b p’	t d t’ dh			k g k’		’
Affricate Ejective			c j sh				
Fricative		f	s z	sh			h
Nasal	m	n	n		ny		
Tap/Flap			r				
Lateral Approximant			l				
Approximant	w				w		

The consonant can be simplex or geminate in the word. When it is simplex or geminated, it can change the meaning of the word. for example, *”gowa” - sewing* and *”gowwa” - fool*. The meanings of the two words are different due to the simplex and geminated consonant ”r”. The simplex and geminate characters are not all consonants; some consonants do not geminate at all, and some are rare at simplex. The **Siwilu fidalla** ch, dh, ny ,ph, sh, ts, zh are never geminated in the Sidaamu Afoo except sh. Consonants cannot be geminated at the beginning and end of the word. Example, */bbaatto/* instead of *”baatto” - earth*.

3.3.2 Vowel phonemes

There are five short vowels [A, E, I, O, U] and five long counterparts[AA, EE, II, OO, UU]. Its representation of the language is depicted in Table 3.2

Table 3.2: Sidaamu Afoo vowel phonemes

	Front	Centeral	Back
High	i/ii		u/uu
Front	e/ee		o/oo
low		a/aa	

They are similar to those in English, but they are uttered differently. Each short or long vowel is pronounced in a similar way throughout its usage in any Sidaamu Afoo literature. The vowel can be shortened or lengthened in the word. Vowel shortening and lengthening in a word can alter its meaning. In Sidaamu Afoo, the shortening and lengthening of the vowel make the spelling error. Many words can be formed by only lengthening and shortening the vowels. It is one of the main reasons for the real-word error. There are many words whose meaning is altered by the vowel shortening and lengthening. For example, *gowa* means sewing, while *goowa* means neck. In the above words, the lengthening and shortening of the vowel /o/ make the meaning difference between the two words. All vowels have the same character in the word. Table 3.3 shows some examples of how to differentiate the meanings of words in the case of vowel shortening and lengthening in the word. It can also be shortened and lengthened at the beginning, middle or/and end of the word. For instance,

At beginning

/**aa**na/, 'on the something',

/**oo**so/,'children'

At middle

/m**aa**la/, 'The meat'

/G**aa**la/, 'Camel'

At end /Od**oo**/, 'News'

Table 3.3: An example of the shortening and lengthening of the vowel

vowel	shorthen	meaning	longer	Meaning
a	Sada	Fame	Saada	Cattle
	Jawa	Elder	Jaawa	-
	Gala	Spent the night	Gaala	Camel
e	Tenne	at that time, then	Teenne	Flies
i	Sinna	Branches	Siinna	Coffee Cups
o	Dora	Clay soil	Doora	Elect
	Hoga	Leaf of "Enset"(false banana)	Hooga	Loss
u	Kula	to tell	Kuula	blackish blue

3.4 Morphophonemic process

The maximum number of consonants in a sequence is two. According to [35], there are different types of morphophonemic rules in Sidaamu Afoo; (i) epenthesis, (ii) metathesis, and (iii) a set of assimilation rules.

3.4.1 Epenthesis

As explained, in Sidaamu Afoo, the maximum number of consonants in a sequence is two. However, The epenthesis vowel **i** is inserted in the following condition.

- When suffixes such as $/-tV/$ are added to mono-consonantal verb stems, the epenthetic vowel /i/ is inserted between the stem and suffix. For example,
  $/Y - tu/ \rightarrow /Yitu/$, 'She/they said'.

- The epenthetic vowel **/i/** is inserted into the word when three consecutive consonants that appear in the **/i/** sound must be inserted between the second and the third consonant to keep linguistic rule secure. For example
  $/Sirb - tu/ \rightarrow /Sirbitu/$, 'She/they sang'
  $/Kubb - tu/ \rightarrow /Kubbitu/$, 'she/they jumped'
  The word /kubb-tu/ cannot appear like /kubbt/ this is not allowed in Sidaamu Afoo. To avoid situation **/i/** sound must be insert between geminating last 'b' and 't' sound.

So the correct form should be **/Sirbitu/** and **/kubbitu/** the correct representation of the word.

- When a single causative **/-s/** or a double causative **/-siis/** is added to stem-final obstruents. for example
  $/it - s/ \rightarrow /itis - /$, 'feed'
  $/it - siis/ \rightarrow /itisiis - /$,'cause to be eaten'

3.4.2 Metathesis

Metathesis is a phonological process whereby two adjacent consonant phonemes are transposed. The phenomena happen in the Sidaamu Afoo when the stem of the verb ends without stress for those which have **/-n/** as an initial suffix. For example, **/ninke inte/**-*'We ate'* the word **/inte/** constructed from two sounds **/?it/**-*'eat'* and **'-ne'** and expected to be **/itne/** but the language never allow such king of phonological syntax. For that reason, the word is represented in the **/inte/** manner to facilitate speech convenience for communication.

3.4.3 Assimilation

Assimilation is the behavior of sound influenced by or influenced by neighborhood sounds to be in some manner. The assimilation can be avoidance (removal of sound from the word), of sound in partial or complete assimilation.
For Example,

Word	**Assimilated sound**
/dunkeemmo/- [dunkeemmo] - 'we carry'	**n + k → nk**
/hajanjeemmo/- [hajañjeemmo] - *'we order'*	n+j ->ñj
/hanbeemmo/- [hambeemmo] - *'we forget'*	n+b ->mb
/kul-neemmo/-/kulneemmo/ [kulleemmo]- *'we tell'*	l+n->ll
/kad- kada/ - [kakkada]-*'jumped over'*	d+k ->kk

There are morphophonemic phenomena in the Sidaamu Afoo language like glottal stops. For example

Word	Morphophonemic form
/Saa/ - *'Cow'*	**/Sa'a/** - *'To pass'*
/Tee/ - 'She'	**/Te'e/** - *'That one'*

3.5 Major parts of speech

This section describes how parts of speech in Sidaamu Afoo can be characterized according to their morph syntactic properties. Works in have stated that the Sidaama language has five basic word classes: noun, ad-position, adverb, conjunction and verb [93, 94]. Each of these classes again can be divided into other sub-classes. For instance, noun class is categorized as a proper noun, common noun, pronoun, Preposition and postpositions are subclasses of ad-positions. The subclasses in turn can be divided into subclasses and the subdivision process may continue iteratively depending on the level and aim of the investigation

3.5.1 Noun

Nouns are any word that can be used to name or identify a place, object, or idea. Two types of grammatical genders exist in Sidaamu Afoo nouns. These are masculine and feminine, and the entire nouns of the language belong to one of these gender categories. Similarly, there are two numbers (singular and plural) that can be identified by the morpheme it adds. The plural form of a given noun can be formed by adding a suffix to the root noun. Various types of suffixes can be added to transform a singular noun into its plural form. All of these suffixes change the singular noun to plural without variation in meaning. The last vowel of the singular noun is dropped before the suffix is added. There are several forms of suffixes that turn verbs into nouns, namely, ***-a, -o, -e, -anso, -atto, -ano, -ille, -imma***, and ***-aancho***. Because the citation form of any word in Sidaama ends in -a, -o, or -e, it is sometimes difficult to decide whether nouns ending in one of these vowels derive from verbs by adding a normalizing suffix to the verb stem, or the verbs derive from the nouns.

3.5.2 Verb

In Sidaamu Afoo verbs are words that are used to indicate action, state of being, and event occurrences within time boundaries. It shows what the subject is doing in the sentences. It

can be transitive, intransitive, modals and auxiliary verbs. Transitive verbs are those verbs that transfer message to complement or object whereas, intransitive verbs do not transfer message to complement and hence, do not have complement or object. The following examples illustrate this fact.

Durreette hando hidhitu- *'Durreette bought an ox'*. The verb **"hidhitu"**-'*bought*' is transitive and a "**hando**"-'*Ox*' is the direct object because it receives the action (a hando-'Ox' is what is being bought).

3.5.3 Pronoun

Pronouns are words that are used to take the place of nouns in sentences. Similar to that of nouns, pronouns have numbers and gender. For example, ***ise/isi***-*she/he* which means **ise**-*'she'* is feminine (singular) whereas isi which means **isi**-*'he'* is masculine (singular) and **insa**-*'they'* is plural can be masculine or feminine. Pronouns can also be categorized based on their functions and meanings in the sentence. These are personal pronouns, possessive pronouns, demonstrative pronouns, relative pronouns or reciprocal pronouns. For example.

Durreette hando hidhitu- *'Durreette bought an ox'*.

Ise hando hidhitu- *'She bought an ox'*.

3.5.4 Adverb

Adverbs are any words that explain or modify verbs. These can be adverbs of time, place, manner, frequency, etc. Adverbs precede verbs they modify in the Sidaamu Afoo language. For example.

Dawassi rahe dayi- 'Dawassa came quickly'. Here ***rahe*** -*'quickly'* is an adverb.

Durreette ga'a hadhano.- 'Durreette will go tomorrow '. Here, **ga'a**-*'tomorrow'* is an adverb.

Qananiisi yanna wo'manta qolchanno.-*'Kananisa wins every time'*. Here, **Yanna wo'manta**-*'every time'* is an adverb.

3.5.5 Preposition

Words that can have full meaning only in combination with some other words like nouns, adjectives, or verbs are termed prepositions/postpositions. They do not take any affix and belong to the closed part of speech. Prepositions or postpositions may precede or follow the category to which they add syntactic meaning. Let us see the following sentences:

Addisu maxafa xarapheesu aana wori.-*'Addisu put the book on the table'*. **aana**-*'on the'* in this sentence is postposition. As it can be seen it followed the category to which it adds meaning that is **xarapheesu**-*'table'*. But in the sentence:

Addisu mini widira dodi.-*'Addisu ran to home'*. The word **widira**-'to' is a preposition and in this case, it precedes the category to which it adds meaning.

Chapter Four

System design and architecture

4.1 Overview

System architecture design helps to understand the given NLP tasks. NLP tasks have their own properties that make them different from many other deep learning tasks. This task aims to detect and suggest corrections for the non-word and real-word spelling errors in the Sidaamu Afoo text. This chapter discusses the proposed architecture of an isolated and context-based spell checker for Sidaamu Afoo. Section 4.2 gives a general description of the proposed spell-checker model. Sections 4.3 and 4.4 explain the data used in the model and how the data is preprocessed. Section 4.5 explains the features used in the model and the methods that prepare it in a suitable format for each model. Section 4.6 describes the methods of the data preparation for the encoder-decoder model. The methods of converting data into vector representations using one hot representation and embedding representations are discussed in sections 4.8 and 4.9. Sections 4.10 and 4.11 discuss the methods for creating a model needed for spelling detection and correction, as well as the flow of the steps to detect and correct a spelling error in Sidaamu Afoo.

4.2 Proposed architecture

This research is focused on both isolated word correction and context-based word correction for written Sidaamu texts. Accordingly, the architecture of the proposed spell checker contains three components, namely preprocessing, feature extraction and spell detector and corrector, as shown in figure 4.1 and the detail architecture in figure 4.2. The preprocessing phase is responsible for accepting the data from the user, cleaning it of unnecessary noise,

normalizing the data and tokenizing it into smaller tokens. The feature extraction phase is responsible for extracting a feature from data in a suitable format for a model entry. Lastly, the spell detector and corrector phase accepts the feature from the feature extraction phase and analyzes it for spelling errors and provides suggestions for invalid words.

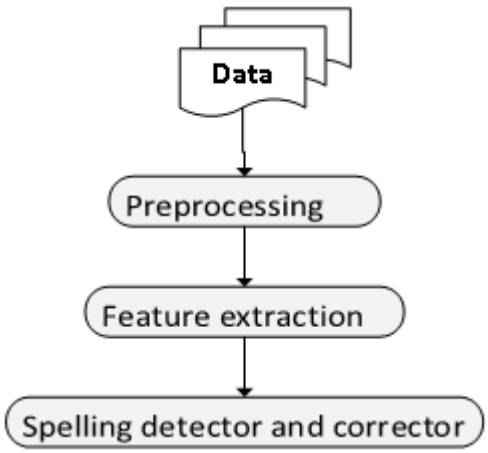


Figure 4.1: General architecture of the proposed model

In this study, the experiment is conducted for two independent spell checkers, namely an isolated-word spell checker and a context-based spell checker, as shown in the spell corrector and detector phases in the architecture 4.2. An isolated-word spell checker's task is to check non-word errors in a given input sentence word by word, without taking the context of words into account. It detected it using the dictionary lookup approach with hashing from a prepared dictionary. After a non-word error is detected, an encoder-decoder model is used to correct the detected misspelled word, with the encoder encoding the character of the detected misspelled word as a source sentence and the decoder predicting the corrected word as a target sentence in terms of neural machine translation. A context-based spell checker's task is to check both non-word and real-word errors from a given input sentence word by word by taking the context of the words into account. The LSTM model with an attention mechanism is used to detect and correct the contextual error. The LSTM has some sort of memory, providing them with the possibility to forget unnecessary information and keep the necessary information for next-word prediction [68]. It predicts the next word using the context of previous words and checks whether the current word is contextually correct or not by computing the edit distance between the current word and the predicted words.

In other words, for the current word, first check for a non-word error, then check for a

contextual error. To check for the spelling error of the current word, all previous words must be free of any spelling errors. For instance, the sentence (S) has n words, where n is the total number of words in the sentence. S=$w_1, w_2, w_3, ..., w_n$, w_1 must be a correctly spelled word in order to check w_2 for contextual spelling, whereas w_1 to $w_n - 1$ must be free of any spelling error in order to detect and correct w_n contextual spelling error. In this case, the n-gram features are extracted from the given sentences, where n starts from the first word and is incrementally tokenized to the last word.

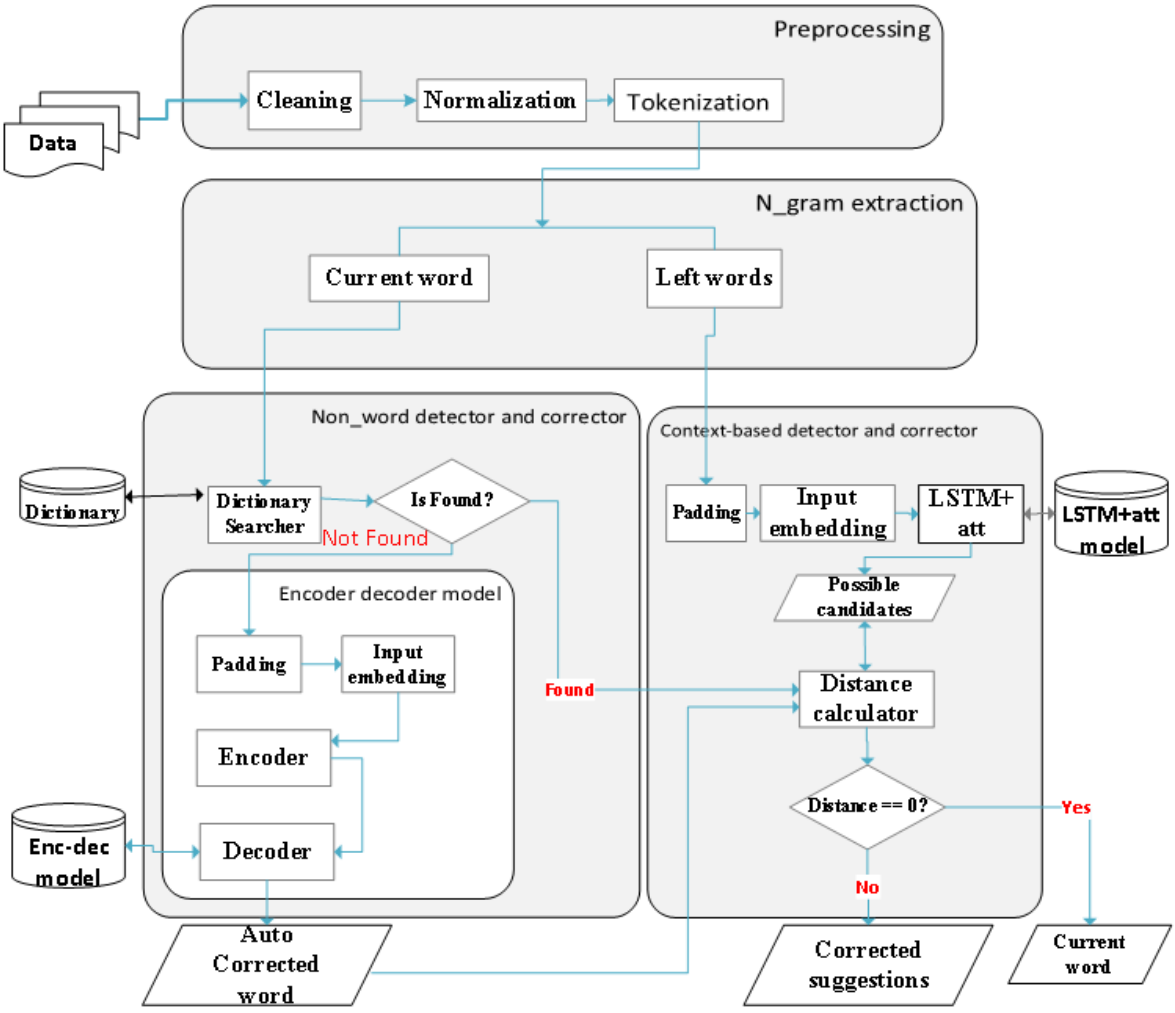


Figure 4.2: Detail architecture of the proposed model

4.3 Data

A data or text data is a language resource consisting of a large and unstructured set of texts. Our data is text data, which consists of sequences of sentences having different lengths. All

experiments in this study are performed on the data gathered from the most publicly available websites in the Sidaamu Afoo language and some digital text collected from different sources. Typically, real-world data is incomplete, inconsistent, inaccurate (contains errors or outliers), and often lacks appropriate representation of trends. To make the data suitable for spelling correction, it should pass each of the spelling correction components.

4.4 Preprocessing

Data preprocessing is a process of preparing the raw data and making it suitable for a spell correction model. It is the first and crucial step while creating a spell correction model. As we explain above, real-world data generally contains spelling errors, unwanted characters, non Sidaamu Afoo characters and may be in an unusable format that cannot be directly used for spell correction models. Data preprocessing is a required task for cleaning and making it suitable for a spell correction model, which also increases the accuracy and efficiency of a spell correction model. We have a total of 61,725 raw sentences with 425,628 words and 102,564 unique words. The preprocessing involves the following steps:

4.4.1 Cleaning text

We have removed the extra space between the words, punctuation and extra long and extra short sentences. Except for punctuation that used for glottal sound in Sidaamu Afoo, such as single hyphen (') and double hyphen (''), and sentence delimiter punctuation used for tokenizing sentences, such as full stop (.), question mark (?), and exclamation point (!), all special characters and punctuations, are removed. We removed extra short sentences with fewer than three words and extra long sentences with more than twenty words during the cleaning process. 6,285 sentences were removed from a total of 61,725 sentences, which is 10.17% of the corpus. We believe that this may not affect the effectiveness of the final model while deep learning requires a large set of data. After data cleaning, 55,440 sentences remain for model training and testing purpose. Finally, to conduct a spelling correction task, it is important to ensure that the sentences in our data set are spelled correctly. The language expert checks spelling errors to ensure the correctness of some data, while other data, such as the Sidaamu Afoo bible and the Sidaama national and regional state constitution, is assumed

to be correctly spelled because it was written by a language expert.

4.4.2 Normalization

The goal of normalization in this study is to ensure the text is in the same case. We used the Python programming language to conduct experiments, and it treats uppercase and lowercase differently. For example, in the case of ”Q” and ”q,” Python treats the threats *'Quchuma'* and *'quchuma' - city* as two distinct words. Sidaamu Afoo has only lowercase and uppercase letters. We convert our text data to lowercase using Python's built-in lower()[1] function.

Algorithm 1 Algorithm for data cleaning

Input: Raw text
Output: Cleaned text

```
function CLEANTEXT(text)
    text= re.sub(”\s+ ”,” ”,text)                ▷ To remove more than one space
    text = re.sub('[^A-Za-z”]', ””, text)
    text = text.lower()
end function
cleaned_corpus ← CLEANTEXT(text)
```

4.4.3 Tokenization

Tokenization is the process of segmenting the text into smaller pieces called *tokens*. As per the task performed, the corpus can be tokenized into paragraphs, sentences, phrases, words, sub-words, or characters. An understanding of sentence and word boundaries is crucial in the tokenization process. We did not get any literature to review for a sentence delimiter of Sidaamu Afoo. For Afaan Oromo [40], a language that uses the Latin script and has the same spoken family as Sidaamu Afoo, uses a space to show the end of one word, as well as an exclamation point (!), a period (.), and a question mark (?) to show a word and sentence delimiter. So, we tokenized the corpus into sentences using sentence delimiters like an exclamation point (!), a period (.), and a question mark (?). In order to know the boundary of end sentences, we have added *”bos”* to indicate the beginning of a sentence and *”eos”* to indicate the end of the sentences.

[1]The **lower()** is the python built-in function that converts all uppercase characters in a string into lowercase characters and returns it.

We use a built-in function of the *tf.keras.preprocessing.text.Tokenizer(filters=' ')* tokenizer [2] class tokenizes the collection of sentences into words, which is tokenized by white space and automatically removes punctuation in the words. We update the *filters* parameter in the function to keep the single and double quotation marks. It is also used to assign a unique integer (often referred to as an index) to each unique word in our vocabulary. It creates a dictionary *word_index* that contains key/value pairs of the unique word. This method creates the dictionary index based on word frequency; the most common word will have a tokenized index value of 1, the next most common word will have an index value of 2, and the last least frequent word will have an index of the total number of unique words minus one. For instance, the sentence S, *sidaamu dagoomu qoqqowi qaru quchumi hawaasaati*. Assume *sidaamu* appeared 20 times in the corpus, *dagoomu* appeared 18 times, *qoqqowi* appeared 21 times, *qaru* appeared 5 times, *quchumi* appeared 3 times, and *hawaasaati* appeared 8 times. The tokenized result in the form of the dictionary *word_index* representation is in Table 4.1. After tokenizing the all sentences, we get 58,801 unique words and each word has a unique number.

Table 4.1: Example of the word index

Word_index	
Word	**Index**
...	0
qoqqowi	1
Sidaamu	2
dagoomu	3
hawaasaati	4
qaru	5
quchumi	6

4.5 N_gram extraction

In this study, N-gram extraction is one of the features extracted and is used to prepare the corpus in a suitable format for entry into the model. Since we have no well-prepared corpus

[2] *https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer*

for this purpose, we extracted the features based on the models' requirements. The n-gram feature is extracted by splitting sentences into different lengths of n-grams, where n starts at two and iterates incrementally up to the end of the sentences. The basic idea behind splitting the data into different lengths of n-grams is to check the contextual spelling of the word by predicting the next word. To correct the context-based spelling by predicting the next words, we used the LSTM model with attention mechanisms, which requires data in the form of sequences of words. For instance, the above sentence, *sidaamu dagoomu qoqqowi qaru quchumi hawaasaati* are split into:

- **Sentences**: *sidaamu dagoomu qoqqowi qaru quchumi hawaasaati*
- **word_index**: 'sidaamu':1, 'dagoomu':2, 'qoqqowi':3, 'qaru':4, 'quchumi':5, 'hawaasaati':6
- **N_gram model for this sentences**

  [1,2]=['sidaamu', 'dagoomu']

  [1,2,3]=['sidaamu', 'dagoomu', 'qoqqowi']

  [1,2,3,4]=['sidaamu', 'dagoomu', 'qoqqowi', 'qaru']

  [1,2,3,4,5]=['sidaamu', 'dagoomu', 'qoqqowi', 'qaru', 'quchumi']

  [1,2,3,4,5,6]=['sidaamu', 'dagoomu', 'qoqqowi', 'qaru', 'quchumi', 'hawaasaati']

Algorithm 2 An algorithm for splitting words and creating the sequence of the n-grams

Input: Normalized data

Output: Sequences of n-grams and word list

function TOKENIZE(data)
    $seq \leftarrow []$
    $tokenizer \leftarrow Tokenizer()$
    $tokenizer.fit\_on\_texts(data)$
    $word\_index = tokenizer.word\_index$
    for $sentences\ in\ data$ **do**
        for $word \leftarrow 1 \ldots lengthofsentences - 1$ **do**
            $ngram \leftarrow sentences[: word + 1]$
            $seq.append(ngrams)$
        end for
    end for
    $Return\ word\_index, seq$
end function

In order to predict the next word (output), previous words (input words) must be known. In our data, the list of n-grams contains both input and output data. The sequence of n_gram has been split into the input and expected output. In each n-gram, the last word is an expected output, and the remaining left-hand words are input. For instance, in the bigram model, the first word is the input, and the second word is the expected output. For trigrams, the first two words are in the input and the last word is in the expected output.

Data encoding and splitting are demonstrated in algorithm 3.

Algorithm 3 An algorithm for creating the encoding and splitting n-grams

Input: Sequences of n-grams
Output: splitted n-grams

1: **function** ENCODESPLIT_DATA(length)
2: $seq \leftarrow legth\_grams$
3: $tokenizer \leftarrow Tokenizer()$ ▷ Using keras tokenizer
4: $tokenizer.fit\_on\_texts(seq)$
5: $sequences \leftarrow tokenizer.texts\_to\_sequences$
6: $sequences \leftarrow np.array(sequences)$
7: $vocab \leftarrow len(tokenizer.word\_counts) + 1$
8: $data\_x \leftarrow sequences[:, :-1]$
9: $data\_y \leftarrow sequences[:, -1]$
10: $data\_y \leftarrow to\_categorical(data\_y, num\_classes \leftarrow vocab)$
11: $return\ data\_x, data\_y, vocab, words\_to\_index$
12: **end function**

The algorithm takes the sequence of the n-gram in lines 1 and 2. The text data is converted into a numerical value by using the Keras Tokenizer class in lines 3 and 4. Line 6 converts the numerical values into an array. The total number of the vocabulary is counted in line 7. Line 8 and Line 9 split the sequence of n-grams into X and Y, i.e., the last word of the n-gram is Y, and the left one is X. The input data is encoded at line 10 and the required value is returned at line 11.

4.6 Parallel data preparation

In this study, to train and test encoder-decoder models, the source is the corpus with spelling errors and the target is its spell-corrected counterpart is needed. In the literature, different techniques have been employed to generate the parallel corpus for spell correctors. For the high-resourced language, the confusion set was prepared that spelling errors based on

humans usually make [45, 22]. However, for the low-resource language, the researcher generates spelling errors using different methods. In the study [15, 86, 85], get a spell-corrected corpus and automatically replace a correct word with an incorrect word obtained from a list of incorrect words written by humans. Some study is generated synthetic data by adding phonetic similarity [19]. Many works of literature generate spelling errors by random operations of typographic error including jumbling, deleting, inserting, repeating and substituting characters in individual words [3, 91, 21, 17], splitting sentences at random positions, changing word ordering, and dropping words [43].

To the best of our knowledge, there are no well-prepared incorrect and correct pairs of data in Sidaamu Afoo. Due to the lack of data with error patterns in Sidaamu languages, we decided to create a noised dataset that contains spelling errors by adding artificial spelling errors to the original corpus. For each instance of a unique word, different noisy words are generated by introducing one operation of typographic error including insertion, deletion, substitution or transposition. We have randomly added a typographic error in a random position in the word. In the training data, added 80% of a spelling error in the word, so that some words can remain as it is. The error-generated algorithm is depicted in algorithm 4 and the snip shot of the code is depicted in appendix A.1 and sample data is shown in B.2

Algorithm 4 An algorithm for generating random spelling error in the word

Input: Correct word
Output: Misspelled word

```
1:  function ADD_SPELLING_ERROR(word,error_rate)
2:      assert(0.0 ≤ error_rate < 1.0)
3:      if length of word < 3 then
4:          return word
5:      end if
6:      rand ← randoms_number
7:      prob ← error_rate/4
8:      if rand < pro then
9:          Delete random character in the word
10:     else if pro < rand < pro * 2 then
11:         Replace random character in the word
12:     else if prob * 2 < rand < prob * 3 then
13:         Add random character in the word
14:     else if prob * 3 < rand < prob * 4 then
15:         Transpose random character in the word
16:     else
17:         pass
18:     end if
19: end function
```

4.7 Padding

The extracted feature of our data has a different length, but the neural network model requires each input sequence of tokens to be the same shape and size. To make all tokens the same shape, padding is necessary. The sequence is padded to the maximum length in the entire corpus. There are two kinds of padding: pre-padding and post-padding. Pre-padding means all the sequences are padded with zeroes in the beginning according to the longest sequence's length or a chosen length longer than the longest length. During post-padding, all the sequences are padded with zeroes at the end according to the longest sequence's length or a chosen length longer than the longest length.

We have padded each n-gram sequence with zeros to the length of the longest n-gram and each word. The maximum sequence length is identified as the length of the longest n-gram sequence. In the corpus, the longest n-gram is 21. So, we padded all the n-grams to a length of 21 at pre-padding. We used the *pad_sequences* function to pad all n-gram sequences to length 21.

For instance, n-gram sequences [4,5],[4,5,3],[4,5,3,7],[4,5,3,7,8],[4,5,3,7,8,6] are padded in to a length 21 are:

4,5 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,3]

4,5,3 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,3,1],

4,5,3,7 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,3,1,5],

4,5,3,7,8 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,3,1,5,6]

4,5,3,7,8,6 = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,3,1,5,6,4]

4.8 One-hot representation

After changing the dataset to an integer representation, it has to be changed into a two-dimensional longer vector which is, a one-hot vector representation in order to make suitable spell correction. This one hot representation uses a unique vector representation for each word of the sentences. The neural network does not directly operate on the vocabulary represented with the integer. To operate the vocabulary by the neural network it should change to vector representation, which is called one-hot vector representation. We represented the vocabulary by 1 corresponding to its position, else with 0. For instance, If one of our input sentences is *"sidaamu dagoomu qoqqowi qaru quchumi hawaasaati"* the one hot encoding vectors for this sentence are:

$$sidaamu = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad dagoomu = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad qoqqowi = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad qaru = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad quchumi = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad hawaasaati = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

4.9 Input Embedding

From sequences of vector representation (one-hot representation), word embedding will be formed. The one-hot encoding conveys no information about the meanings of words. In particular, it does not reflect whether given words are similar to each other or completely different. When we have passed one hot vector data directly to the neural network, there may face some problem training waste rather than learning when zero values are encountered. It also requires very high memory when vocabulary is large. To solve this problem, we have used word embedding, which changes the higher dimensional one-hot vector to a lower dimensional vector.

In NLP, word embedding is about the word context. It captures semantic and syntactic relationships between words. When using embeddings, semantically or syntactically similar words are represented by a similar vector i.e. vector having low distance[68]. It optimizes our strategy (i.e. data processing and model complexity) to enable our model to be better able to learn the relationships among words. Probably the first significant neural network-based word embedding model was proposed by [63] in the form of a neural probabilistic language model. One of the word embedding methods is wor2vec developed by Mikolov [68]. In its current form, our data is providing a forward context for our model to learn. Word embedding converts words to real number numeric representation between -1 and 1. In other words, we know what words are more or less likely to follow other words due to our n_gram sequences or word sequences. However, the tokens used to represent each word are nothing but frequency counts which provide the model with limited information regarding the contextual information among the words.

We used the Keras embedding layer to generate word embedding. The Embedding layer is seeded with random weights between -1 to 1 and learns an embedding for every word in the training dataset. And obtain the embedded vectors for each word. After that, the vectors are used to represent words in a sentence. It specifies the input dimension (the size of the vocabulary in the text data), the output dimension (the size of the vector space in which the words are embedded), and the input length (the length of the input sequences)

4.10 Model building

A corpus was collected from different sources to detect and correct the spelling errors in Sidaamu Afoo. To make the corpus suitable for the given task, the corpus were preprocessed. We prepared data that was used for the building of a different model. The n-gram(tri-gram, bigram, and uni-gram) model, the encoder-decoder model, and the neural language model with and without attention are basic models.

4.10.1 Dictionary construction

A word dictionary is a list of corrected words that helps to detect and correct misspelled words. The unique words of the corpus were used as a dictionary. Each word in the dictionary was free from spelling errors, which is valid to represent Sidaamu Afoo words. To check for errors in the dictionary, the tokenized words were sorted by the frequency of words in the corpus. We checked the least frequent words manually because we assumed that the word would be the least frequent in the case of a spelling error. Well-prepared words were stored in the form of a dictionary and accepted as corrected words.

A simple dictionary list of words is adequate for non-word error detection and can also be used to produce suggestions for correction by finding words that closely resemble the misspelling. The most straightforward and widely used method for a computer spell checker to detect non-word errors is dictionary look-up. The spell checker then looks up each word in the text to be checked in its list and flags as misspelled any that are not found. The question to be answered at this stage is how many and which words should be included in the list.

The dictionary size affects the response time since it checks the user's word against all dictionary words. The HashMap data structure was chosen by the developer because it is fast, efficient, and accurate, to retrieve from the dictionary. It is also efficient in storage space, making it ideal for use in situations where memory was limited. The words were stored in the disk and positioned in the hash function for fast retrieving and accessing data from the lists of words.

4.10.2 Neural language model construction

The neural language model detects and corrects context-based spelling errors in the given sentences. Depending on the size of the training corpus, neural language models were applied for contextual spelling error correction. The collected corpus from different sources was preprocessed to construct the neural language model. For good prediction, the preprocessed corpus was split into different n-gram lengths, and each n-gram was divided into an input part (the predictor) and an expected output part (the labels), where n can be one, two, three, or the maximum length of the sentences minus one. In the corpus, the maximum length of the sentences is 20. For the m-word sentence w_1, w_2, w_3, w_4, ..., w_m can be splitted into $w_1w_2, w_1w_2w_3, w_1w_2w_3w_4, ..., w_1w_2w_3...w_m$ based on the its respective numeric values. We padded each sequence of the n-gram to the maximum length of the n-gram in the training corpus. Then, we consider the last element of all n_gram sequences as a label. Then, we need to perform one-hot encoding on labels corresponding to the total number of words in the training corpus. For instance, we have sentences like *'sidaamu dagoomu qoqqowi qaru quchumi hawaasaati'* and this will convert into a sequence with their respective tokens *'sidaamu':1, 'dagoomu':2, 'qoqqowi':3, 'qaru':4, 'quchumi':5, 'hawaasaati':6*. Thus, output will be ['1' ,'2' ,'3','4', '5', '6']. Likewise, our all sentences will be converted into sequences. Then, we will make a n_gram model for good prediction. Below is explained how it works.

- **Sentences**: *sidaamu dagoomu qoqqowi qaru quchumi hawaasaati*
- **word_index**: 'sidaamu':1, 'dagoomu':2, 'qoqqowi':3, 'qaru':4, 'quchumi':5, 'hawaasaati':6
- **Text sequence:** [1,2,3,4,5,6]
- **N_gram model for this sentences**

  [1,2,3,4,5,6]=['sidaamu', 'dagoomu', 'qoqqowi', 'qaru', 'quchumi', 'hawaasaati']

  [1,2,3,4,5]=['sidaamu', 'dagoomu', 'qoqqowi', 'qaru', 'quchumi']

  [1,2,3,4]=['sidaamu', 'dagoomu', 'qoqqowi', 'qaru']

  [1,2,3]=['sidaamu', 'dagoomu', 'qoqqowi']

  [1,2]=['sidaamu', 'dagoomu']

- **Padded each n_gram in maximum length(maximum length is 6 in this case)**

  [1,2,3,4,5,6]=[1,2,3,4,5,6]

  [1,2,3,4,5] = [0,1,2,3,4,5]

  [1,2,3,4] = [0,0,1,2,3,4]

  [1,2,3] = [0,0,0,1,2,3]

  [1,2] = [0,0,0,0,1,2]

- **Preparing predictor and labels for each padded n_grams**

  [1,2,3,4,5,6] =[1,2,3,4,5],6

  [0,1,2,3,4] = [0,1,2,3,4,5],5

  [0,0,1,2,3,4] = [0,0,1,2,3],4

  [0,0,0,1,2,3] = [0,0,0,1,2],3

  [0,0,0,0,1,2] = [0,0,0,0,1] 2

Finally, using the encoded data, the model is trained for future use. The basic idea here is the predicting the label by entering the predictor in the model. Unlike n_gram language model, a recurrent neural network (RNN) model architecture incorporates previous data to make future predictions that all the inputs are related to each other. In other words, it has a built-in memory function that stores information from previous words and uses that information when predicting the next word. In addition, this approach solves the data sparsity problem and contextual relationship by representing words as vectors and using them as inputs to a neural language model. Word embeddings obtained through neural language models can capture the contextual information between words and the semantical relation of the words [68]. However, normal RNN, it has a vanishing gradient problem, which is the problem of remembering long-distance relations. To tackle the vanishing gradient problem, we use the LSTM unit which was designed by Hochreiter and Schmidhuber [30]. When we are predicting the next word for a sequence, all words of the sequence don't contribute equally to the prediction of that word. So, we have added an attention layer to the output of LSTM. The model is compiled and run by defined parameters and saved for future use.

4.10.3 Encoder-decoder model construction

The dictionary lookup and encoder-decoder model are combined together to detect and correct the non-word spelling error. The encoder-decoder model takes the characters of the detected misspelled words using a dictionary lookup approach and predicted the corrected words. It operates at the character level. It has been commonly used in machine translation in the past. Then it was used for different NLP tasks and which have achieved significant results on complex tasks, mainly data that have different lengths of the sequence, like machine translation, question answering, image captioning, etc. In the architecture of the neural machine translation, the encoder consumes source sentences and the decoder provides translated target sentences. In this study for spelling correction, the misspelled word is treated as a source, which is translated into the target with the error-corrected word. Encoder-decoder model has two basic components: encoder and decoder. Its goal is for the encoder to map a variable-length misspelled input to a fixed-length vector, and the decoder maps the vector representation back to a variable-length correctly spelled output, where the lengths of the input and output may vary.

The encoder-decoder model is constructed when the model is trained. The encoder model takes the input sequence and encodes all information of the input data into a list of vectors known as context vectors. For the decoder model, we passed the embedding of the target (correct) word, this method is known as teacher forcing. The decoder receives the correct output from the training set as the previously decoded result to predict the next output rather than receiving the previously predicted result. In our Encoder-Decoder model, we have used a RNN-based model to design the encoder and decoder for the encoder-decoder model. Recurrent Neural Networks (RNNs) are popular models that have shown great promise in many Natural Language Processing or NLP tasks. The RNN are described in section 2.4.6. The LSTM are used for each RNN nodes to tackle the vanishing gradient problem. Lastly, the model is compiled and run using the defined parameter and save it for future use.

4.11 Spelling detection and correction techniques

The spelling detection and correction phase is one of the basic components of the study. It shows the techniques and algorithms used to detect and correct spelling errors in the Sidaamu

Afoo text. In the architecture diagram Figure 4.2, there are two independent spell detectors and correctors: a non-word and a context-based spell detector and corrector. The non-word spell checker is a spell checker that detects and corrects non-word spelling errors without considering the context of the word. whereas the context-based spell checker detects and corrects the spelling error by considering the context of the word. It corrects both non-word and real-word spelling errors.

4.11.1 Non-word spelling detection and correction

To detect the non-word spelling, we find the input word in the dictionary. The **dictionary searcher** is responsible to search for input words into the dictionary. The dictionary is the list of the correctly spelled words of the Sidaamu Afoo. Dictionary searcher searches input words from the entire dictionary. However, the size of the dictionary may affect the time complexity when searching for input words in the dictionary [63]. If the size of the vocabulary is huge, it may take a long search time. To reduce the searching time of the dictionary, different techniques were explored as explained in section 2.3. The most efficient and widely used method is hashing which accelerates the search process. It provides O (1) average time complexity to lookup input words in the hash table [52]. More specifically, hashing is used for this research to search an input string in a pre-compiled hashing via a key or a hash address associated with a word and retrieve the word stored in the hash table.

The hash function retrieves input words from the hash table. The hash table contains vocabulary in the form of the python dictionary (key, value) format i.e. each vocabulary is stored in the hash table with the computed result of the vocabulary. As a result, for this study hash table is implemented by python. In spell-checking context, if the words stored at the hash address are the same as the input string which is the value of the hash address. However, if the input string/word and retrieved word are not the same or the word stored in the hash address is null, the input word is indicated as a misspelling.

Algorithm 6 Algorithm of the hash function

Output: Boolean True or False
Input: Testing word $w_0, w_3, .., w_n.$ and Hash table

1: **function** $Search(key)$ ▷ key=input words
2: $value \leftarrow HashAdress(key)$
3: **if** $HashFunction(key) == value$ **then**
4: $return\ true$
5: **else**
6: $return\ False$
7: **end if**
8: **end function**

Once a non-word spelling is detected, an error correction algorithm is called to find corrections for the incorrect word. We have constructed a Levenshtein edit distance and encoder-decoder model.

Levenshtein Edit Distance

As described in section 2.4, Levenshtein Edit Distance is used to compute the distance between two strings. It is also widely used for non-word spell correction [40, 48, 12]. It was implemented to find the minimum operation which includes insertion, deletion, substitution and transposition to find the candidate correction for the misspelled one. Insertion occurs when a letter is inserted, deletion occurs when a letter is removed, substitution indicates to the replacement of a letter and transportation occurs by swapping two letters into the misspelled word in order to produce the corrected words. The distance is used to search for appropriate candidate corrections for misspelled words i.e. the minimum distance words between misspelled words and dictionary words were provided for the candidates of correction.

Encoder-decoder model

In this study, we used the encoder-decoder model to correct the non-word spelling error of Sidaamu Afoo text. We regard the detected word as the source language and its corrected counterpart as the target language in terms of neural machine translation. As explained above in section 2.4.7, the encoder-decoder model has two basic components: encoder and decoder. We use a recurrent neural network (RNN) special LSTM encoder to compute a sequence of hidden state vectors h = $[h_1, h_2, ..., h_n]$. These hidden features represent information from

each element in the input sequence.

Encoding is the process of converting the sequences of inputs into a single-valued context vector. The context vector is used to understand the information the LSTM learns till time steps. To encode into a context vector, the encoder accepts the embedded vector of the incorrect words' characters and encoded its information to the context vector. It accepts a single element of the input sequence at each time step, processes it, collects information for that element and propagates it forward. At each time step t, it reads one character, then updates and records a hidden state. When reading a character, it calculates the current hidden state h_t using the current character x_t and the previous hidden state h_{t-1}. When it finishes reading the end-of sequence character \n, it selects the last hidden state h_{T_x} as a context vector c. So, to read the N length character, the N time step is used where N is the length of the input. For example, the word *"sidamu"* is the incorrect word due to the deletion of character *"a"* from the word *"sidaamu"*. The input word has 6 characters $[s, i, d, a, m, u]$, so LSTM reads the sequence of characters at 6-time steps. Each character is represented as a vector using character embedding.

$$h_t = f(h_{t-1}, x_t) \tag{4.1}$$

and

$$c = h_{T_x} \tag{4.2}$$

Where f is some non-linear function that maps a character of misspelled word x_t into a hidden state ht at time t by considering the previous hidden state h_{t1}. The last hidden state h_{T_x} is selected as a context vector c.

The decoder part uses the output of the encoder as an initial hidden state, current character and its own hidden state, in order to predict the next character. It trained to output a corrected word by predicting the next character based on the context vector c and all the previously predicted characters. Then, it generates the corrected character y by sequentially predicting a character y_t conditioned on the misspelled character of the word's context c as well as previous character $y_1, y_2, ..., y_{t1}$. It performs implicit language modeling as follows.

$$P(y) = \prod_{t=1}^{T} p(y_t|y_1, ..., y_{t1}, c) \tag{4.3}$$

The decoder then sequentially generates the corrected character based on the context vector at equation 4.3. It first sets the context vector as an initial hidden state of the decoder LSTM. At each time stamp t, it generates one character based on the current hidden state and the context vector. Then, it updates the hidden state using the generated characters at equation 4.1. Here the model keeps predicting the next character until the end charcter $\backslash t$ character is received or the maximum decoder sequence length is exceeded.

It provided automatically corrected words for the erroneous word. We used the LSTM for each RNN nodes. Figure 4.3 shows an example of the LSTM Encoder-Decoder model for correcting the misspelled word *"sidamu"* instead of *"sidaamu"*. The encoder LSTM reads the character of detected word *"sidamu"* one by one with each time step. When it reads the first character *"s"*, it embeds the character into vector and computes the current hidden state h_1 using embedded value of the input *"s"*. Then, it reads the second character *"i"*, embeds it, and updates the hidden state h1 to h2 using embedded matrix of *"i"*. The procedure continues until the encoder reads the last character *"$\backslash n$"* and gets the final state h_7. The final state h_7 is selected as a context vector c.

Algorithm 7 Algorithm of non-word spelling correction

Output: Corrected sequence $o_0, o_1, ..., o_N$
Input: Input sequence $c_1, c_2, ..., c_N$

function $correctoutput$
    $encodedSequence \leftarrow encodedsequence(c)$
    $modelOutputs \leftarrow runModel(encodedsequence)$
    $p_0, p_1, ..., p_N \leftarrow getProbabilities(modelOutputs)$
    $index \leftarrow 0$
    while $i \leq N$ **do**
        $predictedIndex \leftarrow argmax(p_i)$
        $o_i \leftarrow indexToCharacter(predictedIndex)$
        $i \leftarrow i + 1$
    end while
end function

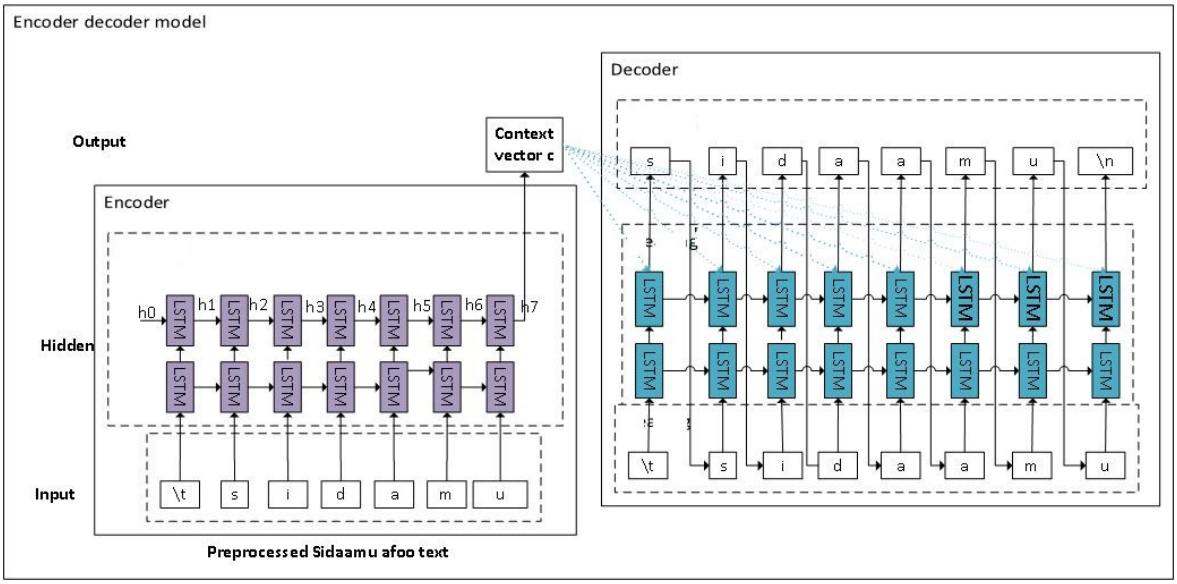


Figure 4.3: Encoder decoder spelling correction example

The decoder LSTM tries to predict the corrected character greedily using the context vector c. It predicts only single top value character within each time steps. It first predict *"\t"* as the first character. Then, it computes a hidden state h1 based on the context vector c and initial *"\t"* value, and predicts the first character *"s"* according to h_1. It then calculates the next hidden state h_2 using the previous character *"s"* vector, the context vector c, and predicts the second character *"i"* according to h_2. The steps continues until it predicts the end of sequence character *"\n"*. If the predicted word and target word do match, the detected word is replaced by the predicted word. If a predicted word and target words do not match, the misspelled word is left as it is.

4.11.2 Context-based spelling detection and correction

The context-based spell checker detects and corrects both non-word and real-word spelling errors. We address the problem of detecting and correcting contextual errors using the language model. The language model assigns low probability to rare occurring sentences and higher probability to more common sentences. From the natural language correction perspective, sentences with contextual spelling errors should have a lower probability than their corrected versions. The reason for this is the fact that the language models are trained on clean corpora, which usually consist of correct text with a minimal number of errors. Because nowadays language models operate on a word level, we often refer to them as word-

level language models.

We conduct the experiment of the contextual spelling corrector using both a statistical and a neural language model for the given input data. The sentences are split in the same fashion with training data that incrementally split into the different lengths of n_grams. For each n_gram, the last word is the word that we want to check for contextual correctness in the sentence. In this way, each word of the given sentence is checked for spelling correctness. To detect and correct context-based spelling in Sidaamu Afoo, some top next words were predicted based on the remaining words of the n_grams, and then the last word (input word) of the n_gram was searched to see if it was one of the predicted words. The distance calculator is responsible for calculating the distance between the input word and the predicted words. If one of the distances between the input word and the predicted words is zero, the input word is decided as a correct word; otherwise, it is a misspelled word.

Statistical language model

The statistical approach uses the chain rule of probability and the Markov assumption to rewrite the probability of a whole sequence as a product of n-gram probabilities. The sentences are split into tetragram, trigram, and bigram lengths of the sequence, and the spelling error of each word is checked one by one. To check the spelling of a current word, tetra-gram predicts the top N words using the previous three words. If the last word is in the predicted lists or if the probability is not zero, the input word is correctly spelled. Using the tetragram results in models that capture more context and, as a result, are expected to provide better estimates. However, they often suffer from data sparsity, which is caused by words that are unobserved in the training data. As a result, we used backoff smoothing, which employs a lower n-gram if the likelihood of a higher n-gram occurring is zero. If a word is not found in the predicted candidates even after smoothing, it is a misspelled word, and the words with the minimum distance are provided for the suggestion.

LSTM model

We used the stacked LSTM for the neural language model. When checking the spelling of each word, it checked using a left context window of the given n-grams. For the given sentences each word from beginning to end, the contextual spelling is checked one by one,

and the correctness of the word is based on the all left words of n-gram.

In the decoding process, the neural language model with the LSTM network reads the embedded vector of the left context window each input n-grams(left context window) as input and predicts the next word. Neural language models are normally utilized for text generation to predict the next token or next few tokens in a sequence given the preceding tokens as context. Our neural network greedily decodes to predict for the most likely sequences. Greedy decoding predicts the top best candidates. However, in our case, instead of keeping only the 1-best candidate, we keep the n-best predicted candidates and then calculate the edit distance between each candidate and the test word. We observed that n for the n-best list is a sensitive hyper-parameter, i.e. when we increase the size of this hyper-parameter, we obtain better vocabulary coverage and more suggestions at the expense of many less probable candidates. In this case, the decoder may choose an incorrect word as a possible suggestion. Therefore, the value of n of the n-best list is the total number of candidates for the incorrect word. But, we suggest for user only candidate which has minimum distance. If the minimum distance(ed) between the current word and one of the n-best predicted words is zero, the current word is spelled correctly. If, ed > 0 this indicates that there are other suggestions for the current position in best predicted candidates. If the current word is not found in the predicted best candidates or found but after several suggestions, there are chances that for the current context one of these suggestions is better than the current word. So, the model provides alternative words for the suggestion which has edit distance one between current word and predicted candidates.

LSTM + Attention model

This model is similar to the previous LSTM model, except we have added an attention layer to the output of LSTM. All of the words in a sequence do not contribute equally to the prediction of the next word for that sequence. Thus to provide weights to each of the words in a sequence, we add an attention layer. The context-based spell detector and correction using a neural language model are depicted in algorithm 8.

Algorithm 8 Context-based spell checker of Sidaamu Afoo using LSTM language model

Input: Input n-gram sequence $s_0, s_1, ..., s_N$
Output: Corrected sequence $o_0, 0_1, ..., o_N$

function CORRECTEDSUGGESTION(S)
    $Sug \leftarrow []$
    $leftW \leftarrow s_0, ..., s_{n-1}$
    $currentW \leftarrow s_n$
    $encodedLeftInput(leftW)$
    $lenleftW \leftarrow lengthOfleftW$
    $b_0, b_1, ..., b_B \leftarrow LSTMatt.predict(encodedLeftInput).argsort(n)$ ▷ predict best n words b-best
    for $b_0 \dots b_B$ **do**
        $output \leftarrow decode(b)$
        if $EditDistance(b, currentW) == 0$ **then**
            $Break$ ▷ Correct word
        else if $EditDistance(b, currentW) == 1$ **then**
            $suggestion.append(b)$
        else
            Continue ▷ No suggestion
        end if
    end for
    return Sug
end function

Chapter Five

Experiment and evaluation

5.1 Overview

In this chapter, the tools and environments that are used to implement the designed algorithm and the experiment that is conducted to demonstrate spelling error detection and correction accuracy could be presented. It explains how the corpus used for this study was prepared and used and the challenges faced when preparing. In addition, the experimental setup, the evaluation parameter and the result achieved have been discussed. Section 5.2 discusses the source of data and the total number used in each model. Section 5.3 explains the tools, environments, and the total number of experiments conducted. Section 5.4 lists and discusses the hyperparameter selected in the model. Section 5.5 the evaluation result of the spelling error detection and correction of all conducted experiments. Section 5.6 discusses the result of the experiments.

5.2 Data collection and preparation

When conducting research on spell checking, a corpus is a crucial tool. A corpus is a large and structured set of texts. A huge corpus is needed for deep learning models. However, based on the complexity of the problem and the complexity of the learning algorithm, the size of the data may vary from research to research. The problem of spell-checking is the problem of vocabulary coverage. However, standard dictionaries are unable to fully capture the vocabulary of human languages due to their complexity and abundance of words and concepts, as well as domain-specific idioms, proper names, technical terminologies, and special jargon[48]. Most researchers collect data in a specific domain and evaluate the model

in the sense of that domain. Even though in a specific domain, for low-resource languages, it is difficult to cover all the vocabulary that is in the domain. Sidaamu Afoo is one of the low-resource languages with no well-prepared data for deep learning model training and testing.

To the best of our knowledge, there is no standard established corpus for the Sidaamu Afoo spell checker. To accurately train and test the spell checker models that have been created, texts in the Sidaamu Afoo language were collected from a variety of sources. The texts were collected from the publicly available website to conduct the experiment, which was collected from the Faith Comes by Hearing (FCBH) website and the Ethiopian Press Agency. The publicly available corpora contain text from publicly accessible sources and were collected from the internet by a web crawler. The Sidaamu Afoo Holly Bible and the Bakkalcho newspaper were crawled from the FCBH website [1] and the Ethiopian press agency[2] website, respectively.

The most important part of any natural language processing task is the preparation of training and testing the corpus properly. Since the collected corpus contains unnecessary characters and non-Sidaamu Afoo words, text preprocessing is very important to clarify the corpus. It is a very important aspect of corpus preparation to clean up unnecessary errors before training and testing the spell-checker model. The experiment in the Sidaamu Afoo spell checker was conducted with a prepared text, and the texts were prepared by the researchers themselves. The corpus was collected from various sources and manually cleared from any kinds of unnecessary errors. Each word in the corpus were make free from spelling errors, which is valid to represent Sidaamu Afoo vocabulary words. We create a well-understandable and clean Sidaamu Afoo corpora with linguistic expertise to create an error-free corpus for Sidaamu Afoo to maximize the accuracy and performance of the model. Training corpus preparation included preprocessing feature extraction and model building. The total amount of data collected from different sources is shown in the table 5.1.

We have put all the corpus in different text files named *sideBible.txt, Bakkalcho.txt and constitution.txt*. At the processing stage, the cleaning is done for each set of data. Then the text is transformed to lowercase. Therefore, all the letters will be in the same case. By using regular expressions, all special characters have been removed from the sentences and extremely

[1] https://live.bible.is/bible/Sid
[2] https://press.et/bakalcho/

Table 5.1: source and size of the data collected for experiment

Data source	**Sentences**	**Vocabulary**	**Unique vocabulary**
Bible	57095	373220	91880
Bakkalcho	4012	43998	16356
constitution	618	8410	2727
Total	**61,725**	**425,628**	**102,564**

short and long sentences are removed. After preprocessing our corpus, all corpus is merged into a single file. A total of 55,440 sentences remain to train and test a model. The sentences split 90% for training and 10% for testing using the $train\_test\_split$ function. In order to reduce bias, we set the shuffle true in the function. So, we used 49,896 sentences to train a model and 5544 sentences to test a model. The unique word of the training sentences is used as a dictionary. Totally unique word of training data is 58,801 words and that is used as a dictionary in the study. Also, it is used for training the encoder-decoder model by adding spelling errors in each unique word. As we explain above, the spelling correctness of some text is checked by language experts, whereas another is assumed as correctly spelled text because it is written by a language expert. In addition, we sort the unique words based on the frequency in the corpus and the spelling correctness of the least frequently occurred words because the least frequent words can be misspelled words. The sample tokens and counter misspelling is represented in Appendix B.2

Each sentence of the training data is split into different lengths of n-grams. We have 349,896 n-grams. Where n can be 2, 3, ..., 20. We used to train the neural language model.

Table 5.2 is generalizing the total number of entire data used in each model.

5.2.1 Testing data

The test dataset is taken from our corpus by splitting into 10% for this purpose, which is a correctly spelled corpus and consists of 5544 sentences. As we explained in section 4.6, we generate a random error in each sentence and evaluate the data based on the correct test sentences. The test data is preprocessed in the same way as the training data.

5.3 Experimental setups

Python programming language is used in this research. Python is an open-source programming language that is very efficient in machine learning and deep learning applications. It has a lot of libraries that are already developed and can be used to implement different machine learning and deep learning algorithms. The Windows platform is used to develop, validate, and verify the proposed model. For our experiment, we used a Google Collaboratory GPU RTX5000 with a RAM specification of 12 GB to write and execute the code. It contains almost all the important libraries pre-installed and makes the execution fast because of the GPU. To write the report of the experiment, we used an overleaf editor of the markup language LaTeX. It is used to create professional-formatted documents and has the ability to collaborate with an advisor and a co-advisor.

To answer the research question, two main experiments were conducted for Sidaamu Afoo: An isolated word spell checker and a context-based spell checker. The isolated-based spell checker detects and corrects the non-word spelling error. It detects and corrects without taking into account the context of the words. whereas, the context-based spell checker is detect and corrects both non-word and real-word spelling errors. To detect and correct this kind of spelling error, the context of the word is considered. For each type, we conducted different experiments. Table 5.2 show the experiment conducted for each experiment and the total number of data used.

Table 5.2: Experiments and total number of data

Types	**Detection**	**Correction**	**Data**	
			size	unit
Isolated	dictionary lookup	edit distance	58801	words
	dictionary lookup	encoder decoder model		
context-based	N-gram	Statistical language model(N-gram)	58801 279,457 3642,86	tri-gram bi-gram uni-gram
	Neural language model (LSTM) + edit distance	Neural language model (LSTM)	349,896	n-gram lists
	Neural language model (LSTM + attention) + edit distance	Neural language model (LSTM+ attention)		

5.4 Parameter selection

Model optimization is one of the taasks in the implementation of deep-learning algorithms. Deep-learning optimization depends on some hidden elements that influence the behavior of the model. To improve predictions using neural networks, certain parameters must be modified. These hidden elements are known as hyperparameters. Hyperparameters can be selected either automatically or manually. Automatic selection is the default parameter in the library. whereas the manual selection is the user-defined parameter. In manual selection, it is a trial-and-error approach that gives the best result. There is no specific, definite, thumb rule on how many hyper-parameters one should choose [5].

We conducted encoder-decoder model experiments on various parameters to achieve the desired result using our training data. We define the parameter manually. Table 5.3 contains the list of hyperparameters and their corresponding values which resulted in the optimal performance of the proposed model. We selected the embedding dimensions of 128 and 256. And next, we selected latent dimensions or hidden units of 256, 512, and 1024. Our model is trained with the Adam optimizer with a learning rate of 0.01,0.001 and 0.0001. In order

to select the best dimension and embedding layer using the training data, we have done the experiments using both embedding dimensions with each latent dimension(hidden unit). To select the best dimension and embedding layer, we have used 64 batch size, and the 0.001 learning rate of Adam's optimizers, and the dropout rate, which is 0.2. Then we have done the experiments with 100 epochs. And finally, we have got the following results specified in Table 5.4. These results are almost the same, only slightly different in time. In all the experiments below, the loss level goes the same way from a higher loss level to a lower one.

Table 5.3: List of hyperparameters and its values used in proposed model

Hyper parameter	**Values**
Embedding layer	128,256
Latent dimension	256, 512, 1024
Batch size	64,128,256
Regularization	Dropout with 0.2, 0.3, 0.5 and Early stopping
Optimizer	Adam with a learning rate of 0.01, 0.001 and 0.0001
Epoch	100

Table 5.4: Training loss and time taken for selected embedding dimension and latent dimension

Embedding	**Hidden unit**	**Loss**	**Time take) (second)**
128	256	0.2052	14,153
128	512	0.2013	14,153
128	1024	0.2036	14,157
256	256	0.2201	14,156
256	512	0.2033	14,155
256	1024	0.2073	14,151

We chose the latent dimension and embedding dimension based on the minimal loss and quick training time based on the aforementioned findings. As a result, the outcomes are somewhat comparable, but the outcomes for the 512 hidden units and 128 embedding di-

mensions show minimal variation from one another. As a result, we chose latent dimension 512 and embedded dimension 128.

Next, we select the dropout rate to avoid overfitting. We have compared the model with dropout rates of 0.2, 0.3, and 0.5. In addition, we used the Earlystopping mechanism, which terminates the training if it does not improve the loss for three consecutive epochs for the model.

Table 5.5: Training loss for selecting dropout rate

Embedding	**Hidden unit**	**Dropout rate**	**Loss**
128	512	0.2	0.2052
128	512	0.3	0.3101
128	512	0.5	0.3182

Lastly, we select the learning rate. We started with large values like 0.1 and then, we tried with exponentially lower values: 0.01, 0.001, and 0.0001 respectively and its loss is 1.0182, 0.2013, and 0.2304 respectively. Finally, we got the best result with a learning rate of 0.001.

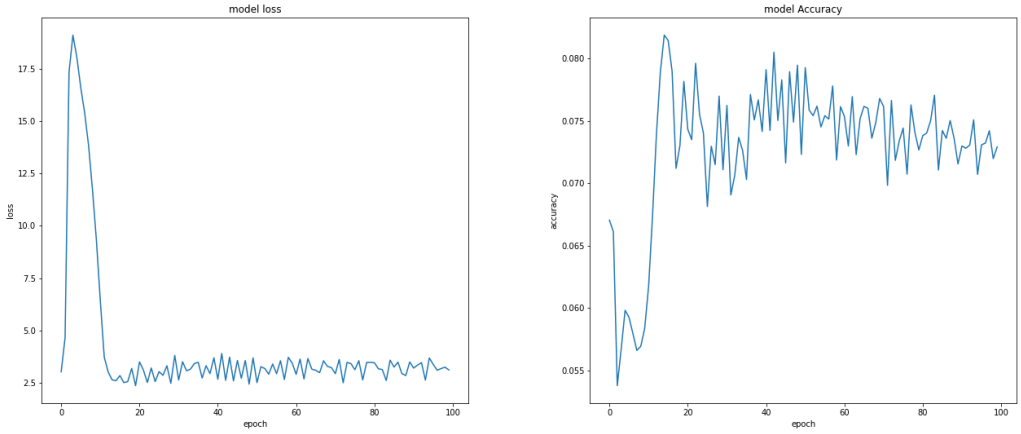


Figure 5.1: The loss and accuracy model by lr=0.1

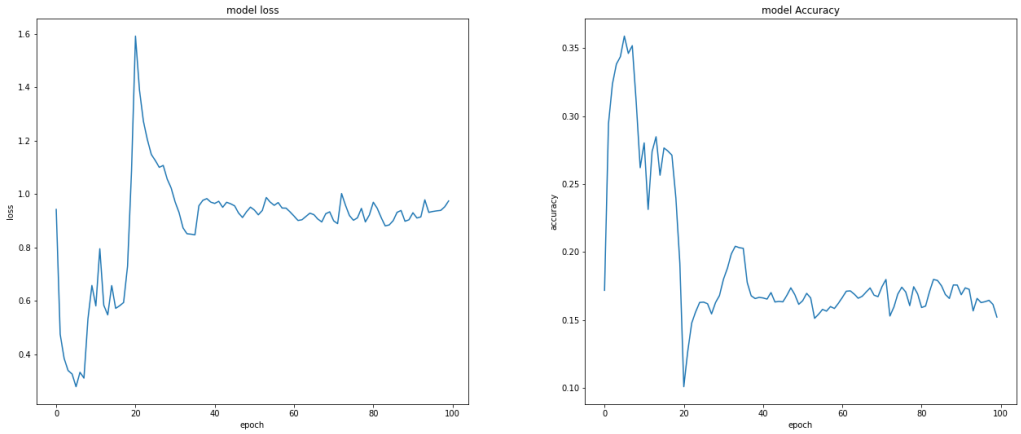


Figure 5.2: The loss and accuracy of the model by lr=0.01

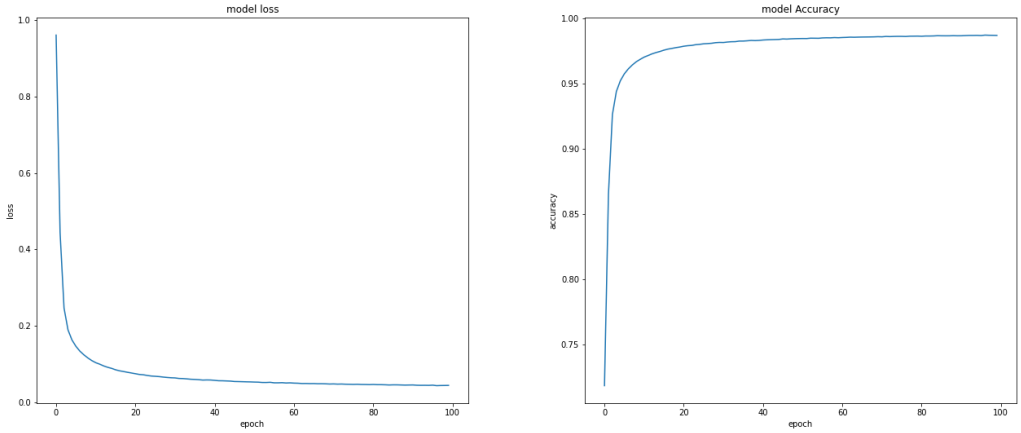


Figure 5.3: The loss and accuracy of the model by lr=0.001

5.5 Experiment result and evaluation

In this thesis, the performance and effectiveness of the spell checker are evaluated for two independent techniques that depend on the types of errors: isolated-word spell checker and context-based spell checker. The isolated-word spell checker detects and corrects only non-word spelling errors. Whereas, the context-based spell checker detects and corrects both non-word and real-word spelling errors in the Sidaamu Afoo text. For the isolated-word spelling errors detection and correction, we conducted two experiments:

- Detecting using a dictionary lookup and suggest correction using the Levenshtein edit distance.
- Detecting using a dictionary lookup and suggest correction using an encoder-decoder model.

We conducted three experiments for the context-based spelling(both non-word and real-word):

- N-gram language model with edit distance(N-gram model).
- Neural language model using LSTM for each RNN node with edit distance (LSTM model).
- Neural language model using LSTM for each RNN node (LSTM + attention model) with edit distance.

The performance of the spelling error detection and correction were evaluated for both isolated-word and context-based spelling errors. To evaluate the experiment, one of the typographic edit operations, such as insertion, deletion, substitution, or transposition error, is randomly added to the test words. The 5544 sentences were tokenized into words and amounted to 43,763 words. The typographic errors were added to those 43,763 test words and given to language experts to identify correct or incorrect words. Language experts manually counted and identified which words were correct and which were incorrect. Accordingly, a list of 8,290 misspelled words was provided. Among the 8290 misspelled words, there were 7961 non-words and 329 real-word errors. Depending on the information we get from the linguistic expert, we evaluate the performance of the detection and correction of our model. Both isolated and context-based spelling are evaluated independently.

5.5.1 Spelling detection experiment result

To evaluate the performance of the isolated-word and context-based spelling error detection, we used recall (both correct and incorrect), precision (both correct and incorrect), and accuracy, which are commonly used for evaluation purposes. These measures generally indicate how often spelling errors are rejected and how often spelling correctness is accepted The evaluation technique has four categories:

- **True positive(TP)**-The number of valid words recognized by the model.
- **True negative(TN)**- The numbers of invalid words recognized by the model.
- **False negative(FN)**- The number of valid words that are not recognized by the model.
- **False positive(FP)**- The number of invalid words that are not recognized by the model.

The performance of the proposed system is evaluated based on the above-explained conditions. Therefore, to compute the performance of the spell checker, the total number of TP, TN, FN and FP is counted from the test data with respect to the output of our model. The FP and FN are errors in the model. Sometimes FN happens due to the small size of the corpus, which doesn't include some of the testing vocabulary in the training data. FP will happen if the particular invalid word is included in the training data. So, FP will not happen in our model because our training data is error-free. On this basis, the recall (both correct and incorrect), precision (both correct and incorrect), and accuracy of the isolated-word and context-based spelling error detection are evaluated. These measures generally indicate how often spelling errors are rejected and how often spelling correctness is accepted. The following table shows the mathematical equation for each evaluation metric.

Table 5.6: Evaluation metrics

Metrics	**Equation**
Recall of correct (R_c)	$R_c = \frac{TP}{TP+FN}$
Recall of incorrect (R_i)	$R_i = \frac{TN}{TN+FP}$
Precision of correct (P_c)	$P_c = \frac{TP}{TP+FP}$
Precision of incorrect (P_i)	$P_i = \frac{TN}{TN+FN}$
Accuracy(A)	$A = \frac{TP+TN}{TP+TN+FP+FN}$

Experiment 1: Isolated-word spelling detection evaluation result

The isolated-word spelling focuses only on non-word spelling. The spelling errors were checked at the word level to detect non-word spelling errors through the comparison of dictionary lists based on similarity measurement of words without considering the context of the word. We evaluate the effectiveness of the dictionary lookup method by how much it correctly identifies the correct and misspelled words in the test data. From the 43763 tested words, 7961 words are non-word-spelling that were evaluated. The evaluation result is shown in table 5.7.

Table 5.7: Isolated word spelling detection result

Error	Approaches	**TP**	**TN**	**FN**	R_c**(%)**	R_i**(%)**	P_c**(%)**	P_i**(%)**	**A(%)**
Isolated-word (Non-word) detection	Dictionary lookup	32,763	7961	3039	91.51	100	100	72.37	93.06

Experiment 2: Context-based spelling detection evaluation result

Unlike isolated-word spell checkers, context-based spell checkers detect spelling errors by considering the context of the words. It focuses on both non-word errors and real-word errors. To evaluate the context-based spelling, the test set of sentences were pre-processed in the same manner as the training corpus. Each test sentence is divided into a list of n-gram sequences. For instance, the sentences *Sidaamu dagoomu qoqqowi qarru quchumi hawaasaati* is splitted in to different length of n-gram. The last word of each n-gram is the testing word, and the others are context words for the last word, as shown in table 5.8. All context words of an n-gram are given to the model to predict the next words. Depending on the predicted next words, the input word was identified as misspelled or not. If the input word is found in the predicted words, it is detected as correct; otherwise, it is misspelled. Each of the input words is evaluated. In this paper, we evaluate the effectiveness of the context-based spell checker that models correctly identifying the correct and incorrect words in the testing data. From the given 43763 tested words, 8290 words were errors(i.e., 7961 were non-word errors and 329 were real-word errors) that were evaluated. The result of context-based spell detection is shown in table 5.9

Table 5.8: n-gram list of sentences

bos sidaamu dagoomu qoqqowi qarru quchumi hawaasaati eos	
Left context	testing word
bos	sidaamu
bos sidaamu	dagoomu
bos sidaamu dagoomu	qoqqowi
bos sidaamu dagoomu qoqqowi	qarru
bos sidaamu dagoomu qoqqowi qaru	quchumi
bos sidaamu dagoomu qoqqowi qaru quchumi	hawaasaati
bos sidaamu dagoomu qoqqowi qaru quchumi hawaasaati	eos

Table 5.9: Context-based spelling detection result

Error	Approaches	**TP**	**TN**	**FN**	R_c**(%)**	R_i**(%)**	P_c**(%)**	P_i**(%)**	**A(**
Contex_based (Both non-word and real-word) detection	N-gram + ed	30,260	8290	5213	85.3	100	100	61.39	88.0
	LSTM + ed	30,564	8290	4909	86.16	100	100	62.8	88.7
	LSTM + attention + ed	30570	8290	4903	86.18	100	100	62.84	88.8

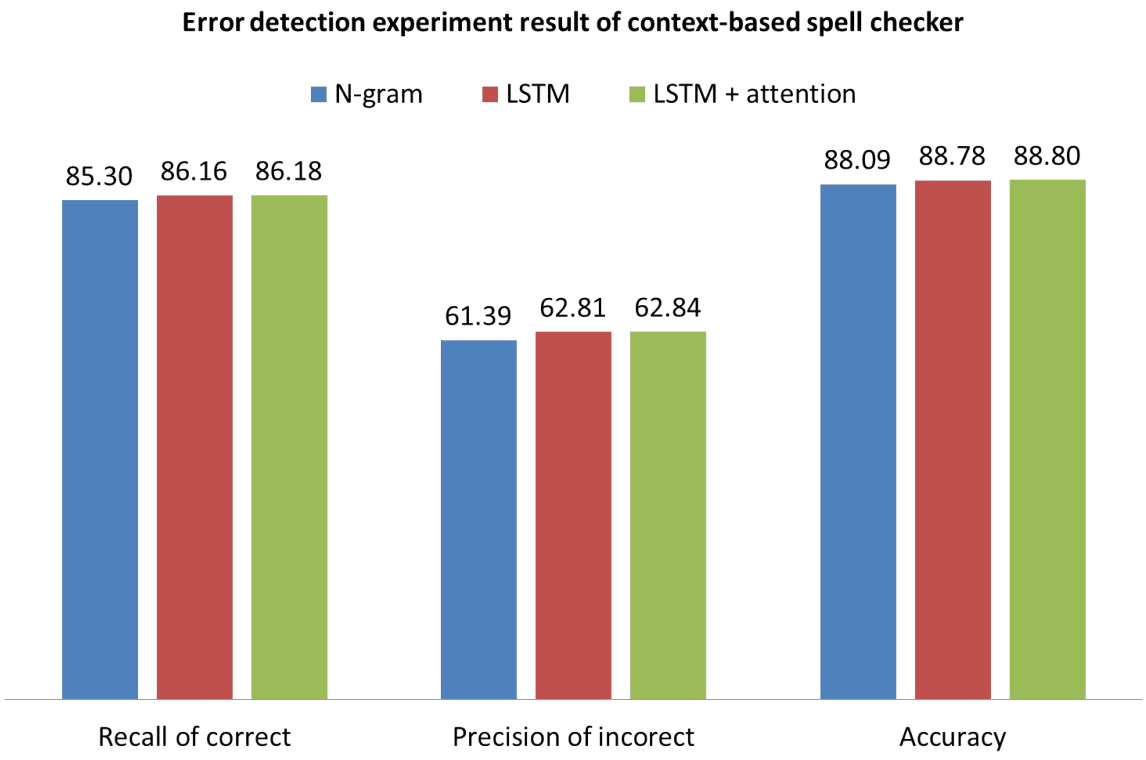


Figure 5.4: Context-based spelling detection evaluation result

5.5.2 Spelling correction experiment result

Spelling correction is evaluated in terms of the spell checker's ability to present the end-user with relevant and accurate suggestions for all true negatives (i.e., incorrect words flagged by the spelling checker). So, the error correction recall is used to evaluate the performance of the suggestions for the misspelled words. The error correction recall of a spelling checker should only be based on true negatives, and not on all negatives since the aim is to determine how well the spelling checker can suggest corrections for incorrect words. So, the error recall correction rate is an indication of the ratio of the total number of correct suggestions to the total number of errors detected in percentage terms [46]. Assume N_1 is the total number of the correctly detected error(TN), and N_2 is the total number of correct suggestions found for the detected misspelled words. so the precision of the correction is computed using equation 5.1

$$Errorcorrectionrecallrate = \frac{N_3}{N_1} = \frac{numberofcorrectsuggestions}{numberofmisspellsdetected} * 100 \qquad (5.1)$$

Each of the spell-correcting experiments, except the encoder-decoder model, were evaluated in the top one, top five and top ten suggestions. The top one matches suggest that the correctly spelled word for a particular misspelled word can be found in the first candidate suggestions provided by the model. The top five matches suggest that the correctly spelled word for a particular misspelled word can be found in the first five candidate suggestions (including the ones in the top one match) provided by the model. Similarly, the top ten matches suggest that the correctly spelled word for a particular misspelled word can be found in the first ten candidate suggestions (including the ones in the top one and top five matches) provided by the model. However, the encoder-decoder model were evaluated using the top one suggestion because it greedily predicts only the top one suggestion. We will recommend for for future researchers to use the beam search to increase the error correction recall. We compute the error correction recall for isolated-word and context-based spell correction independently.

Experiment 1: Isolated word error correction experiment result

For isolated-word spelling error correction, we conducted two different experiments: the minimum edit distance(ED) and the encoder-decoder model(Enc-dec). Once the non-word spelling errors were detected using the dictionary lookup method, the edit distance algorithm was used to provide suggestions for misspelled words by calculating the string similarity between the input word and the dictionary list. whereas the encoder-decoder model corrects the misspelled word by taking the vector of incorrect characters of the misspelled word and then predicting the corrected word. The isolated-word spelling correction experiments were evaluated only for the top one suggestion. The results of the isolated-word spelling correction are depicted in the below figure 5.5 and in the result in the table 5.10 and the error correction recall rate is in table 5.11

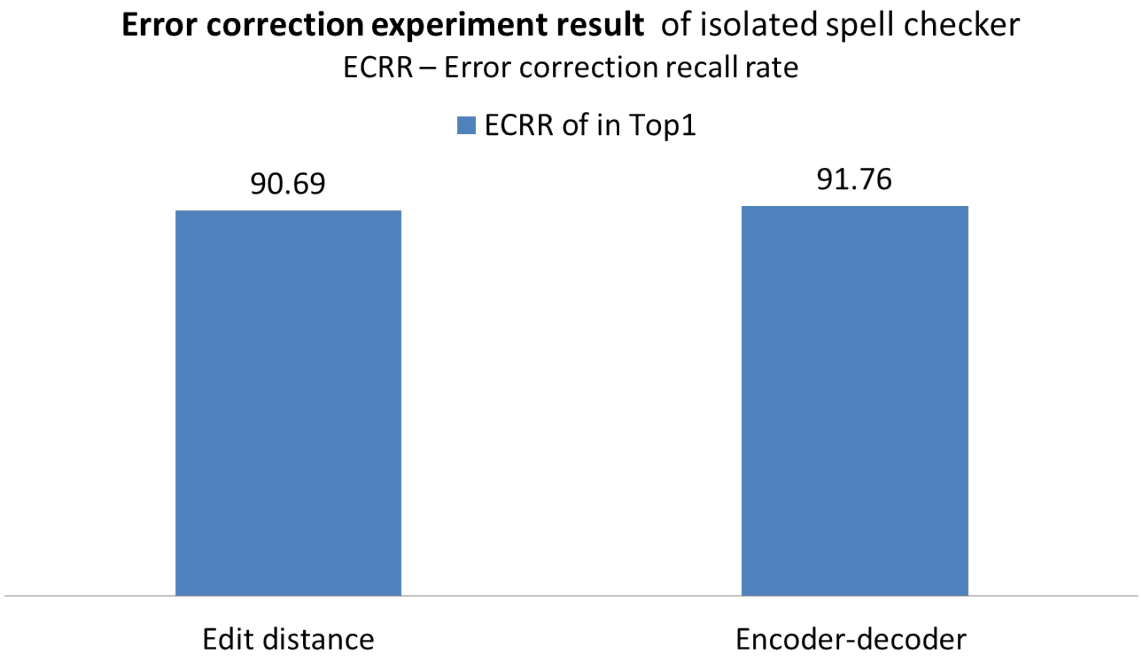


Figure 5.5: Isolated word error correction experiment result

Experiment 2: Context-based spelling correction experiment result

For the context-based spelling error(both non-word and real-word) the experiment of the n-gram, LSTM without and with attention were conducted. The evaluation is based on the provided top one, top five and top ten suggestions for each experiment. The total number of the model that provided the correct suggestion for the detected spelling is in the shown table 5.10 and the error correction recall rate is shown in table 5.11.

Table 5.10: Isolated and context-based error correction Evaluation results

Error type	**Models**	**Total TN**	**Sugestion**			
			Top1	Top 5	Top 10	Rest
Non word	ED	7961	7321	-	-	7961
	Enc-dec	7961	7305	-	-	7961
Context-based	N-gram + ed	8290	5014	5067	5071	8290
	LSTM + ed	8290	6125	6139	6142	8290
	LSTM+attention + ed	8290	6136	6152	6158	8290

Table 5.11 shows the error correction recall rate of the models. Figure 5.5 and 5.6 are show the performance of the non-word and contextual error correction for different models respec-

tively.

Table 5.11: Isolated and context-based error correction result

Error type	**Approaches**	**Top1 (in ECRR)**	**Top5 (in ECRR)**	**Top10 (in ECRR)**
Isolated (non-word)	Edit distance	90.69	-	
	Encoder-decoder	91.76	-	-
Context-based (non-word and real word) correction	N-gram + ed	60.48	61.12	61.17
	LSTM+ed	73.88	74.05	74.09
	LSTM+attention+ed	74.02	74.20	74.28

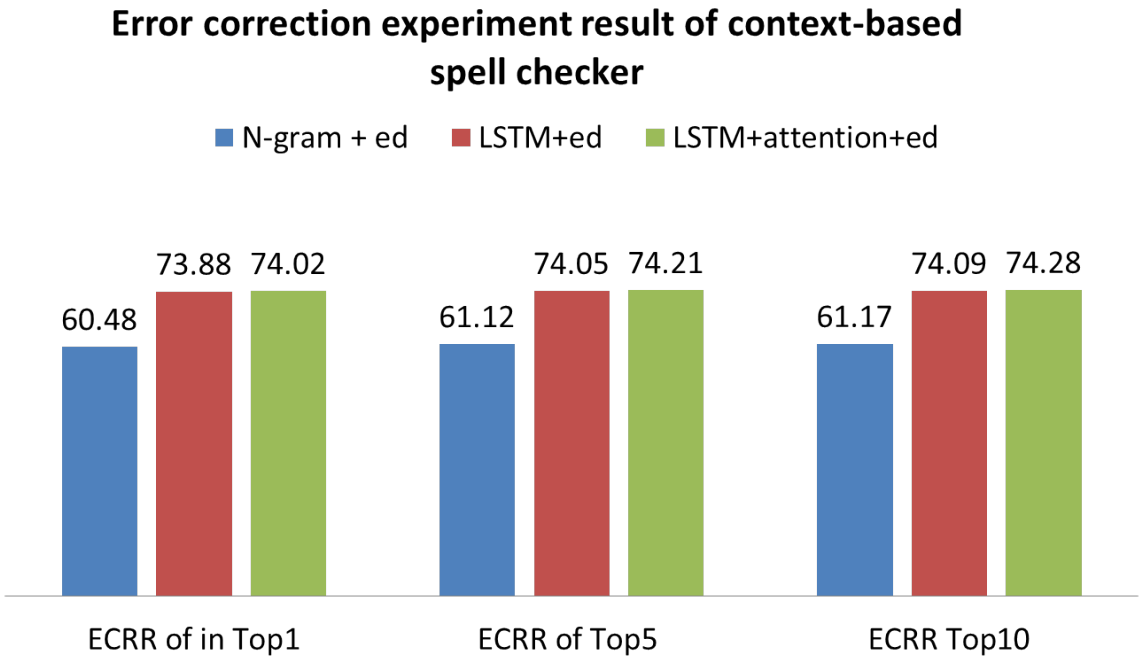


Figure 5.6: Context-based error correction experiment result

5.6 Result discussion

To achieve the goals of the research, different experiments were conducted, which were to detect non-word and real-word spelling errors and provide a correct suggestion for detected errors in Sidaamu Afoo's written text. Each experiment's detection and correction performance is evaluated using recall, precision, and accuracy. The isolated spell checker and context-based spell checker detection evaluation results of each experiment are shown

in tables 5.9 and 5.7, respectively. The correction evaluation result of each experiment is also depicted in tables 5.11. Based on the result provided, the best approach is selected for detection and correction.

For the isolated word, the dictionary lookup approach for detection and encoder-decoder model offered the best correction performance in contrast to the spell correction results of minimum edit distance result. The performance of the error correction recall rate of the encoder-decoder model and the edit distance algorithm was evaluated for isolated word correction. The encoder-decoder model achieves good results compared to the minimum edit distance approach. The minimum edit distance involves directly looking up input words in the dictionary, whereas the encoder-decoder model, on the other hand, calculates the probability of the character in the embedded vector rather than searching the input word in the dictionary. For a context-based spell checker, which detects both non-word and real-word spelling errors, the majority of the corrected suggestions for misspelled words are found in the first position (top 1). The LSTM model with attention achieved a better result for detecting and correcting contextual errors compared to other approaches. Here, the attention mechanism improves the performance of detection and correction because it focuses on the relevant information.

The experimental results of non-word detection show a value of 91.51% recall of correct, 100% recall of incorrect, 100% precision of correct, 72.37% precision of incorrect and 93.05% accuracy provide the dictionary lookup method. The error correction recall rate of the non-word spelling error using the Levenshtein edit distance and encoder-decoder model shows the related result of 91.76% and 90.69% respectively. However, the time that responses the suggestion is different. The encoder-decoder model provides the suggestion in the best time interval compared to the Levenshtein edit distance. So. the result of the experiment of the isolated word detection and correction has high accuracy in flagging words as valid/invalid and providing suggestions for detected non-word spelling errors. The meaning of the experimental result is, the system correctly detects the 7961 non-word spelling error correctly but 3039 words were detected incorrectly. So, the performance of the system's error detection **accuracy** scores 93.05%, which means the system can identify effectively and efficiently valid and invalid words in the given text. The **coverage of the correct word** in the test data is calculated by the **recall of correct** and it scores 91.51%. This shows that the dictionary may cover many correct words in the testing data and it performs well with

some limitations. We identified some reasons that minimize the performance of coverage of the correct word in the system. One of the identified reasons is Sidaamu Afoo's vocabulary coverage. This means some correctly spelled testing words are not found in the training data, so the model does not detect those valid words as valid. Thus, the model requires additional data to perform more results. Based on this, even if all words that exist in the language could not include, we can say the training texts used in the experiment were almost complete and covered all words used to test the system except some words. The **coverage of the incorrect word** is also calculated using the **recall of incorrect** which scores the 100%. This shows that all non-words errors in the given texts were perfectly detected.

The **precision** of correct and incorrect is also another metrics to measure the effectiveness of accepting and ignoring words in the spell checker. Precision is the about the exactness of the model. In the result, the valid words accepted and recognized by the spell checker score 100%. This means no invalid words are accepted as valid in the system. All the words that are recognized as valid words are valid words in Sidaamu Afoo and no invalid words are accepted as correct in the system. In **precision of incorrect**, from recognized words as incorrect, 72.37% were accepted. However, the system recognized and accepted some correct words as incorrect in the testing data.

For the detection and correction of context-based spelling errors, three experiments were conducted: n-gram with edit distance, and LSTM with and without attention with edit distance. The approaches used either the statistical language model(N-gram) or the neural language model(LSTM, LSTM+attention) are based on the statistics generated from the training corpus. The effectiveness of the provided correct suggestion is compares both systems on the top 1, top 5 and top 10 candidates. Most of the correct candidates are found in the top 1 candidates. This means the firstly predicted word is the correct suggestion for the detected error words. The LSTM model without attention and with attention with edit distance shows a good detection and correction result for the context-based spell checker for Sidaamu Afoo. Those approaches can retain the sequence information of the words in the memory cell and which has the semantic relation between the sequence of words. However, LSTM with attention can store longer contextual relevancies of words found in longer sentences of datasets where we compare it to the LSTM without attention. So, the LSTM with attention and edit distance performs good results for a context-based spell checker for Sidaamu Afoo.

Chapter Six

Conclusion and recommendation

This chapter finalizes the whole work that gives a general conclusion about the study and findings of the work, and suggests some future works for upcoming researchers.

6.1 Conclusion

Misspelled words, which are present in written text created by humans, are a common phenomenon in Sidaamu Afoo and these misspelled words can be classified as non-word errors or real-word errors. Sidaamu Afoo is the official working language of Sidama National Regional State. There is no spell checker that are tried yet. As a spell checker is a necessity for different NLP applications like machine translation, word prediction, name entity recognition, etc, developing a spell checker for Sidaamu Afoo is the paves the way for another researcher. So, we proposed an isolated-word spell checker which detects and corrects only non-word spelling error and a context-based spell checker for Sidaamu Afoo which detects and correct both non-word and real-word spelling error by considering the context of the words.

Different experiments were conducted to detect and correct the isolated word and context-based spelling and the results were recorded in this study. For the isolated-word spelling, two different experiments were conducted. The first experiment conducted to detect and correct non-word spelling error is the dictionary lookup and Levenshtein edit distance respectively. The second experiment is the dictionary lookup to detect and encoder-decoder model to correct for non-word error is used. To deal with the problem of context-based spelling, an experiment of the n-gram language model and neural language model using LSTM and LSTM with attention methods was conducted. As a result, this thesis work is done on a scarce

resource language with a data set of 49,896 sentences to train a model and 5544 sentences to test a model collected from data sources such as Sidaamu Afoo Holly bible, Bakkalcho newspaper, Sidaama national regional state constitution used for training and testing purpose. To get the desired result the data were preprocessed and the model is trained by the tuned parameter.

The system was evaluated by comparing the results found with manually annotated test cases by the language experts. Recall (correct and incorrect), precision(correct and incorrect) and accuracy are used to evaluate the performance of spelling detection. The performance of the dictionary lookup shows 91.51% recall of correct, 100% recall of incorrect, 100% precision of correct, 72.37% precision of incorrect and 93.05% accuracy. From the correctly detected words as invalid, the Levenshtein edit distance and encoder-decoder model were evaluated in terms of providing correct suggestions for invalid words using the precision of correction. The Levenshtein edit distance shows an error correction recall rate 90.69%, whereas the encoder-decoder model shows 91.76%. The experiments result of the context-based error detection, the n-gram language model with edit distance shows an accuracy of 88.09%, LSTM with edit distance shows an accuracy of 88.78% and LSTM with attention and with edit distance accuracy of 88.8%. To provide the correct suggestion for correctly detected spellings error were using n-gram, LSTM and LSTM with attention methods with edit distance that score the error correction recall rate of 60.48%, 73.88% and 74.02% for the top 1 candidates respectively.

Generally, we conclude that, for the isolated word correction method which detects and corrects only non-word spelling errors, the dictionary lookup approach to detect non-word and encoder-decoder model to correct is the optimal model for the Sidaamu Afoo. For the context-based spell checker for Sidaamu Afoo, LSTM with attention and with edit distance shows a better result in the case of addressing the problem of context-based spelling error than LSTM and the n-gram language model.

6.2 Recommendation

Though our proposed model performs well on the provided corpus, it needs further improvement to reach a more correction level. So we recommend the following for future work.

- Since our work is the first attempt to detect and correct non-word and real-word

spelling errors in Sidaamu written text, we face the challenges of memory failure when training the neural language model. As a result, we are forced to conduct our experiment using a small set of data. As we have noted from the above chapters, deep learning requires a huge amount of dataset for training. In contrast, the corpus taken for this study was not enough and cannot represent the languages. Hence, We recommend that anyone who is interested in improving the performance of our work should conduct experiments using the upgraded memory size(RAM greater than 12 GB) and include a higher number of datasets since neural networks need more datasets in order to perform better.

- When decoding the output, we have used the greedy algorithm for both the encoder-decoder model and the neural language model. The greedy search algorithm predicts the highest-scoring candidates in each step separately. To improve the performance of the correction, using beam search helps get accurate candidates, and it considers multiple best options based on beam width when predicting.

- Currently, the most advanced deep learning approach transformers achieve better results for various NLP applications and spell checkers as well. Anyone who wants to improve the performance of the Sidaamu Afoo spell checker can use a state-of-the-art transformer model.

- This study is focused on a single distance error of typographic error, we recommend future handling of multiple distance errors and other types of errors like cognitive and typographic errors.

- Finally, by using our system's output it is possible to develop another NLP application like a part-of-speech tagger, grammar checker, machine translation, question-answering, text-to-speech synthesis, speech-to-text synthesis, text summarization, dialogue system, name entity recognition, machine translation, part-of-speech and etc.

References

[1] James Allen. *Natural Language Understanding*. 2nd editio edition, 1995.

[2] R Kibble. *Introduction to natural language processing*. University of London, United Kingdom, 2013.

[3] Benjamin Van Durme. Robsut Wrod Reocginiton via Semi-Character Recurrent Neural Network . 2015.

[4] Bill Manaris. Natural Language Processing : A Human-Computer Interaction Perspective Natural Language Processing : A Human – Computer Interaction Perspective. *Advances in Computers*, 45, 2018.

[5] Shashank Singh and Shailendra Singh. HINDIA: a deep-learning-based model for spell-checking of Hindi language. *Neural Computing and Applications*, 33(8):3825–3840, 2021.

[6] Vibhakti V Bhaire, Ashiki A Jadhav, and Pradnya A Pashte. Spell checker. 5(4):5–7, 2015.

[7] Karen Kukich. Techniques for Automatically Correcting Words in Text. 24(4), 1992.

[8] Sumit Sharma and Swadha Gupta. A Correction Model for Real-word Errors. In *Procedia Computer Science*, volume 70, pages 99–106. Elsevier B.V., 2015.

[9] Ranjan Choudhury, Nabamita Deb, and Kishore Kashyap. *Context-Sensitive Spelling Checker for Assamese Language*, volume 740. Springer Singapore, 2019.

[10] Prabhakar Gupta. A context-sensitive real-time Spell Checker with language adaptability. pages 116–122, 2020.

[11] Manolito Octaviano Jr and Allan Borra. A Spell Checker for a Low-resourced and Morphologically Rich Language. pages 1853–1856, 2017.

[12] Suzan Verberne. Context-sensitive spell checking based on word trigram probabilities. Master's thesis, 2014.

[13] Ritika Mishra and Navjot Kaur. A Survey of Spelling Error Detection and Correction Techniques. 4:372–374, 2013.

[14] Bc. Jakub N´aplava. Natural Language Correction. *Charles university*, 2017.

[15] Shaona Ghosh and Per Ola Kristensson. Neural Networks for Text Correction and Completion in Keyboard Decoding. 14(8):1–14, 2017.

[16] Mir Noshin Jahan, Anik Sarker, Shubra Tanchangya, and Mohammad Abu Yousuf. Bangla real-word error detection and correction using bidirectional LSTM and bigram hybrid model. *Advances in Intelligent Systems and Computing*, 1309(January):3–13, 2021.

[17] Osman Büyük. Context-Dependent Sequence-to-Sequence Turkish Spelling Correction. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 19(4), 2020.

[18] Sunyan Gu and Fei Lang. A Chinese Text Corrector Based on Seq2Seq Model. *Proceedings - 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, CyberC 2017*, 2018-Janua:322–325, 2017.

[19] Chiao Wen Li, Jhih Jie Chen, and Jason S. Chang. Chinese spelling check based on neural machine translation. *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation, PACLIC 2018*, (December):367–375, 2018.

[20] Zijia Han, Chengguo Lv, Qiansheng Wang, and Guohong Fu. Chinese Spelling Check based on Sequence Labeling. *Proceedings of the 2019 International Conference on Asian Language Processing, IALP 2019*, pages 373–378, 2019.

[21] Damar Zaky and Ade Romadhony. An LSTM-based Spell Checker for Indonesian Text. *Proceedings - 2019 International Conference on Advanced Informatics: Concepts, Theory, and Applications, ICAICTA 2019*, pages 1–6, 2019.

[22] Ahmad Ahmadzade and Saber Malekzadeh. Spell Correction for Azerbaijani Language using Deep Neural Networks. (February), 2021.

[23] Gheith Abandah, Ashraf Suyyagh, and Mohammed Z. Khedher. Correcting Arabic Soft Spelling Mistakes using BiLSTM-based Machine Learning. *International Journal of Advanced Computer Science and Applications*, 13(5):621–626, 2022.

[24] Yasmin Moslem, Rejwanul Haque, and Andy Way. Arabisc: Context-Sensitive Neural Spelling Checker. *Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 11–19, 2020.

[25] Manar Alkhatib, Azza Abdel Monem, and Khaled Shaalan. Deep learning for Arabic error detection and correction. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 19(5), 2020.

[26] Michael Flor and Yoko Futagi. On using context for automatic correction of non-word misspellings in student essays. *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications, BEA 2012 at the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2012*, pages 105–115, 2012.

[27] Andrew Carlson and Ian Fette. Memory-based context-sensitive spelling correction at web scale. *Proceedings - 6th International Conference on Machine Learning and Applications, ICMLA 2007*, pages 166–171, 2007.

[28] Michal Richter, Pavel Straňák, and Alexandr Rosen. Korektor – A System for Contextual Spell-Checking and Diacritics Completion. *Proceedings of COLING 2012: Posters*, (December):1019–1028, 2012.

[29] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent Advances in Recurrent Neural Networks. (December 2017), 2017.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[31] Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. pages 1–9, 2015.

[32] Kazuhiro Kawachi. A GRAMMAR OF SIDAAMA (SIDAMO), A CUSHITIC LANGUAGE OF ETHIOPIA. 2007.

[33] Rebecca Treiman and Brett Kessler. Writing Systems and Spelling Development. *The Science of Reading: A Handbook*, (October):120–134, 2008.

[34] Victor Kuperman, Amalia Bar-On, Raymond Bertram, Rober Boshra, Avital Deutsch, Aki Juhani Kyröläinen, Barbara Mathiopoulou, Gaisha Oralova, and Athanassios Protopapas. Prevalence of Spelling Errors Affects Reading Behavior Across Languages. *Journal of Experimental Psychology: General*, 150(10):1974–1993, 2021.

[35] Anbessa Tefera. *A Grammar of Sidaama*. PhD thesis, Hebrew university of Jerusalem, 2000.

[36] ERIC MAYS, FRED J. DAMERAU, and ROBERT L. MERCER. Context based spelling correction. *Context based spelling correction*, 1991.

[37] Andargachew Mekonnen. Development of an Amharic Spelling Corrector for Tolerant-Retrieval. pages 22–26, 2012.

[38] Andargachew Mekonnen Gezmu, Andreas Nürnberger, and Binyam Ephrem Seyoum. Portable spelling corrector for a less-resourced language: Amharic. *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, pages 4127–4132, 2019.

[39] Gaddisa Olani Ganfure and Dida Midekso. Design And Implementation Of Morphology Based Spell Checker. *International Journal of Scientific & Technology Research*, 3(12):118–125, 2014.

[40] Tirate Kumera. Context Based Spellchecker for Afan Oromo Writing. Master's thesis, 2018.

[41] Atakilti Brhanu Kiros and Petros Ukbagergis Aray. Tigrigna language spellchecker and correction system for mobile phone devices. *International Journal of Electrical and Computer Engineering (IJECE)*, 11(3):2307–2314, 2021.

[42] Sébastien L'Haire. FipsOrtho: A spell checker for learners of French. *ReCALL*, 19(2):137–161, 2007.

[43] Sadidul Islam, Mst Farhana Sarkar, Towhid Hussain, Md Mehedi Hasan, Dewan Md Farid, and Swakkhar Shatabda. Bangla Sentence Correction Using Deep Neural Network Based Sequence to Sequence Learning. *2018 21st International Conference of Computer and Information Technology, ICCIT 2018*, (1):1–6, 2019.

[44] Steven M Ross and Gary R Morrison. EXPERIMENTAL RESEARCH METHODS. (May), 2014.

[45] Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. Automatic spelling correction for resource-scarce languages using deep learning. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Student Research Workshop*, pages 146–152, 2018.

[46] Hsuan Lorraine Liang. Spell Checkers and Correctors: a Unified Treatment. (November):119, 2008.

[47] Aadil Ahmad Lawaye and Prof Bipul Syam. Design and Implementation of Spell Checker for Kashmiri Design and Implementation of Spell Checker for Kashmiri. (December), 2017.

[48] Nigusu Yitayal. context based spell checker for Amhric. Master's thesis, Jimma university, 2016.

[49] Sabri A Mahmoud. Context-Sensitive Arabic Spell Checker Using Context Words and N-Gram Language Models Models. (August 2016), 2013.

[50] Mark Sheridan Nonghuloo and Karthik Krishnamurthi. Spell Checker for Khasi Language. *International Journal of Software Engineering*, 8(1):1–12, 2017.

[51] Amita Jain and Minni Jain. Detection and correction of non word spelling errors in Hindi language. *2014 International Conference on Data Mining and Intelligent Computing, ICDMIC 2014*, 2014.

[52] William F. Gilreath. Hash sort: A linear time complexity multiple-dimensional sort algorithm. (September 2004), 2004.

[53] Wubetu Barud Demilie. Multilingual spelling checker for selected ethiopian languages. *International Journal of Advanced Science and Technology*, 29(7):2641–2648, 2020.

[54] Abduljebar Kedir. Designing Dictionary Based Spelling Checker for Afaan Oromoo. Master's thesis, 2019.

[55] Youssef Bassil and Mohammad Alwani. Context-sensitive Spelling Correction Using Google Web 1T 5-Gram Information. *Computer and Information Science*, 5(3):37–48, 2012.

[56] Muhammad Maulana Yulianto and Riza Arifudin. Autocomplete and Spell Checking Levenshtein Distance Algorithm to Getting Text Suggest Error Data Searching in Library. 5(1):67–75, 2018.

[57] Naushad Uzzaman. A Bangla Phonetic Encoding for Better Spelling Suggestion. (January), 2004.

[58] P. Brown, J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, R. Mercer, and P. Roossin. A Statistical Approach to French/English Translation. *Biological and Artificial Intelligence Systems*, 16(2):547–561, 1988.

[59] Jay M. Ponte and W. Bruce Croft. Language modeling approach to information retrieval. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, pages 275–281, 1998.

[60] Jing Bai and Jian-Yun Nie. Using Language Models for Text Classification. *Proceedings of the Asia Information Retrieval Symposium, Beijing, China*, (1):1–6, 2004.

[61] Christopher D.Manning. Speech and Language Processing: An introduction to natural language processing. *SPEECH and LANGUAGE PROCESSING An Introduction to Natural Language Processing Computational Linguistics and Speech Recognition*, pages 1–18, 2021.

[62] ABDULLAH CAN ALGAN. PREDICTION OF WORDS IN TURKISH SENTENCES BY LSTM-BASED LANGUAGE MODELING. Master's thesis, 2021.

[63] Pascal Vincent. A Neural Probabilistic Language Model. 3:1137–1155, 2003.

[64] Si Lhoussain Aouragh, Abdellah Yousfi, Saida Laaroussi, Hicham Gueddah, and Mohammed Nejja. A new estimate of the n-gram language model. *Procedia CIRP*, 189(June):211–215, 2021.

[65] Lidia Mangu and Eric Brill. Automatic Rule Acquisition for Spelling Correction.

[66] Joshua T Goodman. A Bit of Progress in Language Modeling Extended Version. 2008.

[67] Ciprian Chelba and Mohammad Norouzi. N-gram Language Modeling using Recurrent Neural Network Estimation. (March), 2017.

[68] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, pages 1–12, 2013.

[69] Shweta Kadam. Neural Network Part1: Inside a Single Neuron. https://medium.com/analytics-vidhya/neural-network-part1-inside-a-single-neuron-fee5e44f1e, 2020.

[70] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[71] Forecasting stock prices with long-short term memory neural network based on attention mechanism. *PLoS ONE*, 15(1):1–15, 2020.

[72] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 1724–1734, 2014.

[73] Linhong Weng, Qing Li, and Xuehai Ding. Recurrent convolutional Neural Network with Initialized Filters for Text Classification. 15(2):75–85.

[74] Mariawit Shimelis. Amharic Spelling Error Detection and Correction System : Amharic Spelling Error Detection and Correction System : Morphology-based approach. Master's thesis, Addis Ababa University, 2020.

[75] Naveen Sankaran and C. V. Jawahar. Error detection in highly inflectional languages. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, (July):1135–1139, 2013.

[76] Kenneth W. Church and William A. Gale. Enhanced Good-Turing and Cat-Cal. (May):82, 1989.

[77] E. J. Yannakoudakis and D. Fawthrop. The rules of spelling errors. *Information Processing and Management*, 19(2):87–99, 1983.

[78] Michael Allen Bickel. Automatic correction to misspelled names: A fourth-generation language approach. *Communications of the ACM*, 30(3):224–228, 1987.

[79] Mark D. Kernighan, Kenneth W. Church, and William A. Gale. A spelling correction program based on a noisy channel model. pages 205–210, 1990.

[80] Andrew R. Golding. A Bayesian hybrid method for context-sensitive spelling correction. 1996.

[81] Michael P. Jones and James H. Martin. Contextual spelling correction using latent semantic analysis. pages 166–173, 1997.

[82] Yi Hsien Wang and I. Chen Wu. Grammatical and context-sensitive error correction using a statistical machine translation framework. *Software - Practice and Experience*, 39(7):701–736, 2009.

[83] Zsolt Bitvai and Trevor Cohn. Non-linear text regression with a deep convolutional neural network. *ACL-IJCNLP 2015 - 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, Proceedings of the Conference*, 2:180–185, 2015.

[84] S. Sooraj, K. Manjusha, M. Anand Kumar, and K. P. Soman. Deep learning based spell checker for Malayalam language. *Journal of Intelligent and Fuzzy Systems*, 34(3):1427–1434, 2018.

[85] Xiangci Li, Hairong Liu, and Liang Huang. Context-aware stand-alone neural spelling correction. *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, pages 407–414, 2020.

[86] Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. NeuSpell: A Neural Spelling Correction Toolkit. pages 158–164, 2020.

[87] Zhiwei Wang, Hui Liu, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zitao Liu. Learning multi-level dependencies for robust word recognition. *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pages 9250–9257, 2020.

[88] Charana Sonnadara, Surangika Ranathunga, and Sanath Jayasena. Sinhala Spell Correction - A Novel Benchmark with Neural Spell Correction Sinhala Spell Correction A Novel Benchmark with Neural. (September), 2021.

[89] Hao Li, Yang Wang, Xinyu Liu, Zhichao Sheng, and Si Wei. Spelling Error Correction Using a Nested RNN Model and Pseudo Training Data. 2008.

[90] Yingbo Zhou, Utkarsh Porwal, and Roberto Konow. Spelling correction as a foreign language. *CEUR Workshop Proceedings*, 2410, 2019.

[91] Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. Combating adversarial misspellings with robust word recognition. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 5582–5591, 2020.

[92] Desta Legese Lalego. Development of Morphological Analyzer for Sidaamu Afoo. Master's thesis, Addis Ababa University, 2020.

[93] Matoosi W.goorgisi. Afuunna afuullote Bitima II, 2011.

[94] Samuel Balayneh and Taaffasa G.Maarami. Sidaamu Afoo borreessate budi aana mangistete loosaasinera uynanni qajeelshu maanuwwaaale:training manual, 2017.

Appendix One

Sample code

A.1 Parallel corpus preparation

```
def add_speling_erors(token, error_rate):
    """Simulate some artificial spelling mistakes."""
    assert(0.0 <= error_rate < 1.0)
    if len(token) < 3:
        return token
    rand = np.random.rand()
    # Here are 4 different ways spelling mistakes can occur,
    # each of which has different chance based on Sidaamu afoo Writer occured.
    prob = error_rate / 4.0
    if rand < prob:
        # Replace a character with a random character.
        random_char_index = np.random.randint(len(token))
        token = token[:random_char_index] + np.random.choice(CHARS) \
                + token[random_char_index + 1:]
    elif prob < rand < prob * 2:
        # Delete a character.
        random_char_index = np.random.randint(len(token))
        token = token[:random_char_index] + token[random_char_index + 1:]
    elif prob * 2 < rand < prob * 3:
        # Add a random character.
        random_char_index = np.random.randint(len(token))
        token = token[:random_char_index] + np.random.choice(CHARS) \
                + token[random_char_index:]
    elif prob * 3 < rand < prob * 4:
        # Transpose 2 characters.
        random_char_index = np.random.randint(len(token) - 1)
        token = token[:random_char_index]  + token[random_char_index + 1] \
                + token[random_char_index] + token[random_char_index + 2:]
    else:
        # No spelling errors.
        pass
    return token
```

Figure A.1: Generated spelling error in correct word

```
import json
from functions import load_book,clean_text,cleantext,tokenize
```

Reading a data

```
def readData(split_length):
    #Read a training data
    with open('./train_test/trainData'+str(split_length)+'.json', 'r') as fp3:
        trainD = json.load( fp3)

    #Read a testing data
    with open('./train_test/testData'+str(split_length)+'.json', 'r') as fp3:
        testD=json.load( fp3)

    return trainD, testD
train_sent, _ = readData(10)
```

Creating a sequence(ngrams)

```
def n_gram_split(sent, split_length):
    input_sequences = []            # words before the current word/label
    for line in sent:
        token_list = tokenize(line)
        # if(len(token_list) >= 2 and len(token_list)<=18):
        for i in range(1, len(token_list)):
              # print(token_list[:i+1])
              n_gram_sequence = token_list[:i+1]
              line = ' '.join(n_gram_sequence)
              input_sequences.append(line)
    print("The total number of n-grams is:",len(input_sequences))
    with open('./train_test/ngrams'+str(split_length)+'.json', 'w') as fp:
        json.dump(input_sequences, fp)
    del input_sequences
```

Encoding splitting Data it into x and y

```
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.utils import to_categorical
from keras.preprocessing.sequence import pad_sequences
import pickle
import numpy as np
def encoding_data(split_length):
    # Read training n-grams and store in seqs
    with open('./train_test/ngrams'+str(split_length)+'.json', 'r') as fp:
        seqs = json.load( fp)
    # Split training ngram in to 6 parts i.e. len(seqs)//6
    seq_splitted=split_list(seqs, wanted_parts=6)
    seq=seq_splitted[0]
    # Total number of the n-grams[0] after splitting
    print("Total number of the n-grams[0] after splitting:",len(seq))

    # Tokenize into words and convert into sequences
    tokenizer = Tokenizer(filters='!#$%&()*+,-./:;<=>?@[\]^_`{|}~')
    tokenizer.fit_on_texts(seq)
    sequences = tokenizer.texts_to_sequences(seq)
    max_sequence_len=max([len(sent) for sent in sequences]) + 1
    print("The total unique words are:",len(tokenizer.word_counts))
    print("Max length:",max_sequence_len)

    # Pad sequences
    sequences=np.array(pad_sequences(sequences, maxlen=max_sequence_len, padding='pr
    vocab=len(tokenizer.word_counts)+1

    # Split each n-gram in to X and Y
    data_x=sequences[:,:-1]
    data_y=sequences[:,-1]

    # one-hot encoding
    data_y = to_categorical(data_y, num_classes=vocab)
    words_to_index = tokenizer.word_index

    # Save
    with open('./train_test/ngramtokenizer'+str(split_length)+'.pickle', 'wb') as ha
        pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
    # with open('./ngramsforlstm/wordIndex.json', 'w') as fp:
    #     json.dump(words_to_index, fp)
    del seqs,seq_splitted,seq
    return data_x,data_y,vocab,words_to_index
```

```
def  spellcorrection(text,length):
    import tensorflow as tf
    tf.compat.v1.logging.set_verbosity(tf.compat.v1.logging.ERROR)
    from  tensorflow.keras.preprocessing.sequence  import  pad_sequences
    from  tensorflow.keras.models  import  load_model
    from  tensorflow.keras.optimizers  import  Adam
    import pickle
    import time
    import csv

    with  open('./train_test/ngramtokenizer.pickle',  'rb')  as  handle:
        tokenizer19  =  pickle.load(handle)

    file="./model/lstmatt128.hdf5"
    model19  =  load_model(file  ,custom_objects={'attention':  attention})
    model19.compile(loss='categorical_crossentropy',   optimizer=Adam(learning_rate=0.

    start=  time.time()
    ngrams  =  generate_ngrams(text)
    for  ngram  in  ngrams:
        print("current  n-gram  is:",ngram)
        current_word  =  ngram[0][-1]
        if len(ngram[0]) >= 2 and correct != 1 and len(suggestions) != 0:
            seed_text_ltr = " ".join(word for word in ngram[0][:-2]) + " " + suggest
        else:
          seed_text_ltr = " ".join(word for word in ngram[0][:-1])
        seed_text_rtl = " ".join(word for word in ngram[1])
        print("seed  text  is:",seed_text_ltr)
        print(seed_text_ltr+"  "+'\033[1;4m'+current_word+'\033[0m'+"  "+  seed_text_rt
        tokenized=tokenize(seed_text_ltr)

        line  =  ' '.join(tokenized)

        encoded_text  =  tokenizer19.texts_to_sequences([line])
        pad_encoded  =  pad_sequences(encoded_text,  maxlen=20,  truncating='pre')
        candidates=[]
        suggestions = []
        correct = None
        for  i  in  (model19.predict(pad_encoded)[0]).argsort()[-5:][::-1]:
            candidates.append(tokenizer19.index_word[i])
        print("corrected  suggestion:",candidates)
        for prob in candidates:
            ed  =  nltk.edit_distance(current_word,  prob)
            #  print("corrected  suggestion:",pred_word)
            if ed ==0:
                correct = 1
                break
```

```
            elif len(current_word)<=3 and ed ==1:
                suggestions.append((ed, current_word, prob))
            elif len(current_word)>3 and ed <=2:
                suggestions.append((ed, current_word, prob))
            else:
                continue
        if correct == 1:
          print("CORRECT")
          # s.append("correct")
        elif correct != 1 and len(suggestions) > 0:
            correct = 0
            print("Suggestions:", " - ".join([suggest[2] for suggest in suggestions]
            # s.append(" ".join([suggest[2] for suggest in suggestions]))
            print("WRONG")
        elif correct != 1 and len(suggestions) == 0:
            print("No suggestion!")
            # s.append("No")
        print("-------------------------------------------------------------------
        print('Time taken: ',time.time()-start)
        # words.append(current_word)

```

Appendix Two

Sample data

B.1 Sample training text

aarooni tenne kakkalo baala shiqishe angasi mannu aana diriirse maassi'rihu gedensaanni kakkallanni darginni dirri. gattinori goddate giddora e'anno. ikkollana yohaannisi kunni manchi daafira coyi'rinori baalu halaaleho yitu. kunino xaa geeshsha owaante aanni afamanno. hakkunni gedensaanni daawiti kaaliiqa mare yihudu katamma giddonni mitto amadoni. hatte yannara aasa maganosi kaaliiqa kaaliiqa'ya. aaroonino angasi gotti asse siqqotenni baattote aana noo buko gani. lamunku kaaliiqi albaanni mimmitoho ledde gondooro e'u. afe kae hagiidhino yiinsa. konni kaa ani olantote gargarto albaanni sae massagate gaanni hanafe olu xawira gaadeemmo. tenne gobba baxa magano giwa ikkitinota diegentinoonni. isi qole kaaliiqi kulinoere coyi'ra dihasiissannoe. taamaari rodiise abeseloomi mine ofolte dadillitanni callase heedhu. galte afi'rinokki meentino ane hexxona. hakku barri tunsichoho ikkinnina xawaabba di"ikkinona konne barra mayira hasidhinanni. lowo diri gatinosiha ikkirono gatinosi diri deerrinni roorse gama woxe qolona. kunnira dancha guma laalannokki haqqa baala karre giirate tunganni yiinsa. ikkollana so'risiisannori daanno gede doogo fananno-hura aanne'e. insa asale qola dandiinokkiha beebbaataamonna koattete waaga baata dandiinokkiha buxicho mancho borojjimmate hirtino. wodanchanna dancha coye assa agurino. baabilooni barri muleeti. hakkunni gedensaanni ani ermiyaasi rekaabootaho israeelete magani olantote mootichi kaaliiqi anni'ne iyonaadaabi uyino'ne hajajo wonshitinoonni. yesusi hakkiinni kae ha'ranni hee'reenna lamu illiweelooti hoodesi hadhu. maganinke kaaliiqi baalanka angankera sayise uyinonke. mannaho aanno ba'raa'rino angasi no yitine kulle yee lallawi. wolqaataame gosa asseemmohe. saaooli yinoonnita baxinokki daafira lowo geeshsha hanqi. daawiti teesso ikkoottatera kaaliiqira kakkallanni dargira ate yerusaalamete

aanne'e. galteno minaannase ayirrisso. hakku cubbi ajeennaatini konne leddinihu. suudisi baalu baxisannoho. israeelete roorrooti kiiro albitewiinni xe'inokki xuube seekkite shiqisha hasiissannonsata buuxxuhu gedensaanni qarru amadinonsata affu. ki'ne hatte yannara bayisibao ba'nummo yitinanni. kuneeti lowo geeshsha batidhino baonna beebba gaangeessite nooe. meyati kalaqantinohu labbaahuwiinniiti ikkinnina labbaahu meyatenni dikalaqamino. waa hinkii'litara dagganno beetto waa hayikkisie yeennaatino agi. kuni fushshaati wo'manka woyite gudinse giirranniho. insano naaziretete yesuusa hasi'neemmo yitu. yesuusi kayinni insa widira caaqqi yee lae ki'ne yerusaalamete meento umi'neranna ooso'nera wi'le ikkinnina anera wi'litinoonte. kunnira ani israeelete magani kaaliiqi ki'ne aana noommo. hexxote qaalikki garinni wodancha uyie. mitte lamala woma olantote qoteho dirama dandaaya roorenkanni gobbaya baxeemmo gede assitinoe. baarigaarakki yoote darga garatto yine borreessinoonninte gede mannu baalunku kaphaancho ikkirono maganu kayinni halaalaanchoho. shilqote kakkalo tashshi diassitannohe. barri'ya qeelameenna waammano qixxaabbinoe. mitto barra yesuusi callisi huucci'ranni noowa rosaanosi isiwa daggeenna yesuusi mannu ane ayeti yaanno. hatto assitineenna ani qole waalcho fane dancha attote batise aa hoogummo'nero tennenni affe"e. ikkollana wodani'ya seeri ledo gaaramannohu mannimma'ya widoonni cubbu seeri usuranya assannoehu wolu seeri mannimma'ya giddo noota afoommo. isi gosate ane yoo kulanno. raaheelira soqqantannose gede biliha yinanni borojjicho uyise.

B.2 Sample parallel corpus

- **Target and Misspelled lists**

no nwo	xa'motenni xa'otenni	agaraasine agaraasike
kunni kuxni	uduunne uauunne	lallabbanno laliabbanno
ati ti	baala balaa	yii yfi
fultuhunni fulturunni	iillisheemmosi iililsheemmosi	iyyaarkora iyyarkora
diro dior		heedheenna ehedheenna
aganira agayira	minisi pminisi	kaaliiqi kaliiqi
dukkanu dukkgnu	ilaalaame ilaalfame	manna lmanna

yiqaamiya yfiqaamiya	ilaalu liaalu	hedeemmo hdeemmo
naqqasiro naqqsairo	iimira imira	woyite wyite
yohaannisiwa ylhaannisiwa	hatte xatte	wodani'ya owdani'ya
marte amrte	rufannonsa urfannonsa	madi'ranno madi'cranno
yesuusi esuusi	ayyaani ayyabani	wayi wayvi
mulira muira	halchitini halchiitni	noo ndo
	hasidhe hagidhe	

B.3 Sample output

```
['maganu', "aanno'nena", "ki'neno", 'uuyye']

Input sentence:   omaganu anano'nena ki'nneo uyye
Decoded sentence: maganu aanno'nena ki'neno uuyye
Target sentence:  maganu aanno'nena ki'neno uuyye
==================================================
['baxillunniha', 'gondoorokkino', 'hegere', 'geeshsha', 'disoorratto']

Input sentence:   baxxllunniha gondooroskkino hegre geeshmha disoorratto
Decoded sentence: baxxullunniha gondoorokkino hegere geeshsha disoorratto
Target sentence:  baxillunniha gondoorokkino hegere geeshsha disoorratto
==================================================
```

Figure B.1: Sample encoder-decoder output

B.4 Sample spelling survey

Sidaamu Dagoomu Qoqqow Mootimma Hodh/na Doogote Latishshi Biiro

የሲዳማ ብሔራዊ ክልላዊ መንግስት ትራንስፖርትና መንገድ ልማት ቢሮ

Ta. [illegible]

Barra [illegible]

Sidaamu Qoqowi Polisete Komishinera

Hawaasa

Hajo:- Covdete Yannara Dagate Hodhishira Fushinooni Lalawa la,ano

Covde daganke qawaitanokki gede Ethipiyu deerinni daagate hodhishi kaamala hogobano manu kiiro la'anohunni Fajinonni manni kiiro aleenni hagobe leelituro Qarichinshaani womaashi deero La'anohunni gobbate darinni lalawu fujinoti qaangannite. konni lalawi garinni mittu dagate hodhishi kaameeli hogowara fajinonnihu aleenni hogowe leeliro mittu manchinni 1000/ mito kume/qorichishamano gede lalawu giddo woronni wodhoti ikolanna kayinni gobate deerinni lalabonni lalawa hakawaro worroricho /zonnata/ dirnnsi woyite sidaamuno zoonete deerira noodaafira wodidi qoqowi hakawaro sidaamahono miteenni dirisino kkonni daafira gobate deerinino, ikko wodidi qoqowira Covd lalawi kaino ikkirono Sidaamu qoqowi giddo xawqeesha lalawunni woronni qorichi dikaino Gobate deerinni lalawa kayinse sidaamu Gobanni wolu qoqowira Covdte korikatinni dagate Hodhishira fungonni lalawa kayinse alibaanni loonsanni garinni loosanni no. Konni garinni Covdete yanara woronni lalawi wolu qoqowira kaenna nikke qoqowira kainkki daafira xaageesha kaameela holonni lalawininna wodhonni qorichishantanni noota qoqowinke hodhishu kaameela mamara kulitanni keeshitino.

Konni daafira covde yannora fulo lalawu garinni mitto mancho quwasayise hagowinohura 1000/ mitta kumi/ qorichisha seenmale ikkitino daafira Tinni borro iltu barinni kayinse alibaanni fulitino hodhishu wodho garinni qorichishanni gede xawinseemo.

Keeru Ledo

[redacted]

Affino

- 36 Quchumatenna/ woradate hodh/borroninnira
- Bironke doogote Keranchira Dayirekitoretera
- Dagate Hodhishi Mamari Ta/Dr/Ku/ka/Bu/Dayirektorete /
- Birrote Sooreesira
- Biriote Layiki Sooreesirano Hodhishu Handori sooreesira

Hawaasa


AwashBank
Muli Barra
Looso Hannfemo!


Adiventisitete Rosi Mininni Kayllsse Taboori Mekani Yesuusi
Huucatote Mine - Hatono Yeshi Hoteele Doogo Widooni Hawaassu
Yuniverisite Geeshsha noo koilsho Doogo Hanafisilsate Oixaawo
ከአድቬንቲስት ተ/ቤት - በታቦር መካነ ኢየሱስ-የሺ ሆቴል
መንገድ እስከ ሀዋሳ ዩኒቨርሲቲ ድርስ የሚሰራ የአስፋልት
መንገድ ፕሮጀክት ማስጀመሪያ መርሃ ግብር


Darasnynya aatenna adhatemni
ደረሰኝ በመስጠትና በመቀበል
Sirbaamtu Dagoomu Qoqqoomi Maalimma
Hawassi quchumi E/B/mine
የኮተማችንን ለማት እናፋጥን!!
Quchumanikke latinisonna!!