SCHOOL OF GRADUATE STUDIES

SENTENCE-LEVEL GRAMMAR CHECKER FOR KAMBAATISSA LANGUAGE USING DEEP LEARNING APPROACH

MSC THESIS

TIHUN SEIFU

DECEMBER, 2023

WOLKITE, ETHIOPIA

Wolkite University

School of Graduate Studies

Sentence-Level Grammar Checker for Kambaatissa Language Using Deep Learning Approach

A Thesis Submitted to School of Graduate Studies, in partial Fulfilment of the Requirements for the Degree of Master of Computer Science and Engineering in Computer Science

Tihun Seifu

Major Advisor: Mesfin Abebe (Ph.D.)

Co-Advisor: Abdo Ababor (MSc.)

December, 2023

Wolkite, Ethiopia

APPROVAL SHEET

School of Graduate Studies

Wolkite University

SENTENCE-LEVEL GRAMMAR CHECKER FOR KAMBAATISSA LANGUAGE USING DEEP LEARNING APPROACH

Submitted by:

Tihun Seifu		31/12/2023
Name of Student	Signature	Date

Approved by:

Name of Major Advisor	Signature	Date
Mesfin Abebe (PhD)		01/01/2024
Name of CO-Advisor	Signature	Date
Abdo Ababor (MSc)		3/1/12/2023
Department Head	Signature	Date
Kuma Yadi (MSc)		09/01/2024
Name of Chairman, DGC	Signature	Date
Name of Dean, SGS	Signature	Date

Dean, Graduate Studies

Wolkite University School of Graduate Studies

WOLKITE UNIVERSITY

SCHOOL OF GRADUATE STUDIES

We hereby certify that we have read and evaluated this Thesis titled "**Sentence-level grammar checker for Kambaatissa language using deep learning approach**" prepared under our guidance by Tihun Seifu. We recommend that the Thesis shall be submitted as fulfilling the requirements for the award of a MSc. Degree in Computer Science and Engineering.

Name of Major Advisor	Signature	Date
Mesfin Abebe (PhD)		01/01/2024
Name of CO-Advisor	Signature	Date
Abdo Ababor (MSc)		31/12/2023

As members of the Board of Examiners of the Master of Science Thesis open defense examination, we have read and evaluated this Thesis prepared by Tihun Seifu and examined the candidate. We hereby certify that, the thesis is accepted for fulfilling the requirements for the award of the degree of Masters of Science (MSc) in Computer Science and Engineering.

Dr. Kindie Biredagn		Dec-31-2023
Name of the external examiner	Signature	Date
Bermuda Yilleta		31/12/2023
Name of the internal examiner	Signature	Date
Kuma Yadi		1/1/2024
Name of the chairman	Signature	Date

Final approval and acceptance of the thesis are 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).

DEDICATION

To my elder brother (Dr. Demelash Seifu) who passed away suddenly, to my dear father (Seifu Abamo) who taught me to reach the stage I have now and to my beloved family.

BIOGRAPHICAL SKETCH

My name is Tihun Seifu. I was born in a very rural area. However, my growth is in Hossana city in the central region of Ethiopia. Incidentally, I was born being with a disability. Being born with disability was a shocking event for my family and everyone around us. I am favourite child in my family. Even though I was born with disability, I can do everything. Being disabled doesn't stop me from doing anything. In my family, my father is the strongest, so he raised me in a strange way without me knowing that I was disabled. On this occasion, I would like to thank my father, who is my pride next to my God, because he is the reason why I have reached this stage. Although I was born being with disability, my father was very strong, so when I was old enough to go to school, he took me to the city and sent me to school. I was amazingly talented in both my thinking and my studies that I used to get various awards from my school. Then I completed my high school in 2017 G.C and joined Wolkite University in 2018 G.C and studied software engineering for four years and graduated in 2021 G.C. My time at university was a very interesting, fun, amazing time. However, there were times that challenged me the most. However, Wolkite University was special for me during my stay. The goodness that Wolkite University has done for me did not end with encouraging me. In 2014 E.C, I benefited of the scholarship offered from University for my master's degree. So I completed my research work in Computer Science and submitted it to the department. Now I am as happy as ever because my disability has not stopped me from doing anything and I have a lot of self-respect and I love my life.

ይህ አባባል የውስጤን ይገልፅልኛል ➔ " ላያስችል አይሰጥም"

DECLARATION

By my signature below, I declare and affirm that this thesis is my own work. I have followed all ethical principles of scholarship in the preparation, data collection, data analysis and completion of this thesis. All scholarly matter that is included in the thesis has been given recognition through citation. I affirm that I have cited and referenced all sources used in this document. Every serious effort has been made to avoid any plagiarism in the preparation of this thesis.

This thesis is submitted in partial fulfillment of the requirement for a degree from the School of Graduate Studies at Wolkite University. The thesis is deposited in the Wolkite University library and is made available to borrowers under the rules of the library. I solemnly declare that this thesis has not been submitted to any other institution anywhere for the award of any academic degree, diploma or certificate.

Brief quotations from this Thesis may be used without special permission provides that accurate and complete acknowledgement of the source is made. Requests or permission for extended quotation from or reproduction of this thesis in whole or in part may be granted by the head of school or department or dean of the school of graduate studies when in his/her judgment the proposed use of the material is in the interest of scholarship. In all other instance, however, permission must be obtain from the author of the thesis.

Name: Tihun Seifu Abamo Signature: 

Date: 31/12/2023

School/Department: Computer Science and Engineering

ACKNOWLEDGMENTS

First of all, I am eternally grateful to Almighty God for helping me in my life. I would like to thank Wolkite University from my heart for providing me with a free scholarship to study for a master's degree. Next, I would like to express my deep sincere gratitude to my major advisor Mesfin Abebe (Ph.D.) and Co-Advisor Mr. Abdo Ababor (MSc.), for giving their endless support, supervision, and valuable comments. I would like to thank Hossana College of Teachers Education Kambaatissa linguistic department head Mr. Asmako Yohannis (MA) and Mr. Tekle Abebe (MA). I would like to thank Wachamo University Durame campus Kambaatissa linguistic department head Mr. W/Kidan Arficho (MA) and other persons who stand for me for Kambaatissa linguistic-related concepts and corpus preparation. I would like to thank my best friends, who have been helping me by sharing their valuable ideas. I would like to express my deepest love and respect to my families, who have been by my side with their endless support and encouragement during my studies. I would especially like to thank my dear father (Seifu Abamo).

Name: Tihun Seifu

Signature: 

Date: 31/12/2023

ABBREVIATIONS AND ACRONYMS

AND	Adjective Noun Disagreement
ANN	Artificial Neural Networks
AVD	Adverb Verb Disagreement
BERT	Bidirectional Encoder Representations from Transformers
Bi-GRU	Bidirectional Gated Recurrent Units
Bi-LSTM	Bidirectional Long-Term Memory
CFG	Context-Free Grammar
CNN	Convolutional Neural Network
CRF	Conditional Random Fields
DNN	Deep Neural Networks
FNN	Feed-forward Neural Networks
GRU	Gated Recurrent Units
LSTM	Long Short-Term Memory
LSTM-CRF	Long Short-Term Memory Network-Conditional Random Fields
NL	Natural Language
NLP	Natural Language Processing
OSV	Object Subject Verb
POS	Part of Speech
QALB	Qatar Arabic Language Bank
RNN	Recurrent Neural Network
SOV	Subject Object Verb
SVD	Subject Verb Disagreement

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF TABLES IN THE APPENDIX

LIST OF FIGURES IN THE APPENDIX

LIST OF ALGORITHMS

ABSTRACT

In the modern world, the most basic and culturally accepted means of communication is language, and the use of grammar is crucial to language fluency. Finding grammatical errors in natural language processing applications involves checking whether the words in a sentence conform to the predefined grammar rules for number, gender, tense, and the necessary information to convey the information in written language. Incorrect input sentences can have agreement problems, such as subject-verb, adjective-noun, and adverb-verb agreement problems. This study developed a sentence-level grammar checker for the Kambaatissa language using a deep learning approach. In particular, we focus on implementations of gating methods such as the LSTM class and the more recently proposed GRU. For the development of the proposed model, Python programming languages and packages were used. Among the packages, the TensorFlow and Keras packages can effectively perform grammar error checking of the proposed model. GRU and LSTM test cases were used for evaluation. Finally, the test results show that the LSTM accuracy is 83% recall, 83% precision, 83%, f1_score is 83% and kappa score is 78%. The GRU performed 83% accuracy, 83% of recall, 83% precision and 83% f1_score and kappa score is 77%. The main challenge of this study was the rich and complex morphology of the Kambaatissa language and to find a sufficient amount of Kambaatissa sentence. The results of this study help to advance the Kambaatissa language's grammar checking technology. For writers, students, and language learners seeking to ensure that written text is grammatically correct and consistent, advanced grammar proofreading is an invaluable resource. Future research directions include expanding the coverage of the grammar checker to handle more complex grammatical constructions and integrating it with textual support models for wider usability.

Keywords: deep learning, grammar checker, GRU, Kambaatissa language, LSTM, sentence-level

CHAPTER ONE

1. INTRODUCTION

1.1. Background of the Study

Kambaatissa language, spoken by the Kambaata people, belongs to the broad Afro-Asiatic family and is a highland eastern Cushitic language, of which Tembaro, Alaba and K'abeena are dialects [1]. Kambaatissa language speakers are spoken in the highlands of the Hambarrichcho massif, which is situated between the Omo and Billate rivers, approximately 300 kilometres southwest of the capital city of Ethiopia, Addis Ababa. The area of Kambaata can be reached via Hosanna (Hadiyya) in the north or via Alaaba Kuliito (Alaaba) in the east [2]. The name Kambaata occurs in various spellings in the literature, the most common ones being Kambata, Kambatta, Kembata, Cambata, and Cambatta. The Kambaata people refer to their language by the term Kambaatissata, which includes the derivational formative -issata for names of languages, or as Kambaati afoo [lit. "the mouth of Kambaata"]. When using the word "Kambaata," it is possible to refer to the ethnic group as well as the nation of Kambaata, so it is often used to refer to the language in linguistic literature [2].

Natural language processing, in its simplest form, is the capacity for a computer or other system to actually comprehend human language and process it in a manner similar to that of a human [3], [4]. Grammar checking is one of the methods most frequently used in applications involving natural language processing. Grammar checkers examine the morphological and syntactic processing of sentences to determine the grammatical structure of sentences. The goal of detecting grammar errors simply entails using a computer to locate grammaticalerrors in human writing, classify those errors, and either repair or eliminate those flaws. It is quite helpful and may be used to automatically check writing by language learners, proofread writers' content, and fix grammar in everyday writing scenarios, among other things. Because of its widespread use, grammar check has gotten a lot of attention from academics both at home and abroad [5]. Itis appropriate to introduce the Kambaatissa language more through deep learning methods using technology. Because the more the machine understands the language, the easier it is for the user to fix grammar errors. Therefore, we proposed using deep learning techniques

called Long Short Term Memory (LSTM) and Gated Recurrent Units (GRU) to check the grammar errors in Kambaatissa sentences.

1.2. Motivation of the Study

Human language is the basis for communicating information. Learning a mother tongue includes learning about the history and culture of parents, relatives, and previous and later generations because language is integrated with both. The Kambaatissa language is a unique and distinct language with its own grammar rules, language structures and error patterns. However, it is clear that there are challenges for individuals or writers, students and language learners writing in Kambaatissa in the context of grammar checking. So for Kambaatissa language, the researcher proposed to do grammar error checking at sentence level. Another is that grammar plays an important role in ensuring clarity, consistency and accuracy in written content. A grammar checker designed specifically for the Kambaatissa language is very important to solve language-specific conflicts and provide corrections for errors, as no grammar checker has been and developed in the Kambaatissa language before. Our motivation for this study is to use LSTM and GRU from deep learning techniques to develop a sentence-level grammatical error checker for the Kambaatissa language. Therefore, the researcher wants to adapt the Kambaatissa language to the computer and teach the machine to make the language approachable to technology. This is not only to help native Kambaatissa speakers, students and writers who are learning the language to understand Kambaatissa grammatical errors and to produce correct grammatically correct sentences to communicate better, but also to preserve, standardize and develop the Kambaatissa language.

1.3. Statement of the Problem

One of the zonal languages spoken in central Ethiopia is Kambaatissa. It is estimated that over a million people currently speak it. Currently, Wachamo University on the Durame campus, Hosanna Teachers College, and preparatory, secondary, and primary schools all use the language as a medium of instruction. The number of Kambaata educational, cultural, religious, magazine articles and other documents in electronic media is expanding quickly as more people have access to the Internet. As a result, grammatical errors are often observed when the language user does these things. Therefore, grammatical errors made by the user could cause the generated text's content to change or

remain meaningless. Meanwhile, the absence of any other computer-aided research on the Kambaatissa language, especially on grammatical error checking, creates a number of challenges for individuals, students, and language users who write in Kambaatissa and strive to achieve grammatical accuracy in their written communication. It also hinders their ability to spot and correct grammatical errors, resulting in grammatical errors, inconsistencies, and written content with unclear structures. This can hinder effective communication, hinder language learning and limit the overall development and level of the Kambaatissa language.

There are different grammar checkers are developed for different languages in the Ethiopian and foreign languages. The most famous Ethiopian grammar checkers are Afan Oromo Grammar Checker, designed by [6], and Amharic Grammar checkers [7], [8]. Also, some foreign languages, such as English grammar checker [5], [9], [10], Arabic grammar checker [11], and so on, have confirmed their own grammar. Although all the mentioned languages have their own grammar checkers, the Kambaatissa language needs a grammar checker because it has its own sentence structure, word formation, and writing styles. Since, there haven't been any published studies or academic research specifically focused on developing or evaluating grammar checkers for Kambaatissa language. This suggests that there is a gap in the existing literature and research regarding this area. So, the researcher wants to conduct research in this area using deep learning approaches that can effectively check grammatical errors.

Research Questions

The researcher aims to address the following research questions in this study:

- **RQ1**. What kinds of errors do we make in grammar when writing in the Kambaatissa language?
- **RQ2**. Which deep learning algorithms are the best for checking grammar in Kambaatissa Language?

1.4. Objective of the Study

1.4.1. General Objective

The general objective of this study is to develop a sentence-level grammar checker for the Kambaatissa language using a deep learning approach.

1.4.2. Specific Objectives

In order to achieve the general objective, this research has the following specific objectives:

- To review literature on grammatical structure in other languages for a better understanding of grammar error checking.
- To collect a representative corpus of written texts in Kambaatissa for training data and evaluation purposes.
- To design a grammar error checker for the Kambaatissa language.
- To design a deep learning architecture suitable for Kambaatissa language.
- To train the deep learning model using the collected Kambaatissa corpus.
- To develop a model for representing the Kambaatissa grammar.
- To evaluate the grammar errors checker's performance using data from the Kambaatissa language test data.

1.5. Scope and Limitation of the Study

The scope of this study is to focus on grammar errors in the Kambaatissa language, especially subject-verb disagreement, adjective-noun disagreement, and adverb-verb disagreement, but the analysis of spelling errors and stylistic errors is not within the scope of this study.

1.6. Significance of the Study

For Kambaatissa language speakers, students, and others, the following significance of the grammar error checker is hoped for:

- Whoever speaks the language can quickly and easily find the information they need with the help of the Kambaatissa grammar error checker.

- Improves the quality of Kambatisa language texts by ensuring that sentences are grammatically correct.
- To improve Kambaatissa language writers' skills, especially the non-natives, to prepare error-free documents like letters, emails, various messages, and others.
- Primary school students, high school students, and college-level Kambaatissa language learners may find it very beneficial as it allows them to become more independent and complete their work.
- By providing a dedicated grammar checker for the Kambaatissa language, the study contributes to the development and standardization of the language.

1.7. Organization of the Study

The study consists of six chapters, and the first chapter include the background of the study, motivation of the study, statement of the problem, objectives, scope and limitations, and significance of the study. In Chapter Two, the literature review and related works are explained in detail. Methods and approaches are presented in Chapter Three with proper explanations. We cover design and implementation in Chapter Four. Chapter Five also include results and discussions. The final chapter include a conclusion and recommendations for further research.

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Introduction

This chapter presents grammatical concepts and theoretical concepts related to the Kambatisa language and grammar checker in a clear way. An overview of the Kambaatissa language's nature is provided at the outset of the chapter's content. The morphological features of the Kambaatissa language, its parts of speech, its grammatical structure, and its grammatical errors are then discussed in detail in the Kambaatissa language section. Furthermore, the various approaches of grammar checking—from rule-based to deep learning—are explained in detail in this chapter. Lastly, in this study, some related grammatical error-checking works in local and foreign languages are discussed in detail.

2.2. The Origin of Kambaatissa Language

Kambaatissa is a member of the Highland Eastern branch of the Cushitic language family. The area of Kambaata can be reached from Hosanna (Hadiyya) in the north or in the east through Alaaba Kuliito (Alaaba). Kambaata's neighbours are speakers of closely related languages, namely Alaaba to the east, the Hadiyya to the north and southeast, and Xambaaro to the west (in Omo Shalaqo wäräda in the KX zone). The Wolaitta and Dawro, who speak Omotic languages in the south and southwest, are Kambaata's neighbours [2]. Bilingualism is common in Kambaata. Standard educated speakers speak Amharic fluently. Hadiyya knowledge is widely seen among the previous generation. In addition to being the name of a language, Kambaatissa is also the name of a minor branch of the Kambaata group that also includes Alaaba, Qabeena, Xambaaro, and Kambaata (in the narrow meaning). As far as we know today, Xambaaro is a slightly different dialect of Kambaata (Korhonen et al. 1986, Treis forthcoming a). There is little difference between Kambaata and Xambaaro and they do not cause any problems to communicate between the two groups [2].

Kambaatissa language is closely related to the Sidama and Hadiya languages spoken in the neighboring regions [12]–[14]. The name Kambaata is found in various spellings in the literature, the most common being Kambata, Kambatta, Kembata, Cambata and

Cambatta [2]. The term "Kambaatissata," used by the Kambaata people to refer to their language, is the origin of the language's name, Kambaati afoo ("Mouth of the Kambaata") [2]. The word Kambaata can be used to refer to both the ethnic group and the Kambaata country [2].

The Kambaata people are an ethnic group primarily residing in the Kambaata zone of the central Ethiopia. They are part of the larger Cushitic-speaking Oromo ethnic group. The Kambaata people have their own distinct language, also called Kambaata, which [15] belongs to the Afro-Asiatic language family. The have a rich cultural heritage, including traditional music, dance and clothing. The Kambaata people are primarily engaged in agriculture, with crops such as maize, teff, and barley being important to their livelihoods. A Kambaatissa-Amharic dictionary has been published by the Kambaata-Xambaaro Zone Information and Culture Main Department (1995 E.C. / 2005) [16].

2.3. The Writing System of Kambaatissa Language

According to the practical observation and experience of linguists and researchers, the official writing system currently taught in primary and secondary schools in the zone, as well as in Hossana College of Teachers Education and Wachamo University on the Durame campus, is based on Latin, but there is some deviation from the International Phonetic Alphabet convention. For example, the word Kambaata is written as Kambaata, and the double letters "aa" indicate its length [17].

As of right now, publications in Kambaatissa languages are mostly found in the educational and religious spheres. The researcher found out how many Kambaatissa speakers in primary and junior high school are limited to reading and writing because Amharic is the primary official written language in both national and regional workplaces. Today, some literary works, newspapers, magazines, education resources, official documents, and religious writings are written and published in Kambaatissata [1].

2.3.1. Kambaatissa Alphabets

Kambaata orthography is based on Latin. Some consonant phonemes [18] are written with more than one sequence of characters: “”, “s”, “y” and “h”. Kambaata has 28 consonants and five short vowels: “a”, “i”, “e”, “u”, “o” and their long counterparts "“aa”, “ii”, “ee”, “uu” and “oo”.

Table 1. Latin alphabet in Kambaatissa language Capital letters with Small letters

Aa	Ba	Cc	CH, ch	Dd	Ee	Ff	Gg	Hh
Ii	Jj	Kk	Ll	Mm	Nn	NY, ny	Oo	Pp
PH, ph	Qq	Rr	Ss	SH, sh	Tt	TS, ts	Uu	Vv
Ww	Xx	Yy	Zz	ZH, zh				

Vowels

The vowels in the Kambaatissa orthography are five. Double vowels show the length of the vowel. Bi-phonemic, or combinations of two short vowels, is types of long vowels.

“a”, “i”, “e”, “u …….. Single vowels
“aa”, “ii”, “ee”, “uu …… double/long vowels

Vowel phonemes come in the middle and end of words after any consonant. Only after the approximant "w" neutralized the phonemic opposition between the short "o" and "a" [19].

Consonants

There are 28 consonant graphemes in the Kambaata writing system. The corresponding graphs are presented in Table 2.1. Unindicted graphs in pointed brackets are identical to the IPA standard in terms of phonemic and orthographic representation [1].

2.4. Kambaatissa Language Morphological Features

Kambaatissa language is a morphologically rich, strongly suffixed and strictly head-ending language. Four open word classes can be defined based on morphological and syntactic features: nouns, verbs, adjectives, and ideophones. The word class of adjectives can be assumed to constitute a sub-class of a word class of attributes. In addition to adjectives, the class of adjectives contains two other case and gender subclasses, namely numerals and demonstratives. The last category does not concern us here. Nouns are primarily used as units and occur as heads of NPs in this function. As an NP head, a noun can govern modifiers, i.e. adjectives, demonstratives, numerals, and relative clauses that

generally precede a noun. In addition to expressing predicate arguments, nouns can function as predicates themselves (in conjunction with copulas) and serve as pronouns. Kambaatissa nouns are obligatorily specified for case and gender [2].

Most feminine nouns have an additional morpheme -ta, some masculine nouns and (optionally) an additional morpheme -ha in the cases, which also receive a citation form; let's take an example. For ease of identification, the first inflectional morpheme after a noun stem is called the "case/gender morpheme", the subsequent morpheme (e.g. -ta) is called the "secondary case/gender morpheme" [2].

Kambaatissa verbs indicate aspect (imperfective, e-/o-perfective and progressive), mood (indicative, imperative-justice and preventive), subordination (main vs. subordinate verb), polarity (positive vs. negative) and subject agreement (person, gender, number and social status of the subject are expressed by portmanteau morphemes on the verb). Kambaatissa verbs can be ordered in order of infinitives to reduce the chances of inflection. Verbs in main clauses have full inflectional possibilities; they are fully finite and, in principle, the only types of verbs that can complete a sentence. The verbal stem is followed by inflectional morphemes (between double bars) and, optionally, by an object suffix [2].

The existence of adjectives for some Cushitic languages is disputed. However, it is reasonable to argue in this paper that Kambaatissa has verb-adjective hybrids if it can be shown that the language has an adjective that has a separate vocabulary from the verb and noun. The arguments presented in Treis are summarized in the following. Adjectives are used as modifiers of nouns in NP [2]. Like verbs, adjectives are modified by degree adverbials. Adjectives can also govern accusative and dative complements by describing something that has a certain quality. In summary, adjectives cannot be included in nouns or verbs. Although they share certain properties with nouns and verbs, what sets them apart from other parts of speech and defines adjectives as a separate part of speech in Kambaatissa is their unique agreement pattern.

2.5. Part of Speech of Kambaatissa

Assigning a tag to a sentence that explains how each word is used in the sentence is known as part-of-speech (POS) tagging. The most typical POS includes nouns, verbs, pronouns, adjectives, adverbs, conjunctions, conjunctions, interjections, and so on. The

POS tagger tries to assign the appropriate POS tag to each word in a sentence, taking into account the context in which the word is found.

Noun

Nouns are required for both gender and case. They can be divided into characteristic case suffixes based on the accusative case form's final vowel [2].

Table 2. Kambaatissa Nouns with meaning

Word (Nouns)	**Meaning**
Adabaa	Boy
Boora	Ox
Sa’a	Cow
Mini	House
Woqqaa	Road
Haqqa	Tree
Meseleeta	Girl
Sulumuta	Heifer
Giiráta	Fire
Uullata	Land
Hixita	Grass

Verb

Word class members for aspect [2] (main distinction: imperfect and perfect), mood (indicative, imperative and defensive) and subordination (main and subordinate verbs), as well as person, gender, number, and social status of the subject in portmanteau morphemes. The main difference between verb forms is between perfective and imperfective examples. The degree of completion varies from predicate to predicate and

is shown, among others; by the predicate high or low person distinction (e.g. most main verbs identify seven persons but only five) [2].

Table 3. Kambaatissa verbs with meaning

Word (Verb)	**Meaning**	**Type of verb**
Hujat	work	
Hujataamm	I will work.	imperfective main verb
Hujatachcheemm	I worked.	a-perfective main verb
Hujachchoomm	I worked.	o-perfective main verb
Hujatan(i)	… (I) working ….	imperfective SS converb
Hujatchch(i)	… (I) having worked ….	perfective SS converb
Hujataniyan	… (I) working ….	imperfective DS converb
Hujachchiyan	… (I) having worked ….	perfective DS converb

Adverbs

Kambaatissa has a few undefined adverbs to express verbal relations. Only leelan, dango, biinin, taabba'idda, ammoo, xa'ita xa'ita, tada, and tema cannot be considered members of main word like verb, noun, and adjective or be considered derived from them. Therefore, these few words form a very small separate, semantically diverse and closed class of words. Adverbs with the copula (VV–t) can be used in prepositions, e.g. temaant 'it's in a minute' [2].

Table 4. Kambaatissa adverbs with meaning

Word (Adverbs)	Meaning
Leelano	gently, gradually
Dangon	abruptly, unpredictably
Biini'e	independently
taabba'iddat	alternatively
Ammo	nevertheless, still
xa'ita xa'ita	still
Tadda	now, at this moment
Temma	now, in a split second

Adjectives

There is a widespread reluctance to use the word adjective in the description of Eastern Cushitic languages but the word is not generally rejected. Argues for Lowland East Cushitic Somali that lexical items denoting qualities are composed of different POS, but the adjective denoting one of the POS is omitted. Interestingly, in the same volume on the same language, claims that Somali has word class onsets, contradicting previous descriptions of them being analyzed as verbs or as a subset of verbs. Kambaatissa is close relatives in the texts written on the Eastern Cushitic languages of the Highlands are more unclear. Neutral adjectives are not limited to areas where they can be understood from the context [2].

Table 5. Kambaatissa adjectives with meaning

Adjective	Meaning	Derived Noun	Meaning
abba(-ta) being	big	abbimata	big, size
biilla(-ta)	easy, light	biillimata	lightness, easiness; embarrassment
biiza(-ta)	generous, kind	biizimata	generosity, kindness
bonqa(-ta)	nice (of enset food)	bonqimata	high quality (of enset food)
busha(-ta)	defiled, dirt-cheap, bad	bushimata	badness, indecent behavior
buxa(-ta)	poor	buximata	poorness, poverty
diiha(-ta)	empty	diihimata	Emptiness
fayya(-ta)	healthy	fayyimata	Health
haraara(-ta)	wide	haraarima-ta	wideness, width
hifata(-ta)	lazy	hifatimata	Laziness
hiyyeessa(-ta)	orphan(ed)	hiyyeessimata	being an orphan
kallu(-ta)	naked	kallimata	nakedness, nudity
laafa(-ta)	soft	laafimata	Softness

Conjunctions

When a conjunction is used to connect words, phrases, and sentences in a conventional meaning, it can stand alone and be morphologically invariant, Kambaatissa has only a few proper conjunctions, namely the adverbial conjunction “te” => “or” and the adverbial conjunction “bagaan” => “but”. Both words occur in multiple places in the sentence [2].

Pronoun

A class pronoun is a closed word class with three subtypes: personal pronouns (simple and reduplicated pronouns), interrogative pronouns, and descriptive pronouns. Personal pronouns (mainly [+ person]) refer to nouns and stand in place. Interrogative and demonstrative pronouns can refer to human or non-human referents, states of affairs, ideas, and speech acts. Pronouns are marked for case, gender, and number. Not all pronouns are independent words in Kambaatissa: possessive and object pronouns are suffixes to nouns and verbs respectively [2].

2.6. Grammatical Structure of Kambaatissa

A sentence is a group of words or phrases that express a complete thought or idea or provide all the necessary information. A sentence must have a subject and a verb or predicate. A Kambaatissa sentence might be an order, a question, an exclamation, or a statement. The predicate indicates what the subject does or contains the sentence's object and verb. The subject of the sentence can be a noun phrase that refers to a person or thing. Kambaatissa language has a different word order or sentence structure than other languages. The order in Kambaatissa sentences is subject-object-verb structure, which means that the verb comes at the conclusion of the sentence.

2.6.1. Kambaatissa Language Word Sequence or Grammatical Structure

Kambaatissa sentence structure follows the order of SOV sequence for example, in this sentence “Amanne beet’uu xíjjee’u.” where “Amanne” is a subject, “beet’uu” is an object, and “xíjjee’u” is a verb. If the sentence is like this Beet’uu amanne xíjjee’u.” the sequence is OSV therefore it is word sequence error problem. Often, incorrect word order problems can be caused by adjective-noun disorder, adverb-verb disorder, or subject-verb disorder or object-verb disorder.

2.6.2. Kambaatissa Grammar Errors

Subject and Verb Agreement

In the following sentence, “Abari kabar walee’’ the subject “Abari” is third-person singular masculine and the verb “walee” is the third person singular masculine. The above sentence shows that the subject of a sentence and the verb agrees. Look this sentence “Abari kabar walete’ee.” the subject “Abari” is the third-person singular masculine in gender and the verb “walete’ee” is the third person singular with feminine in gender. So, the subject of the sentence and the verb is not much in the second example, so sentence has subject-verb disagreement error.

Adjective and Noun Agreement

The Adjective and noun should agree be with a number and gender. For example “Ababi buxichchuah’aa” the adjective “buxichchuah’aa” is the third person singular with masculine in gender and the Noun “Ababi” is the third-person singular masculine in gender, so in this example, the adjective agrees with the noun. On the contrary, let's look at this example “Ababi buxichchuta’aa” the adjective “buxichchuata’aa” is the third person singular with feminine in gender and the Noun “Ababi” is the third-person singular masculine in gender. Therefore, the adjective is not much in the second example, so sentence has adjective-verb disagreement error.

Adverb and Verb Agreement

An agreement error in a Kambaatissa sentence is either a verb-verb disagreement or a grammatical error. For example: “An ga’aata hujachchoomm.” the adverb “ga’aata” and the verb “hujachchoomm” disagree with each other because the adverb refers to future action and the verb refers to present action. If we replace the adverb “ga’aata” with “tema” the verb and the adverb is agreed with each other.

2.7. Different Approaches used in Grammar Checker

NLP has various areas of application; then an NLP application area is grammar checking. Therefore, to ensure the grammatical accuracy of a text, different studies are conducted using different approaches for different languages. The rule-based, statistical-based, hybrid-based, and deep learning approaches to grammar checking are the most popular and widely used.

2.7.1. Rule-Based Approach

The input text is examined using manually generated rules because rule-based grammar checking is a primitive technique that uses manually developing features. This method requires linguists to provide rules. The rule-based grammar checker technique has the advantage that it does not require grammar checkers to be trained and it is easy to add, modify or remove rules, so it provides accurate error messages. It has the drawback of requiring linguists to manually construct each rule, which takes time to develop. Many studies on rule-based grammar checking techniques are carried out for several languages, including English by different researchers, Amharic by Aynadis & Yaregal in 2013 for the first time, and Afan Oromo by Tesfaye in 2011, among others [20].

2.7.2. Statistical-Based Approach

Another approach to a statistics-based grammar checker is a machine learning or data-driven grammar checker. Statistical-based grammar checking technique, in contrast to rule-based grammar checking approach, uses a training corpus instead of manually developed rules to identify the correct content. The corpus can be gathered manually or automatically from a variety of online and offline sources, including periodicals, newspapers, magazines, and journals. This method checks the grammatical structure of a word sequence using N-gram models. If the command is well-known or frequent, the sentence is correct; if the order is strange or unusual, the statement is incorrect. A tagged corpus uses a statistics-based method to produce a sequence of tags. Numerous studies are carried out using statistical grammar validation techniques for several languages [21].

2.7.3. Hybrid Approach

Hybrid grammar checkers combine rule-based and statistics-based approaches to increase the accuracy of grammar checking systems. This method uses N-gram models to solve some errors and manual rules to solve others. This method has been used in different studies, including the Swedish and Tigrinya grammar checks [22], [23].

2.7.4. Deep Learning

Deep learning is a subfield of machine learning that overlaps or focuses primarily on the brain structure and function identified by artificial neural networks [24]. Unlike other machine learning methods, deep learning uses large or deep neural network algorithms. It

is also trained on big data. Deep learning is capable of automatically extracting features and learning from labeled data.

Deep learning is a machine learning concept based on artificial neural networks [25]. Many deep learning algorithms are used for various NLP applications. The most commonly used algorithms are [26], [27] RNN, CNN and DNN, LSTM used for many NLP applications. The algorithms mainly used in this study are RNN, LSTM and GRU.

Recurrent Neural Networks (RNN)

One of the main applications of recurrent neural networks is in deep learning algorithms that solve the sequence identification problem [28]. The main drawback of traditional neural network algorithms is that the inputs and outputs are independent, which means that they are limited in considering serial connections. However, in recurrent neural network architecture, the first input layer is the output of the second layer and the output of the second layer is the input of the third layer and so on. For short or long term memory, a recurrent neural network has its own memory.

The recurrent neural network achieves great success in natural language processing applications. The well-known recurrent neural network algorithms are LSTM, Bi-LSTM, BiGRU and GRU.

Long Short-Term Memory Network (LSTM)

Recurrent neural networks differ from Feed-Forward Neural Networks (FNN) in that they have recurrent connections and may recall recent input. RNNs are limited in that they are unable to learn long-term dependencies. This means that when data intervals are too long, the RNN is unable to learn the data.

The limitation of the recurrent neural network is another problem with extinction gradients. Long-short-term memory was proposed by [29], [30] Hochreiter and Schmidhuber in 1997 to solve long-term dependency problems in traditional recurrent neural network problems. Long-Short-Term Memory (LSTM) is one of the better neural networks than the traditional recurrent neural network in handling the problem of long-term dependencies. Short-Term Memory has the ability to remember long-term information. The chain-like structure of Long-Short-Term Memory is similar to a

recurrent neural network but with a different recurrent module structure. This recurrent module in LSTM consists of four layers compared to traditional neural network [31].

Gated Recurrent Unit (GRU)

A [32] gated recurrent unit was proposed in [30] to allow each repeater unit to handle different time scale dependencies and to adjust the data flow inside it without having a separate memory cell. The GRU cell tries to solve the same problem that the LSTM cell tries to solve using a method that is computationally and conceptually simpler than LSTM.

2.8. Related Works

Studies on grammar checkers for several languages have been conducted using different approaches.

2.8.1. Afan Oromo grammar checker

For Afan Oromo, a rule-based grammar checker was developed by [6]. Debela was inspired to apply the concepts of composition and grammar checker to Afan Oromo using a rule-based method. In order to identify grammatical errors in Afan Oromo, the author developed 123 grammar rules considering the morphological features of Afan Oromo language. Their model registered precision 88.9% and recall of 80%, respectively. In this study, 123 different rules are constructed and used to identify the grammatical errors of the language. Using these error detection rules, it can detect and correct grammatical errors in Afan Oromo texts. However, it cannot deal with anything other than the 123 rules developed to detect grammatical errors in the language because rule-based grammar checkers rely only on predefined rules to identify errors.

2.8.2. Grammar Checker for Wolaita

Tekalgn and Getachaw [33] developed a grammar error detector based on the Wolaita language rules. The researchers developed four modules to detect grammatical errors in simple sentences in the Wolaita language. The model achieved 93.47% accuracy, 55.17% precision and 51.61% recall. This study designed and developed a rule-based approach to simple sentences in Wolaita that can detect and correct grammatical errors in Wolaita texts by used only hand-crafted rules that can handle grammatical errors. However, a

rule-based approach is limited by rules designed to identify grammatical errors in the language.

2.8.3. Arabic Grammar Error Detection

Describe [11] is an on-going initiative to develop a web-based deep learning model for detecting Arabic grammatical errors. Grammar errors were identified using [34], [35] RNN method as well. They used the same databases utilized in QALB joint activities for training, development, and testing, in addition to using the Qatar Arabic Language Bank (QALB) corpus in Zaghouani for training and evaluation. They state that the corpus in their research poses a major challenge to their research because doesn't the corpus tell about error types.

2.8.4. English Grammar Detection

An LSTM-CRF-based machine learning model for analyzing and detecting English grammatical errors was proposed [10] the target of meeting standardized application requirements and improving English grammar accuracy. A single CRF algorithm model is easily limited by the design of a feature point template in the detection process.

This [36] task's primary goal is to identify and fix grammatical errors in sentences written in English. This study used machine learning to develop a [36] technique for identifying grammatical errors in English texts. The first thing this paper does is put a Seq2Seq-based [36] grammar error detection model into practiced. Secondly, this study practiced a grammatical error detection and correction system based on the Transformer model. Third, [37] has realized the application of the BERT model in grammatical error detection and correction tasks, which has significantly enhanced the model's generalizability. Fourthly, this article suggests a hybrid model-based approach to detecting and correcting grammatical errors in English composition. For grammatical error correction, the proper neural network model is applied based on particular application circumstances.

The aim of [9] is to meet the performance criteria of the standard English grammar error correction task in machine translation. To that end, [9] proposed a conventional neural network-based learning algorithm model with a network to achieve high performance.

In order to develop a machine learning algorithm model that can recognize English grammar errors, they [5] compared and analyzed various algorithm models that are currently in use in the field of education. The trained model will then be used in the English composition grammar detection system. Although the method used in this study is used to correct different grammatical errors, the grammatical error detection is only done on Spelling errors and Subject-Verb Agreements.

2.9. Summary

This study reviewed various research methods like rule-based, machine learning and deep learning-based approaches in foreign and Ethiopian local languages to check and develop grammatical errors. During the review, we found a gap in the Afan Oromo [21] that cannot deal with anything other than the 123 rules developed to detect grammatical errors in the language. And also by Tekalgn and Getachaw [22] for Wolaita, which can detect and correct grammatical errors in Wolaita texts used only hand-crafted rules that can handle grammatical errors. In addition, to foreign languages from Arabic [11] and English [5], [9], [10] we tried to understand the studies that used deep learning and could not achieve the performance of their model due to using a small dataset and did not tell us what type of grammatical error they wanted to fix. However, no one proposed to conduct Kambaatissa grammar error checker. Since there has been no previous research on grammar error checking in the Kambaatissa language, this study is not about fix the above-mentioned gaps, but studied used the existing gaps as input. Therefore, deep learning can solve the above problems by learning the sequence of words sequentially and extracting features automatically. So, this study selected design science research approach to conduct experiments LSTM and GRU deep algorithms to check grammatical errors of Kambaatissa language. Because the RNN extension algorithm gives the results promised in NLP. As a result, this algorithm can check for errors that appear in simple sentences.

CHAPTER THREE

3. RESEARCH METHODOLOGY

3.1. Introduction

This chapter details the research methodology for a proposed model using LSTM and GRU deep learning approaches to detect grammatical errors in simple Kambaatissa sentences. It presents a step-by-step procedure for the design and development of the proposed model.

This study identified the proposed model workflow and data sources to develop a corpus for training and validation of our model. In addition to this, the development environment and the tools needed to achieve the planned goal. In order to prevent the model from overfitting, this study also identified a feature extraction approach, deep hyperparameters, and a few regularization techniques. Finally, we concluded by outlining the deep learning model's performance evaluation parameters, which are used to measure how well the proposed model effectiveness.

3.2. Workflow of the Proposed Model

The workflow diagram below shows how the Kambaatissa grammar checker model operates.

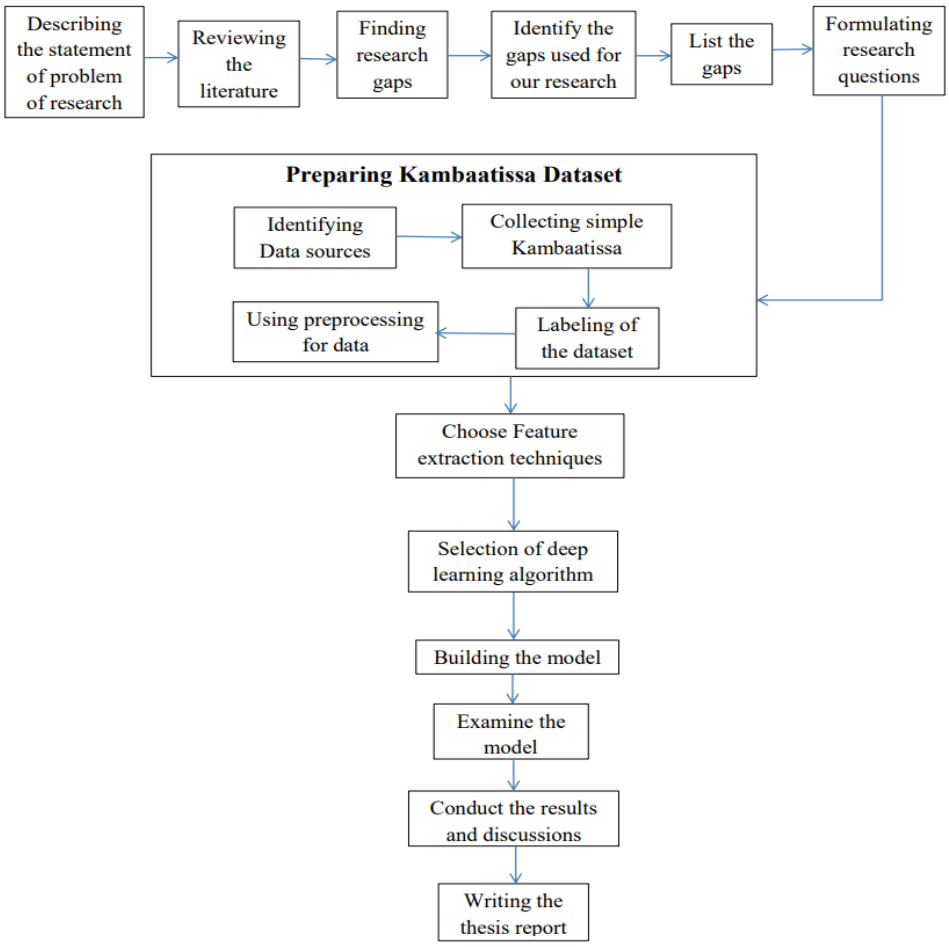

Figure 1. Workflow diagram of the research

3.3. Data Sources and Data Collections

3.3.1. Data Sources

To check grammatical errors in simple Kambaatissa sentences, this study needs a corpus to train and validate the proposed model. However, there is no published Kambaatissa dataset at all. Therefore, data collection becomes the first issue rather than corpus preparation. Accordingly, this study started from zero to collect the Kambaatissa dataset. To do that and achieve the goal of building a suitable dataset for Kambaatissa, this study planned, identified, and collected simple Kambaatissa sentences from the following authorized Kambaatissa linguistics-related data sources: Academic documents: elementary and secondary school textbooks, Hossana College of Teachers Education and Wachamo university Durame campus Kambaatissa language department modules, published Kambaatissa books.

3.3.2. Data Collection

After identifying the data source, this study collected the 5271 simple Kambaatissa sentences, from above listed data sources manually and electrically available data. However, in the collected data, the number of sentences that contain grammar errors such as subject-verb disagreement, adjective-verb disagreement, adverb-verb disagreement, appears rare, but one of them occurs frequently in Kambaatissa sentences. To solve the shortage and increase the dataset variety during proposed model training, this study used Kambaatissa linguistic instructions, to prepare grammatical errors simple Kambaatissa sentences.

3.4. Data Preprocessing

After identifying the necessary materials and data collections, this study performed data preprocessing. Data preprocessing is a technique that is used to remove irrelevant symbols from collected Kambaatissa sentences to train our model with quality data, and includes dataset labeling, data augmentation, data cleaning, data normalization, and tokenization.

3.4.1. Dataset Labeling

The process of categorizing the gathered data is called dataset labeling. The main purpose of dataset labeling for the proposed model is to identify each sentences category in the corpus. Three Kambaatissa linguistics experts performed the four different classes such as CORRECT, AND, SVD and AVD dataset labeling. The language experts were master degree holders from the Kambaatissa department.

3.4.2. Data Augmentation

The goal of data augmentation is a regularization method that can avoid overfitting by training model's cheaply with large numbers of training samples. Text data augmentation techniques have two advantages under this study, i.e., to balance classes' distribution and to increase dataset variety. Thus, available simple sentence data is represented in different ways (to increase dataset variation) without losing the sentences' syntax and semantic meanings.

The modified data set is then fed into the model as new training data to provide effective results on the test data set. A study on data augmentation reported in our paper that

adding various enhancements the training dataset improved model performance by an absolute 1.5% to 2.5% across all datasets [38]. After augmentation our train data set are 12574.

```
function perform_data_augmentation(dataset):
return [augmented for sentence in dataset for augmented in apply_augmentation(sentence)]
function apply_aug
mentation(sentence):
return [
replace_grammar_errors_with_synonyms(sentence),
restructure_sentence_logic(sentence),
inject_grammar_errors(sentence),
split_merge_sentences(sentence)
]
function replace_grammar_errors_with_synonyms(sentence):
return augmented_sentence
function restructure_sentence_logic(sentence):
return augmented_sentence
function inject_grammar_errors(sentence):
return augmented_sentence
function split_merge_sentences(sentence):
return augmented_sentence
```

Algorithm 1. Data Augmentation

3.4.3. Data Cleaning

There are an irrelevant character like Amharic symbols or non-Kambaatissa symbols (@, #, $, %^, &, *, (,), {,}, [,], :,”,’,>,<, etc. may exist in collected dataset. So, to remove such symbols, this study used a data cleaning technique. For example, if the input sentence is **“# dalti waal!”** this sentence includes the # symbol inside Kambaatissa sentences,but these symbols are not necessary. This study tackled such a problem by data cleaning algorithms. Algorithm for Data Cleaning:

Input: Input Kambaatissa_text datasets

Read the Kambaatissa _text datasets;

function clean_text(text):

cleaned_text = remove_special_characters(text)

cleaned_text = correct_whitespace(cleaned_text)

cleaned_text = lowercase_text(cleaned_text)

cleaned_text = remove_extra_spaces(cleaned_text)

return cleaned_text

Algorithm 2. Data Cleaning

3.4.4. Data Normalization

The process of converting texts into their standard format is called normalization. The collected Kambaatissa sentences have different writing formats (upper, lower, or mixed case) and abbreviations. Therefore, to convert such a writing format into the same standard form, the following algorithms are implemented.

```
function perform_grammar_check(text):
tokens = tokenize(text)
tagged_tokens = pos_tagging(tokens)
errors = []
for token in tagged_tokens:
if is_grammar_error(token):
correction = generate_correction(token)
errors.append((token, correction))
return errors
```

Algorithm 3. Data Normalization

3.4.5. Tokenization

Tokenization is the act of splitting a given text into sentences and, if needed, tokens. It is an important step in the NLP preprocessing. It involves dividing a sequence of text, such as a sentence or document, into smaller units called tokens. These symbols are anything in a sentence, including single words, sub-words, or characters, number, punctuation mark, or abbreviation, depending on the particular encoding method used.

We used TensorFlow tokenizer, in order to split strings. The TensorFlow tokenizer is a useful tool for sentence-level grammar checkers because it allows an input sentence to be broken down into smaller parts that can be easily parsed and processed. The main function of the parsing unit is to clean and split the input sentence. Cleaning is the process of removing non-Kambaatissa language sentences and other characters that are not important for this study and converting the input sentences into tokenized words.

Tokenization modules receive standard module input from the normalization module dataset. This study transformed text into sentences and sentences into words using Tensorflow Tokenizer packages and pre-trained libraries, allowing the resultant words could be individually tagged and examined in a character analyzer. Therefore, a tokenization procedure using a deep learning grammar checker approach was applied to both training and test sets. To perform its unique task, Kambaatissa input starts by reading a sentence, and then breaks the sentence into tokens. For instance, the system

splits the input sentence **Lateebi lamu harruchchu yoo's** into seven tokens, and it adds the <start> and <end> marker to the sentence. Finally, it returns the token list: '<s>', '**Lateebi', 'lamu', 'harruchchu', 'yoo's' '.' '</s>'**

```
function tokenize_sentences(input_text):

sentences = split_into_sentences(input_text) // depend on end
punctuation (e.g: '.', '!', '?', etc.)

tokenized_sentences = []

for single_sentence in sentences:

words = split_into_words(single_sentence) // tokenized sentence
using white space

tokenized_sentence = []

for word in words:

tokenized_sentence.append(word)
```

Algorithm 4. Tokenized sentence

3.5. Feature Representation Techniques

Text embedding plays an important role before building any deep learning model in NLP because a deep learning model cannot accept raw data as input and can only handle numerical tensors [8], [34]. Consequently, words must be changed into numerical expressions in order to apply deep learning models to NLP. To do that, word embedding (word2vect and fasttext) [37], [39]–[41] plays a crucial role in converting text into digital data.

3.6. Hyper-parameter Tuning for Deep Learning

In deep learning algorithms, a hyperparameter is a parameter whose value is often set before the start of the learning process and is not determined by training. So, to reduce the number of hyper-parameter trials or miss the usage of the hyperparameters in the proposed model, this study gathered the following deep learning tuning hyper-parameters and their recommended values to train the deep learning algorithms [42], [43]. The

hyperparameters we used in this study are dropout, batch size, learning rate, Adam optimizer, epoch, softmax for activation function and etc.

3.6.1. Hidden Layers

To reduce overfitting and underfitting, the hidden layers applied nonlinear changes to the inputs that are regularly entered into the network.

3.6.2. Regularization Techniques

A deep learning algorithm can suffer from overfitting, which is due to training small data sets and testing the performance of the model. Such problems can also occur in deep-learning based Kambaatissa grammar error checking models. To avoid overfitting, data addition and deletion are implemented in the proposed model. Dropout is the best generalization method used, that is, it gives the best result on the test set. This is done by randomly removing the nodes in each iteration during training and using the entire node when making predictions (testing) on the data set [44], [45].

3.6.3. Loss Function

In a deep learning algorithm, the difference between target labels and prediction labels is measured using a loss function. There are different loss functions that are used to deal with different types of work. Among them are binary cross-entropy loss, fold loss, and quadratic fold loss used in binary classification problems, and those used in multi-class classification are sparse class-entropy loss, cross-class-entropy loss, and Kullback-Leibler divergence loss. And also, mean square logarithm error loss, mean squared error loss, absolute error loss, and others are used again [42], [44], [45]. We used categorical cross-entropy as the loss function in this study.

3.6.4. Optimization Algorithms

In order to increase the efficiency of the model output, optimizers are techniques or methods that minimize the error function, or loss [42], [46]. Among different types of optimizers, we used Adam Optimizer for this study.

3.6.5. Learning Rate

Since the learning rate of the model changes with each step or step size of the search process, it also determines how large or small steps the gradient slope takes to reach the local minimum, or how fast or slow we move to the optimal weights. Because the

learning rate has an important effect, 0.0001 was used as the learning rate in this study. For example, if the rate is too high, the model will simply circle the minimum without ever reaching it, and should the rate be excessively low, the algorithm will become so slow that finding the minimum will take too long (or many iterations). Therefore, an average learning rate is preferable [47].

3.6.6. Activation Function

The activation function makes the network flexible; this also increases the ability to represent nonlinear robust arbitrary functional mappings between input and output and to extract complex information from data. In various activation functions such as sigmoid, softmax, ReLu, and others, we used softmax for the activation function because it has the highest probability in multi-class models to get the probabilities of each class [44], [48].

3.6.7. Batch size

The batch size affects how the learning process unfolds because it determines the range of examples (samples) from the training dataset that are utilized to estimate the error gradient. The main aim of batch size in a deep learning algorithm is to control the amount of training data sampled because we do not have enough resources to hold and store all the training data at once. The batch size was designed to train based on existing resources and reduce overfitting. The batch size is typically chosen between 1 and a few hundreds, and the default [42] batch size is 32 but we were used 128 batch size.

3.6.8. Epoch

Epochs represent the number of times an algorithm must see all the data or iterate over the complete training set, so a training epoch means that the learning algorithm has gone through a single training dataset (using each example once), during which instances are randomly divided into different sizes [42]. In this study, we used 100 epochs.

3.7. Model Performance Evaluation Metrics

When deep learning and machine learning algorithms are applied, it needs some measurement to find out, how well the developed model (artifact) performed the given task, and such measurement is called performance evaluation metrics. However, there are several significant metrics that have been introduced to measure algorithm performance

and validate the model's correctness. Some of the most common performance evaluation metrics are listed [49]–[51].

3.7.1. Confusion matrix

The confusion matrix is a widely used metric for assessing deep learning's algorithms effectiveness [52], which primarily summarizes the classification performance of a classifier using a sample of test data. Furthermore, it is often used for classification problems where the results may contain two or more types of learning.

3.7.2. Evaluation matrix

An evaluation matrix, also known as an evaluation metric, is a quantitative measure that we used to evaluate the model's performance or effectiveness. Evaluation metrics provide a standardized way to measure and compare different approaches or solutions based on specific criteria or objectives.

Evaluation metrics are typically designed to capture different aspects of performance, such as precision, accuracy, recall, or f-score, depending on the type of problem being evaluated. The selection of evaluation parameters depends on the specific goals, requirements, and characteristics of the system or model being evaluated.

Table 6. Evaluation matrix

Metric	**Formula**	**Description**
Accuracy	$\frac{TP+FN}{TP+FN+FP+TN}$	Overall performance of the model
Precision	$\frac{TP}{TP + FP}$	How the positive description is accurate
Recall	$\frac{TP}{TP + FN}$	Coverage of actual positive samples
F1-Score	$2 * \frac{Precison * Recall}{Precison+Recall}$	The harmonic means of precision and recall

3.8. Tools used to Develop

To perform data preprocessing, data analysis, visualizations, and develop the proposed model, this study used different software tools and libraries. The Python programming language is used for implementation, and it provides different packages (libraries) for text analysis. The integrated development environment (IDE), configuration files, code, and all Python libraries are assembled and packaged into a single software package at the preparation stage. These consist of setup a Jupyter notebook, configuring Anaconda, and text files with the Kambaatissa corpus in Python code for the grammar checker [53], [21], [54], [55]. Python libraries and APIs that must be imported are also defined so that code can run without errors. These APIs help the LSTM grammar debugger with high abstraction [56]. These consist of the Python libraries TensorFlow, Numpy, and the Keras API [57]–[59].

3.9. Environments to Develop the Proposed model

This study used personal computer and desktop computer to install the necessary software tools and their packages. And the specification of the personal pavilion computer is window10 pro-OS 64-bit, Intel(R) Core (TM) i5-3230M CPU @ 2.60GHz 2.60 GHz, 2 Core(s), 4 Gigabyte of physical memory, 4.42 Gigabyte of total virtual memory, and 500 Gigabyte hard disk storage capacities.

CHAPTER FOUR

4. DESIGN AND IMPLEMENTATION

4.1. Overview

One of the study's goals is designing a Kambaatissa language grammatical error checking model. A grammar checker takes a sentence from a given language document as input and checks the validity and correctness of the language. There are two phase in this study: preprocessing phase and learning phase. The prototypes include data preprocessing phase, which include data cleaning, tokenization, data labelling, morphological feature annotator, morphological analyzer, data augmentations, vectorization, data normalization and sequence padding. The second was the learning phase, which included dataset splitting, word embedding and deep learning techniques of LSTM and GRU. Therefore, in this chapter, the general architecture of the model and the detailed descriptions of the components of the proposed Kambaatissa grammar error checker model are described.

4.2. Architecture of the Proposed Model

Our model architecture and detail description of modules and components in the modules also as follows:

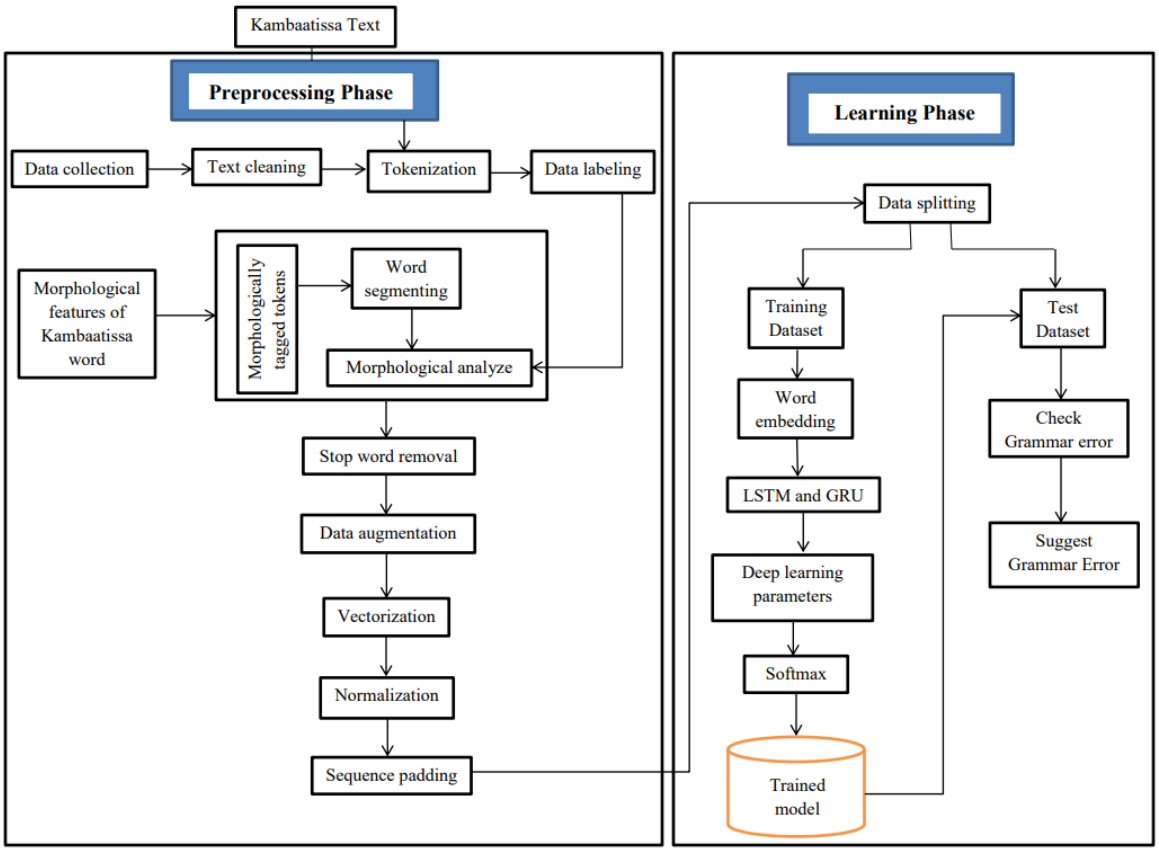

Figure 2. The model's architecture

4.3. Data Preprocessing

An important part of this phase is to perform preliminary work on the given text and is commonly called pre-processing [60]. The preprocessing module is the first part of the proposed model module. The Python programming language and its packages are used to design the proposed deep learning-based Kambaatissa grammar error checker model. Each preprocessing component has an I/O parameter. The outputs of one function or method are inputs for another. Therefore, Pandas Python packages are used to load the Kambaatissa dataset from Google Drive and local drives on Google collab and Jupiter notebook respectively. There will be a null (empty) row or column in the loaded dataset; to remove these, the python function a drop was used, and in addition drop_duplicate method was used to remove duplicated sentences from the corpus. During dataset labelling, Kambaatissa linguistic experts wrote texts categories as Correct, SVD, AND, and AVD to represent the class name of each simple sentence in the collected Kambaatissa dataset.

4.3.1. Data Cleaning

The text data collected by Kambaatissa language may contain numbers, punctuation, stop words and special characters. Such irrelevant signals were cleaned from the dataset to proposed model. The data cleaning process helped us normalize the text and remove unwanted characters or spaces. Following data loading, the data was processed by extracting newline characters from the sentence groups in the provided corpus and removing all punctuation. To make sure our model didn't reveal values that were too large or small and that the produced sentence was of a similar lengths, outliers were also eliminated from the data [61].

```
import re
from nltk.corpus import stopwords
def clean_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove punctuation
    text = re.sub(r'[^\w\s]', '', text)
    # Remove extra whitespaces
    text = re.sub(r'\s+', ' ', text).strip()
    # Remove stop words
    stop_words = set(stopwords.words('Kambaatissa language'))
    tokens = text.split()
    filtered_tokens = [token for token in tokens if token not in stop_words]
    text = ' '.join(filtered_tokens)
    return text
text = "  This is an example sentence with some extra spaces!  "
cleaned_text = clean_text(text)
print("Original Text:")
print(text)
print("\nCleaned Text:")
print(cleaned_text)
```

Figure 3. Implementation of data cleaning

4.3.2. Tokenization

The tokenization modules accept the input from the dataset normalization module. To tokenize the Kambaatissa texts into sentences and sentences into words, this study imported Keras Tokenizer packages and pre-trained libraries. The tokenized words were converted to a word index by a Keras function called fit_on_texts.

4.3.3. Morphological analyzer

A morphological analyzer is a linguistic tool or software component used in NLP and computational linguistics that is designed to analyze the morphological structure of words in a language. The process of extracting grammatical information from tokens in terms of their suffix information can be described as morphological analysis [62] [63], [64]. Morphological analysis involves dividing word units into morphemes and assigning grammatical morphemes to grammatical categories and word morphemes to lexemes [65], [66]. Kambatisa language is morphologically rich and has the basic characteristics of agglutinative languages where prefixes, suffixes and infixes are bound morphemes. For instance, the word "waalee" comprises two meaningful components or morphemes: the root "waal" and the bound morpheme "ee," which designates a feminine or masculine gender.

4.3.4. Data Augmentation

A text augmentation module refers to a component or software tool designed to enhance and distinguish textual information by generating variations or augmentations of the original text. Text augmentation module accepts labeled sentences and augments sentences by replacing one word from the sentences. Generally speaking, the code parses the input sentence using a CFG-based parser and uses a parser based on the grammar rules set by the display grammar, but does not explicitly check or correct the grammar. Therefore, to implement text augmentation techniques, a Python package called the nlpaug library is used to add Kambaatissa texts.

```
!pip install nlpaug
import nlpaug.augmenter.word as naw
aug_avd = naw.RandomWordAug(action='substitute', tokenizer=None,
                            stopwords_regex=None, verbose=3, reverse_tokenizer=None,
                            name='RandomWord_Aug', aug_min=1, aug_max=1,
                            aug_p=0.1 target_words=avd_tokens)
#augmenting AVD by original SVD
all_avd_data=[]
for text_avd in AVD_train_sentence['text']:
  #print(text)
  #all_text.append(text)
  augmented_textbyw=aug_avd.augment(text_avd, n=2, num_thread =2)
  all_avd_data.append(augmented_textbyw)
augmented_sen=[]
for k in all_avd_data:
  for t in K:
    #print(t)
    augmentedAVD_sen.append(t)
```

Figure 4. Implementation of Text data augmentation

4.3.5. Vectorization

The process of converting text or categorical data into a numerical form that can be fed into deep learning models is known as vectorization. It is a crucial step in the preprocessing of many NLP tasks, including grammar checking. We selected appropriate vectorization techniques such as word embedding (Word2Vec and FastText) to implement the chosen vectorization method to convert the text into digital features.

```
from sklearn.feature_extraction.text import CountVectorizer
# Example sentences
sentences = [
    "An foolooccu hoogomm.",
    "Issoo waabita aassaansa."
]
# Initialize the CountVectorizer
vectorizer = CountVectorizer()
# Fit the vectorizer on the sentences and transform them into vectors
vectorized_sentences = vectorizer.fit_transform(sentences)
# Get the feature names (words)
feature_names = vectorizer.get_feature_names()
# Print the vectorized sentences
for index, sentence in enumerate(sentences):
    print(f"Sentence {index+1}: {sentence}")
    print("Vectorized representation:")
    for word_index, word in enumerate(feature_names):
        count = vectorized_sentences[index, word_index]
        if count > 0:
            print(f"    {word}: {count}")
    print()
```

Figure 5. Implementation of Vectorization

4.3.6. Sequences Conservation

The Kambaatissa language searches for numerical sequences to match sentences with a deep learning algorithm. Consequently, the proposed model is trained and tested in this study, in the next subsection. The text-to-sequences component receives the numerical values of each word index from the word-index creation module and represents each word sequence by its index values.

To implement this, modules this study used the Tensorflow Tokenizer function called texts to sequences. The tokenizer is fitted on the input texts using the fit_on_texts method. This step prepares the tokenizer by building the vocabulary based on the input texts.

```python
import tensorflow as tf
def text_to_sequences(texts):
    # Create an instance of Tokenizer
    tokenizer = tf.keras.preprocessing.text.Tokenizer()
    # Fit the tokenizer on the input texts
    tokenizer.fit_on_texts(texts)
    # Convert the texts to sequences
    sequences = tokenizer.texts_to_sequences(texts)
    return sequences, tokenizer
input_texts = ["Sodannu iradona hi'rro.", "This is a sample sentence"]
sequences, tokenizer = text_to_sequences(input_texts)
# Print the sequences
for i, sequence in enumerate(sequences):
    print(f"Input Text: {input_texts[i]}")
    print(f"Sequence: {sequence}")
    print()
# Convert sequences back to texts
reconstructed_texts = tokenizer.sequences_to_texts(sequences)
# Print the reconstructed texts
for i, text in enumerate(reconstructed_texts):
    print(f"Sequence: {sequences[i]}")
    print(f"Reconstructed Text: {text}")
    print()
```

Figure 6. Implementation of Text to Sequences Conservation

4.3.7. Data Normalization

In the context of grammar checking, data normalization, also known as feature scaling, is a data preprocessing technique used to transform data into a common scale or range. We therefore consider the process of transforming or levelling the input text to ensure consistent and consistent representation for the Kambaatissa grammar analysis. Also, grammar checking involves preparing the text data in a proper format so that the algorithms or models can work efficiently.

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
# Input data
data = np.array([[5, 10], [2, 8], [3, 12], [1, 6]])
# Create a MinMaxScaler object
scaler = MinMaxScaler()
# Fit and transform the data
normalized_data = scaler.fit_transform(data)
# Print the normalized data
print("Normalized data:")
print(normalized_data)
```

Figure 7. Implementation of data normalization

4.3.8. Sequences Padding

The Sequence Padding module receives input from the Text to Sequence Protection module and aims to normalize each simple sentence in Kambaatissa. Since it helps us obtain a fixed-length representation, we matched the given length by adding wrapping tokens to short sequences or truncating long sequences.

```
import tensorflow as tf
import pad_sequences
from tensorflow.keras.preprocessing.sequence
def pad_sequences_with_length(sequences, max_length):
  padded_sequences = pad_sequences(sequences, maxlength=max_length,
                                   padding='post', truncating='post')
  return padded_sequences
padded_sequences = pad_sequences_with_length(sequences, max_length)
#print the padded sequences
for sequence in padded_sequences:
  print(sequence)
```

Figure 8. Implementation of sequence padding

4.4. Implementation of Word embedding Techniques

Word embedding methods are techniques used to represent words or phrases as dense vector representations. These techniques aim to capture semantic and contextual information in a continuous vector space, allowing machines to understand and process natural language more efficiently. A few commonly used word embedding techniques in

order to pass Kambaatissa sentences to the deep learning algorithm are: Word2Vec and FastText.

4.4.1. Implementation of word2vec

We used the popular Python library, gensim, to implement word2vect embedding techniques.

```
from gensim.models import Word2Vec
from genism.utils import simple_preprocess
# Prepocess the corpus
tokenized_corpus = [simple_preprocess(sentence) for sentence in corpus]
# Train the Word2Vec model
model = Word2Vec(tokenized_corpus, vector_size=300, window=10, min_count=1, workers=4, sg=0)
data = all_dataset.text.apply(genism.utils.simple_prepocess)
modelofw2v.build_vocab(data,prograss_per=10)
#.epochs
modelofw2v.train(data,total_examples=modelofw2v.corpus_count,epochs=100)
```

Figure 9. Implementation of word2vect

4.4.2. Implementation of FastText

FastText word embedding is an extension of word2vect; its difference is that it embeds each character's n-gram, with words being represented as the sum of these representations.

```
#FastText
import FastText
from genism.models
fastm=gensim.models.FastText(window=10, min_count=1, size=300, workers=4, sg=0)
fdata=all_dataset.text.apply(gensim.utils.simple_preprocess)
fastm.build_vocab(fdata, progress_per=100)
fastm.train(fdata, total_examples=fastm.corpus_count, epochs=fastm.epochs+95)
```

Figure 10. Implementation of FastText

4.5. LSTM Model and its Performance Metrics

LSTM neural network model can learn and predict continuous data, so Recurrent Neural Networks have been used especially on the built-in architecture. We have found that LSTMs are more adept at handling continuous data than recurrent neural networks.

LSTM doesn’t suffer from the fading problem compared to RNN. This RNN LSTM model was implemented using the Keras and TensorFlow neural network packages, which enable LSTM implementation in Python. The proposed LSTM model is built with three layers: the output layer, the LSTM layer, and the embedding layer [61].

4.5.1. Embedding Layer

The embedding layer receives the data from the input layer as input in the form of integers. It is responsible for converting text data from integer form to a short numeric vector representation. The embedding layer is trained to weight the data used as input in terms (as an integer) to reduce the order. The embedding layer helped us to reduce the word size from corpus words to very short length. An embedding matrix stores conversion data. Every term in the matrix is substituted with an index that can be used to determine its vector. Word embedding is done by this algorithm as this reduces the size of the dataset obtained by using Word2Vec algorithm [61].

4.5.2. LSTM layer

By accepting the LSTM layer's input as the embedding layer's output, the LSTM layer learned the time-phase sequence data's long-term dependencies. Because of this, the LSTM layer (reduced by the embedding layer) receives the sequence data, which relates to the Kambaatissa sentences in their vector representations. The LSTM layer can perform additional interactions that help the training set grow longer structures because the layer consists of a cell, an input gate, an output gate, and a forget gate [61].

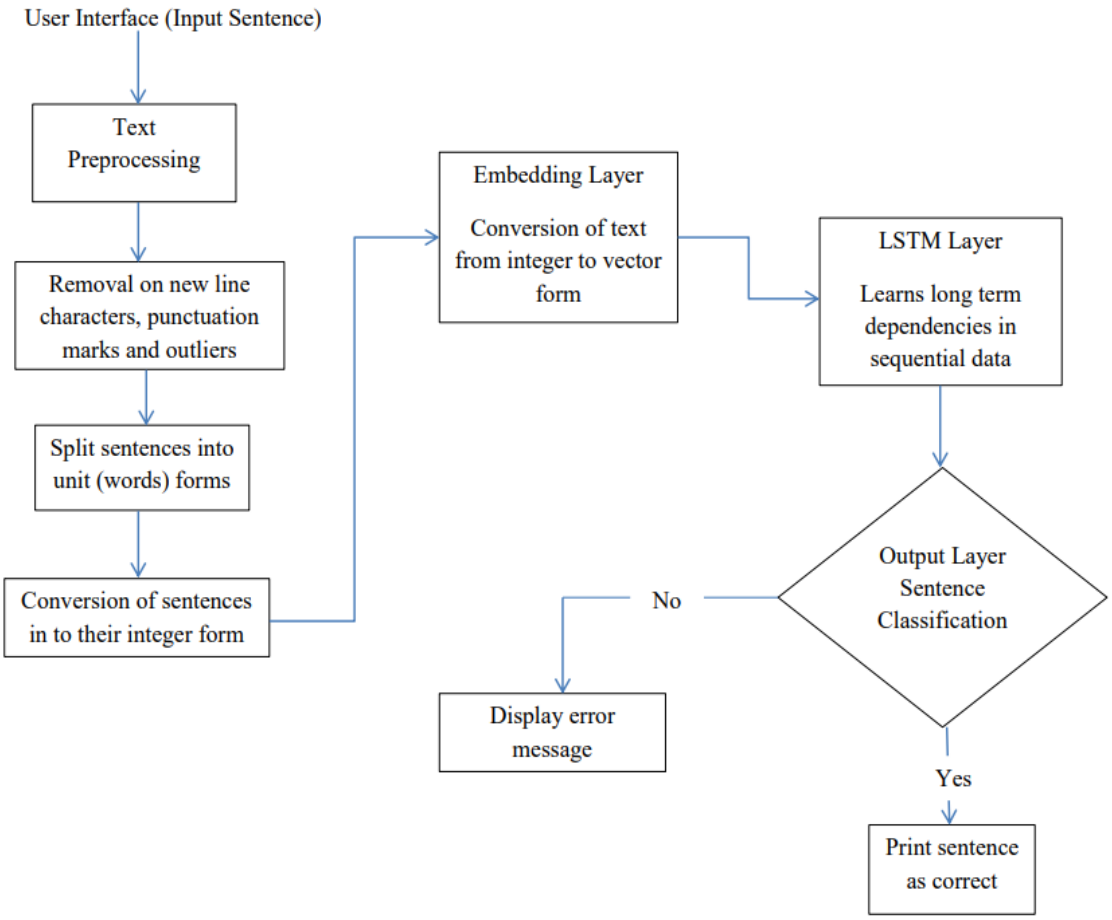

Figure 11. LSTM Implementation Algorithms and Procedure

4.6. Distribution of Dataset

The collected Kambaatissa dataset contains AND, SVD, AVD and corrected sentence segments. In this study, we collected a total of 5271 simple Kambaatissa sentence data for each subclass such as corrected sentences, AND, SVD and AVD. Also 12574 are augmented data.

The experiment was performed on balanced dataset, i.e., 25% of correct sentences, 25% of subject-verb disagreement, 25% of adjective-noun disagreement and 25% of adverb-verb disagreement.

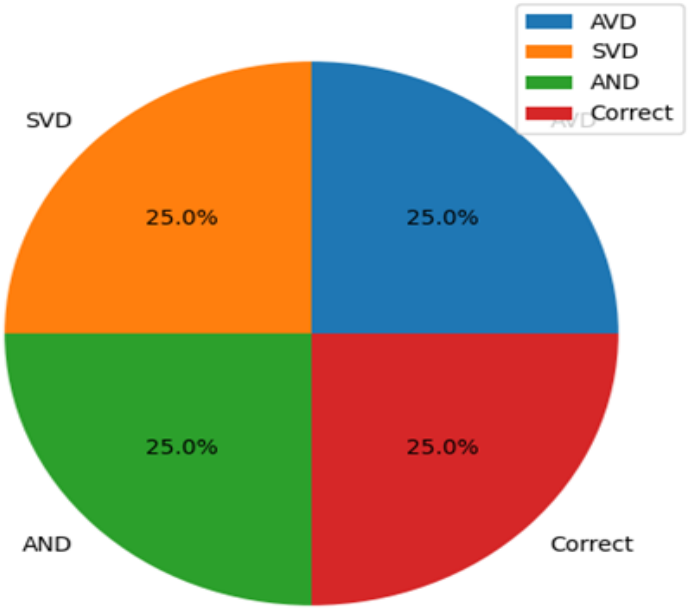

Figure 12. Dataset distribution

4.7. Implementation of Deep Learning Algorithms Module

The implementation of deep learning algorithms was performed after word embedding and sequence normalization. Sentence-Level grammar checker for Kambaatissa language using deep learning model is designed with three layers: embedding layers, LSTM layer and GRU layer with a dropout of 0.5, and four output dense layers with softmax activation (to predict the correct/incorrect label). Deep learning algorithms accept input from word embedding functions in the first layer that means, the output of word embedding components before receiving other deep learning hyperparameters.

```
# Define the deep learning model
model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_length))
model.add(LSTM(units=lstm_units, dropout=0.5, return_sequences=True))
model.add(GRU(units=gru_units, dropout=0.5))
model.add(Dense(units=output_dim, activation='softmax'))
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Figure 13. Implementation of Deep learning algorithm

CHAPTER FIVE

5. RESULTS AND DISCUSSIONS

5.1. Introduction

In this section, we have discussed experimental details and evaluation results of grammatical error checking and different experiments were carried out with identified word embedding techniques, train-test ratio, and deep learning hyper-parameters.

5.2. Performance evaluation and Testing

Deep learning using evaluation and testing is essential to ensure sentence-level grammar performance for the Kambaatissa language. Therefore, we evaluated the performance of our model using hyperparameters such as: 128 batch sizes, 100 epochs, 0.5 dropout, 0.0001 learning rate and Adam optimizer, as well as confounds such as: accuracy, precision, recall, and f1_score. Because the size of the data set varied, we carefully selected the evaluation parameters; as a result, accuracy cannot be accurately predicted, so we evaluated the model's performance using a confusion matrix.

- True Positives – These are those that were accurately predicted to be positive, indicating that both the predicted and actual class values were positive.
- True Negatives: These are values that were accurately anticipated to be negative, i.e., the difference between the actual and predicted values.
- False Positives – These occur when the classifier predicts the positive class an actual class is negative.
- False Negatives – The classifier predicts as negative but the actual class is positive.

After the resampling dataset, we used an 80/20 and 90/10 ratio for split the dataset into training sets and test sets. The performance of our model with the above ratio is verified on our training data set. Overall, we tested the system with various hyperparameters like epoch, batch size, learning rate, optimizer, and dropout in 80/20 and 90/10 ratio. We tested those parameters in LSTM and GRU. The class names or labels are represented by numbers from 0 to 3. Correct sentences by 0, subject-verb disagreement by 1, adjective-noun disagreement by 2 and adverb-verb disagreement by 3.

5.3. Test result using LSTM and GRU with 90/10 splitting ratio

In LSTM, we used parameters such as epoch=100, batch size=128, Adam optimizer, softmax for activation function with learning rate 0.0001 and dropout=0.5. 90% of the data set is assigned for training sets and 10% for validation sets, as shown in figure 14 below, which illustrates the performance of the LSTM network. Therefore, the model performs training accuracy 99.07% and validation accuracy 83.33%.

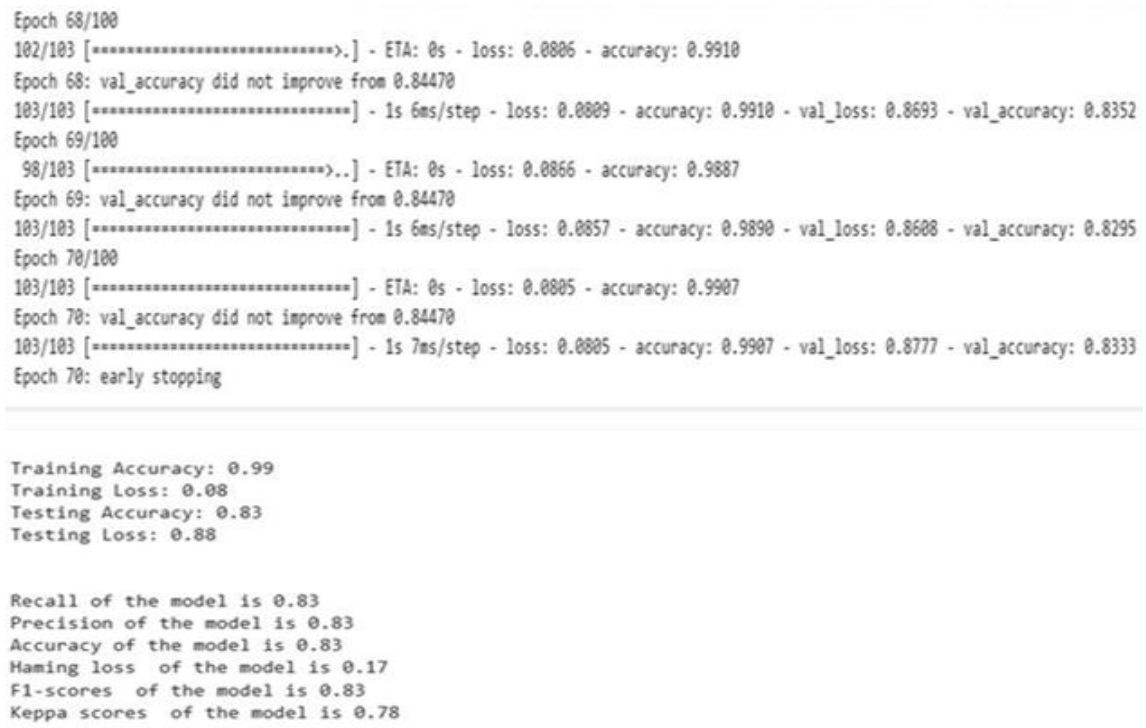

Figure 14. Shows the LSTM performance at 90/10 splitting ratio and epoch 100

The overall performance of each account on the LSTM model is shown blow in the confusion matrix of Figure 15. The confusion matrix show how many of the data is predicted actual class and incorrectly classified for example in 104 sentences are correctly predicted out of 132 sentences, the model predicts 110 sentences for adjective-noun disagreement class, the model predicted 104 sentences properly in the actual class out of 132 sentences for subject-verb disagreement class. In adverb-verb disagreement class, the model forecasted 112 sentences as fittingly out 132 sentences.

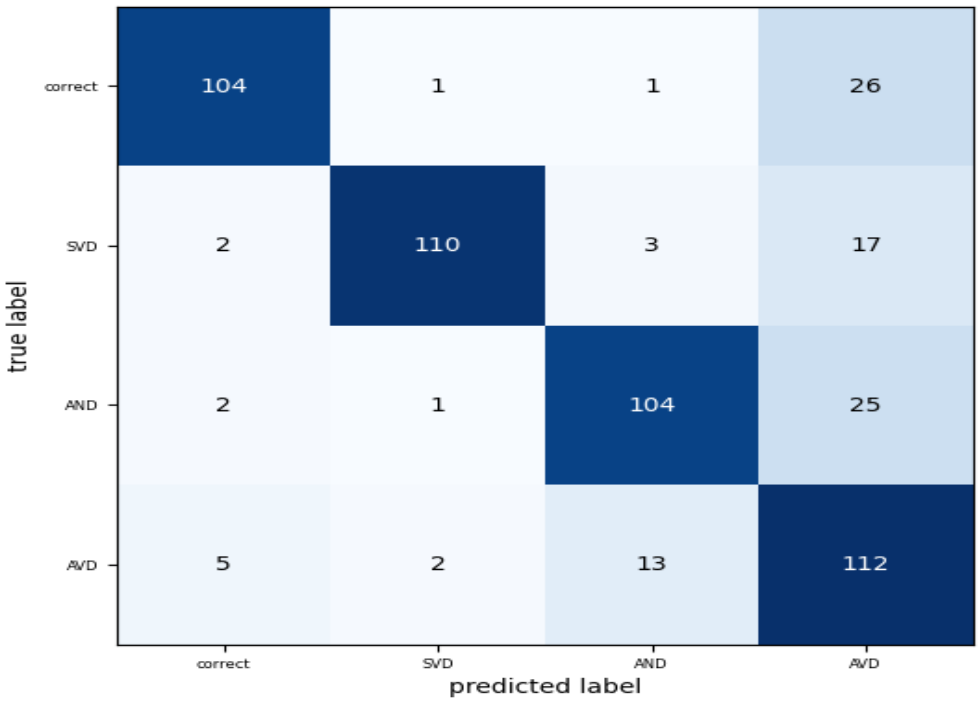

Figure 15. Confusion matrix for LSTM with 90/10

We can see the LSTM training results in Figure 16, where 90% of the dataset is assigned to the validation set and 10% to the test set. We trained the model at epoch 100 with a batch size of 128, but it did early stopping at 70.

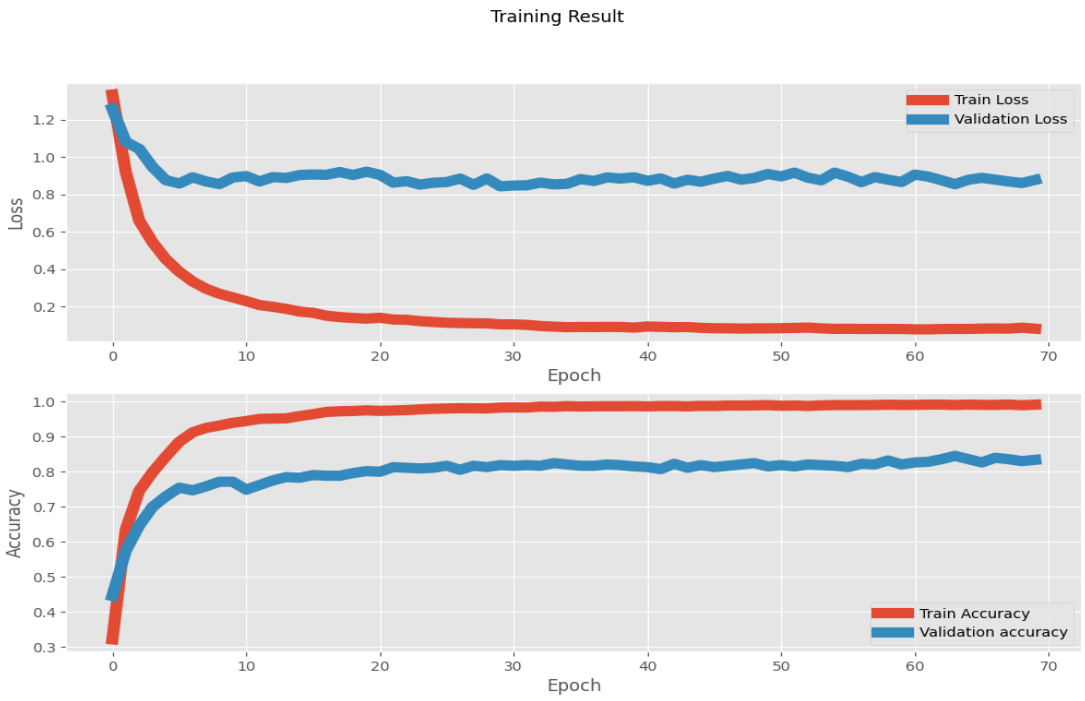

Figure 16. Training and Validation with LSTM 90/10

In GRU, we used parameters such as epoch=100, batch size=128, Adam optimizer, softmax for activation function with learning rate 0.0001 and dropout=0.5. 90% of the data set is assigned for training sets and 10% for validation sets, as shown in figure 17 below, which illustrates the performance of the GRU network. Therefore, the model performs training accuracy 99.56% and validation accuracy 82.58%.

```
101/103 [============================>.] - ETA: 0s - loss: 0.0692 - accuracy: 0.9940
Epoch 51: val_accuracy did not improve from 0.82765
103/103 [==============================] - 1s 7ms/step - loss: 0.0691 - accuracy: 0.9941 - val_loss: 0.9336 - val_accuracy: 0.8220
Epoch 52/100
103/103 [==============================] - ETA: 0s - loss: 0.0677 - accuracy: 0.9944
Epoch 52: val_accuracy did not improve from 0.82765
103/103 [==============================] - 1s 6ms/step - loss: 0.0677 - accuracy: 0.9944 - val_loss: 0.9457 - val_accuracy: 0.8220
Epoch 53/100
103/103 [==============================] - ETA: 0s - loss: 0.0665 - accuracy: 0.9950
Epoch 53: val_accuracy did not improve from 0.82765
103/103 [==============================] - 1s 7ms/step - loss: 0.0665 - accuracy: 0.9950 - val_loss: 0.9329 - val_accuracy: 0.8258
Epoch 54/100
101/103 [============================>.] - ETA: 0s - loss: 0.0661 - accuracy: 0.9952
Epoch 54: val_accuracy did not improve from 0.82765
103/103 [==============================] - 1s 6ms/step - loss: 0.0660 - accuracy: 0.9953 - val_loss: 0.9472 - val_accuracy: 0.8239
Epoch 55/100
 98/103 [===========================>..] - ETA: 0s - loss: 0.0641 - accuracy: 0.9956
Epoch 55: val_accuracy did not improve from 0.82765
103/103 [==============================] - 1s 6ms/step - loss: 0.0639 - accuracy: 0.9956 - val_loss: 0.9293 - val_accuracy: 0.8258
Epoch 55: early stopping
```

	precision	recall	f1-score	support
0	0.83	0.90	0.86	121
1	0.83	0.96	0.89	114
2	0.82	0.86	0.84	125
3	0.83	0.65	0.73	168
accuracy			0.83	528
macro avg	0.83	0.84	0.83	528
weighted avg	0.83	0.83	0.82	528

Figure 17. Shows the GRU performance at 90/10 splitting ratio and epoch 100

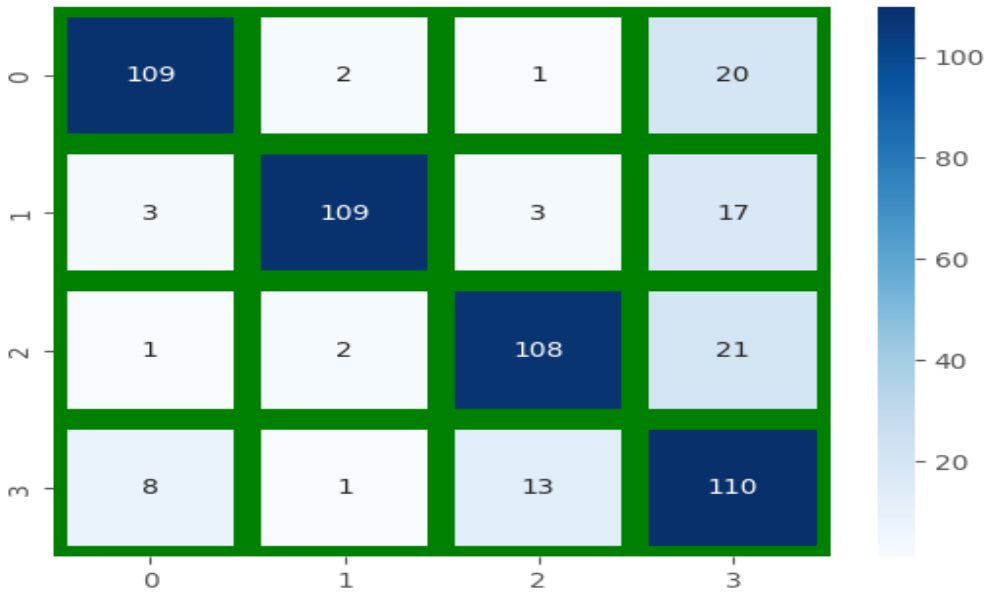

Figure 18. Confusion matrix for GRU with 90/10

The overall performance of each account on the GRU model is shown in the confusion matrix of the above Figure 18. The confusion matrix show how many of the data is predicted actual class and incorrectly classified for example in 109 sentences are correctly predicted out of 132 sentences, the model predicts 109 sentences for adjective-noun disagreement class, the model predicted 108 sentences properly in the actual class out of 132 sentences for subject-verb disagreement class. In adverb-verb disagreement class, the model forecasted 110sentences as fittingly out 132 sentences.

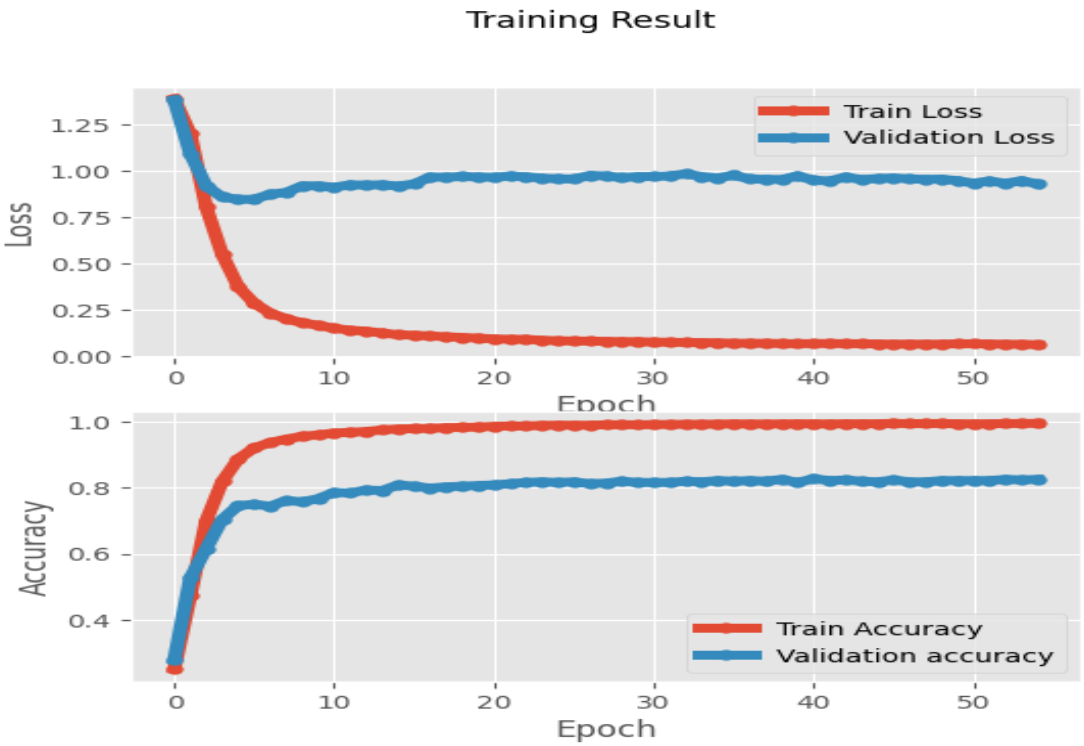

Figure 19. Training and validation with GRU 90/10 ratio

We can see the GRU training results in the above Figure 19, where 90% of the dataset is assigned to the validation set and 10% to the test set. We trained the model at epoch 100 with a batch size of 128, but it did early stopping at 55.

5.4. Test result using LSTM and GRU with 80/20 splitting ratio

```
Epoch 53: val_accuracy did not improve from 0.80966
99/99 [==============================] - 1s 7ms/step - loss: 0.0795 - accuracy: 0.9894 - val_loss: 0.9871
Epoch 54/100
95/99 [===========================>..] - ETA: 0s - loss: 0.0787 - accuracy: 0.9910
Epoch 54: val_accuracy did not improve from 0.80966
99/99 [==============================] - 1s 7ms/step - loss: 0.0785 - accuracy: 0.9910 - val_loss: 0.9801
Epoch 55/100
98/99 [============================>.] - ETA: 0s - loss: 0.0800 - accuracy: 0.9903
Epoch 55: val_accuracy did not improve from 0.80966
99/99 [==============================] - 1s 7ms/step - loss: 0.0800 - accuracy: 0.9903 - val_loss: 0.9800
Epoch 55: early stopping
```

	precision	recall	f1-score	support
0	0.79	0.89	0.84	234
1	0.80	0.97	0.88	218
2	0.81	0.84	0.82	255
3	0.83	0.62	0.71	349
accuracy			0.81	1056
macro avg	0.81	0.83	0.81	1056
weighted avg	0.81	0.81	0.80	1056

Figure 20. Shows the LSTM performance at 80/20 splitting ratio and epoch 100

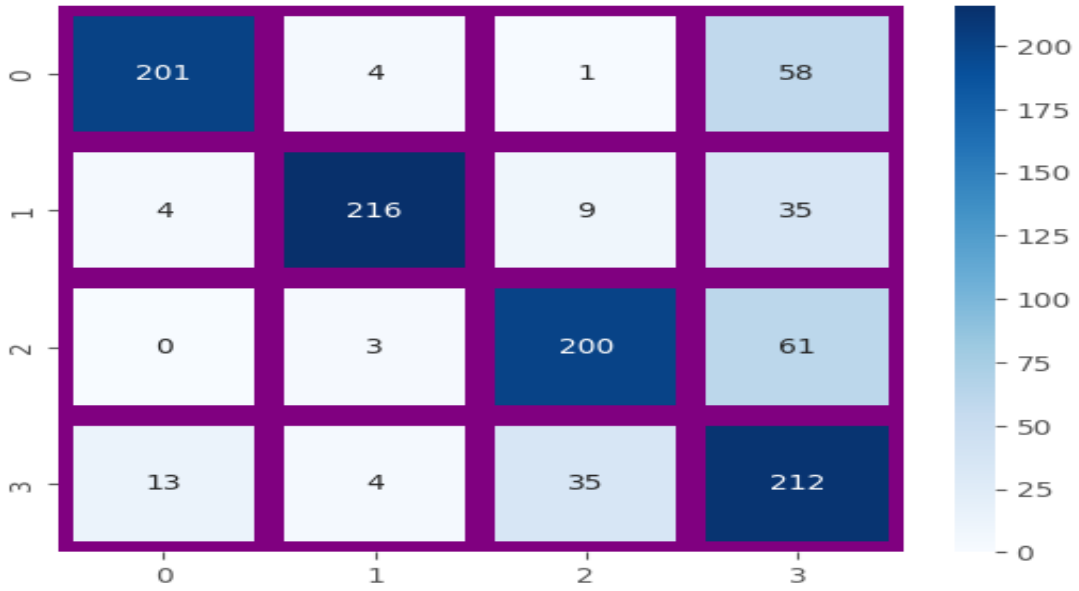

Figure 21. Confusion matrix for LSTM with 80/20

The overall performance of each account on the LSTM model is shown in the confusion matrix of Figure 21. The confusion matrix show how many of the data is predicted actual class and incorrectly classified for example in 201 sentences are correctly predicted out of 264 sentences, the model predicts 216 sentences for adjective-noun disagreement class, the model predicted 200 sentences properly in the actual class out of 264 sentences for subject-verb disagreement class. In adverb-verb disagreement class, the model forecasted 212 sentences as fittingly out 264 sentences.

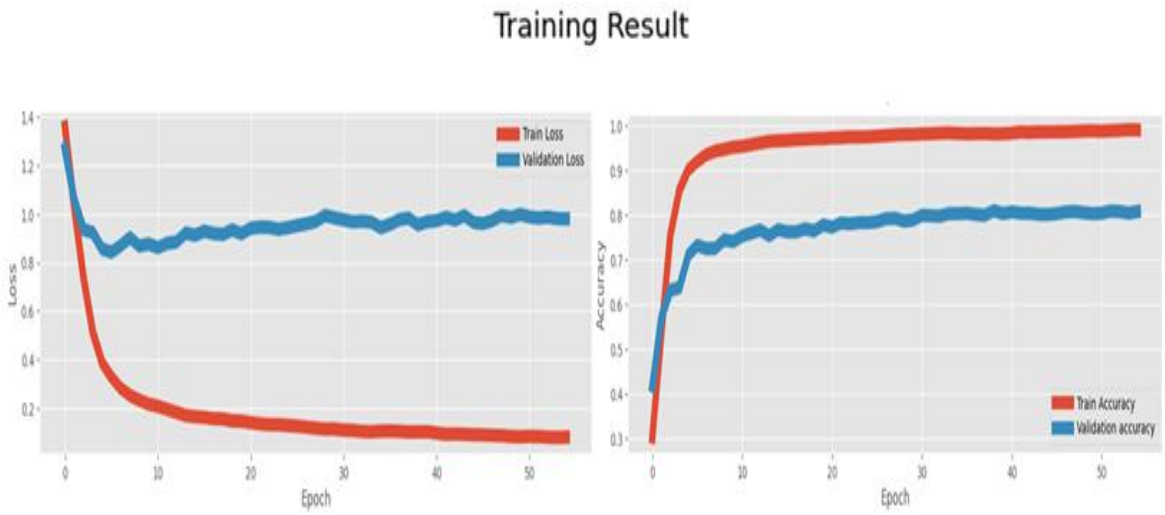

Figure 22. Training and validation with LSTM 80/20 ratio

80% of the data set is assigned for training sets and 20% for validation sets, as shown in figure 22 above, which illustrates the performance of the GRU network. We trained the model at epoch 100 with a batch size of 128, but it did early stopping at 55.

```
Epoch 41: val_accuracy improved from 0.80682 to 0.80777, saving model to training_10/cp.ckpt
99/99 [==============================] - 1s 11ms/step - loss: 0.0767 - accuracy: 0.9908 - val_loss: 0.9930 - val_accuracy: 0.8078
Epoch 42/100
97/99 [============================>.] - ETA: 0s - loss: 0.0756 - accuracy: 0.9907
Epoch 42: val_accuracy improved from 0.80777 to 0.81061, saving model to training_10/cp.ckpt
99/99 [==============================] - 1s 11ms/step - loss: 0.0757 - accuracy: 0.9906 - val_loss: 0.9862 - val_accuracy: 0.8106
Epoch 43/100
94/99 [===========================>..] - ETA: 0s - loss: 0.0755 - accuracy: 0.9905
Epoch 43: val_accuracy did not improve from 0.81061
99/99 [==============================] - 1s 9ms/step - loss: 0.0759 - accuracy: 0.9902 - val_loss: 0.9979 - val_accuracy: 0.8097
Epoch 44/100
98/99 [============================>.] - ETA: 0s - loss: 0.0748 - accuracy: 0.9905
Epoch 44: val_accuracy did not improve from 0.81061
99/99 [==============================] - 1s 8ms/step - loss: 0.0747 - accuracy: 0.9905 - val_loss: 1.0028 - val_accuracy: 0.8030
Epoch 45/100
95/99 [===========================>..] - ETA: 0s - loss: 0.0744 - accuracy: 0.9896
Epoch 45: val_accuracy did not improve from 0.81061
99/99 [==============================] - 1s 6ms/step - loss: 0.0746 - accuracy: 0.9895 - val_loss: 0.9907 - val_accuracy: 0.8106
Epoch 46/100
97/99 [============================>.] - ETA: 0s - loss: 0.0728 - accuracy: 0.9913
Epoch 46: val_accuracy did not improve from 0.81061
99/99 [==============================] - 1s 6ms/step - loss: 0.0730 - accuracy: 0.9911 - val_loss: 1.0127 - val_accuracy: 0.8087
Epoch 46: early stopping
```

	precision	recall	f1-score	support
0	0.81	0.91	0.86	235
1	0.84	0.99	0.91	225
2	0.78	0.82	0.80	249
3	0.81	0.61	0.70	347
accuracy			0.81	1056
macro avg	0.81	0.83	0.82	1056
eighted avg	0.81	0.81	0.80	1056

Figure 23. Shows the GRU performance at 80/20 splitting ratio and epoch 100

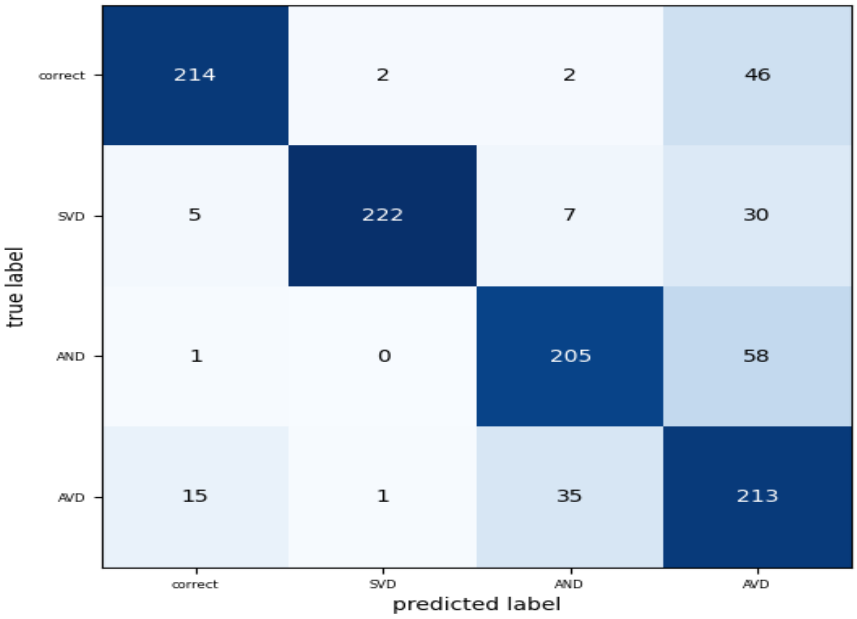

Figure 24. Confusion matrix for GRU with 80/20

The overall performance of each account on the GRU model is shown in the confusion matrix of Figure 24. The confusion matrix show how many of the data is predicted actual class and incorrectly classified for example in 214 sentences are correctly predicted out of 264 sentences, the model predicts 205 sentences for adjective-noun disagreement class, the model predicted 222 sentences properly in the actual class out of 264 sentences for subject-verb disagreement class. In adverb-verb disagreement class, the model forecasted 213 sentences as fittingly out 264 sentences.

Figure 25. Training and validation with GRU 80/20 ratio

80% of the data set is assigned for training sets and 20% for validation sets, as shown in figure 25 above, which illustrates the performance of the GRU network. We trained the model with 46 epoch number with a batch size of 128.

5.5. Discussion of Results

In this study, we have developed a sentence-level grammar checker for Kambaatissa language by using deep learning algorithm using LSTM recurrent neural network and GRU network. The proposed sentence-level grammar checker for Kambaatissa language model is evaluated for SVD, AND, and AVD. In general, we found errors in Kambaatissa simple sentences such as adjective-noun disagree with number and gender and adverb-verb disagree in tense, subject-verb (disagree in number, gender and person). Finally, to ensure that the entered sentence is correct, we have entered the correct sentence.

We split our data set for training and testing into 80/20 and 90/10 and evaluated the model; and we took 10% from training data. To train, validate and test our model's the hyperparameters like epoch = 100, batch size = 128 and dropout = 0.5, softmax for activation function and Adam optimizer with 0.0001 learning rate are used i.e. common metrics for grammar checking model to determine the effect of word embedding techniques, loss function, train-test ratio, and batch size to detect Kambaatissa grammar errors, and evaluated each experimentation results by using accuracy, precision, recall, F1-score, Kappa score and confusion matrix.

The experiment showed that the performance of LSTM and GRU is almost the same. However, LSTM is slightly better than GRU. LSTM achieved 83% training accuracy and 99.07% testing accuracy at 90/10 split ratio and 81% training accuracy and 99.03% testing accuracy at 80/20 split ratio. And also the performance of dividing the data set by 90/10 is better than dividing the data by 80/20%. The experiment shows that the accuracy and loss are changing from one iteration to another iteration while training the model. The reason behind this is that a different sample data is taken for each iteration.

Table 7. Evaluation result of loss functions

Train test	***Algorithm***	***Learning rate***	***Loss function***	***Optimizer***	***AF***	***Accuracy***	***Loss***	***F1_Score***
90/10	LSTM	0.0001	CCE	Adam	Softmax	81%	1.04%	81%
						83%	0.87%	83%
80/20						79%	1.08%	79%
						81%	0.98%	81%
90/10	GRU	0.0001	CCE	Adam	Softmax	81%	0.77%	81%
						83%	0.92%	83%
80/20						79%	1.10%	79%
						81%	1.01%	81%

This study applied categorical cross entropy, without changing other word embedding techniques, when examining categorical cross entropy to check grammatical errors in Kambaatissa. Thus, 80/20 deep learning with LSTM and GRU achieves the highest validation loss with GRU at 1.10% and LSTM at 1.08%. The lowest loss was recorded as 0.77% with GRU deep learning 90/10 categorical cross entropy minimum validation.

Table 8. Evaluation Result of LSTM and GRU with Word embedding techniques

Algorithm	Train test ratio	Feature extracting	Accuracy	Precision	F1_Score	Kappa score
LSTM	90/10	word2vec	81%	81%	81%	75%
		fasttext	**83%**	**83%**	**83%**	**78%**
	80/20	word2vec	79%	79%	79%	71%
		fasttext	81%	81%	81%	74%
GRU	90/10	word2vec	81%	81%	81%	74%
		fasttext	**83%**	**83%**	**83%**	**77%**
	80/20	word2vec	79%	79%	79%	71%
		fasttext	81%	81%	81%	71%

When this study applied, the batch size of 100 without changing other word embedding techniques, and deep learning hyper-parameter, by LSTM and GRU for 80/20, the batch size 10 achieved better performance with precision of 83%, recall of 83% testing

accuracy of 90% by LSTM fasttext word embedding. And when this study applied batch size of 100 by 90/10 with LSTM and GRU the batch size 100 outperformed and recorded the validation accuracy of 83%, Precision of 83%, recall of 83%. After conducting the experimentations, this study achieved the above best experimentation results to check Kambaatissa grammatical errors in both LSTM and GRU. This study applied LSTM and GRU static word embedding techniques with 90/10 and fasttext word embedding techniques with 83% testing accuracy with LSTM and GRU, but LSTM was outperformed through kappa score.

Generally, two research questions were examined in this study, and the following explanation of the findings follows:

RQ1. What kinds of errors do we make in grammar when writing in the Kambaatissa language?

In Kambaatissa language, three types of errors occurred while studying this research. There are SVD, AND, and AVD. Experiments were performed or achieved according to 25% dataset balanced for each class. Accordingly, the grammar error checker result shows 72% subject-verb agreement, 77% adjective-noun agreement and 78% adverb-verb agreement. But grammatical errors are frequently observed in subject-verb disagreement in the Kambaatissa language.

RQ2. Which deep learning algorithms are the best for checking grammar in Kambaatissa Language?

To answer this question, different experiments have been conducted with different word embedding techniques, loss functions, train-test ratios, and batch sizes. We tested two models. The models are LSTM and GRU as we tried to describe above. Surprisingly, both models scored the same record in the splitting ratio of 90/10 using fasttext in static word embedding. However, we found the LSTM algorithm to be better than GRU by a small margin.

As the experimentation result shows, fasttext word embedding techniques achieved better performances, i.e., 83% f1-score and 78% of kappa score, than word2vect. Batch size and the train-test ratio are also examined. As the experiment shows batch size 128 outperformed 83% of the f1-score and 0.17 hamming loss, and 90% for training and 10% for test dataset splitting techniques achieved 83% of f1-score than 80% for training and

20% for test dataset splitting approaches. After conducting the experimentations, as experimentation results shown the combination of 90/10 train-test dataset splitting techniques, fasttext word embedding techniques, LSTM, 0.5 dropouts, softmax activation function, CCE loss function, Adam optimizer with 0.0001 learning rate, 128 batch size parameter combination achieved the best result in this study.

CHAPTER SIX

6. CONCLUSION AND RECOMMENDATION

6.1. Conclusion

One of the applications in NLP is the grammar checker model. Compared to manual checking, the model is designed to reduce the time and cost of checking for grammar errors. A number of studies based on language structure and developmental approaches have been attempted in different languages to achieve the objectives of the grammar checker model. But since no research has been done on grammar checker in Kambaatissa language yet, the researcher aimed to develop a sentence-level grammar checker for Kambatisa language using deep learning approach. The grammar checker is the first deep learning approach for Kambaatissa language. The preprocessing module was responsible for text cleaning, tokenization, data labeling, Kambaatissa morphological features of word, stop word removal, data augmentation, vectorization, and finally normalization and sequence padding. Then, we used static word embedding (feature extraction) techniques were applied to embed Kambaatissa texts in deep learning algorithms.

The model received the word embedding output in the form of a matrix with 300 dimensions for feature extraction as fasttext. After that, both LSTM and GRU trained the common sentence sequences. To calculate the loss by LSTM or GRU, this study used CCE and ADAM optimizer was used to reduce the loss with a learning rate of 0.0001. The probability of each sentence after the minimum loss is reached is sent to the last dense layer and then the last dense layer receives the calculated sentence probability from the layer. Then, based on the probability of each sentence, the softmax activation function classifies each sentence as correct, AND, SVD, or AVD.

This research work used Python 3.10 version, Keras TensorFlow sklearn as the backend and Flask to develop the proposed model. The study has four categories such as correct sentences, SVD, AVD, and AVD. The performance and quality of each experiment have been evaluated by objective evaluation metrics to compare the proposed model's predicted sentence class with language experts' annotated sentence classes by precision, recall, f1-score, hamming loss, and confusion matrix. Ultimately, the performance of sentence-level grammar checker for Kambaatissa language by using deep learning approach was evaluated with confusion metric. This research work model (is called

LSTM) was performing 99 % train accuracy, 83% test accuracy, F1-score of 83%, recall of 83%, and precision of 83% and kappa score of 78%.

6.2. Contribution of the study

The following is a list of this study's contributions:

- A corpus containing 5271 morphologically annotated sentences and 9000 unique words was prepared for this study.
- This study prepared word embedding for Kambaatissa texts like word2vec and fasttext.
- The study used modern and advanced deep learning techniques to examine the grammar error of the Kambaatissa language.
- This study examined static word embedding techniques and categorical cross entropy we used for loss functions form NLP applications.
- Examined both LSTM and GRU deep learning neural network algorithm.

6.3. Recommendations

When developing a sentence-level grammar checker for the Kambaatissa language, it is important to consider the relevant stakeholders. Here are some tips.

- The model can be deployed by entering Kambaatissa sentences and grammar suggestions into the application through Microsoft Office.
- Enter or paste the Kambaatissa simple sentences text that you want to check for grammar errors into the provided text input area.
- The grammar checker will check and highlight any grammar errors it identifies in the input text. This includes errors related to subject-verb agreement, adjective-noun agreement and adverb-verb disagreement.

6.4. Future Works

Using a deep learning approach, we have tried to develop a sentence-level grammar checker model for Kambaatissa language in this research paper to cover as much as we can to solve grammatical errors in sentences. It is believed that the Kambaatissa grammar checker developed in this study needs further improvements. Therefore, we recommend further research improvements in this area to help users identify their grammatical errors, as there are types of grammatical errors that were not covered in this study:

- This study only focused on three grammatical errors of Kambaatissa. However, there are other grammatical errors in Kambaatissa language such as subject-object disagreement, misuse of punctuation marks and others. Therefore, this study recommends that such grammatical errors be included in future works.
- This study was only applied to simple sentences. As the best grammatical error checker, this study strongly recommends to include compound sentences and complex sentences for further studies.

REFERENCES

[1] Y. Treis, Y. Treis, C. Form, and Y. Treis, “The apprehensive in Kambaata (Cushitic): Form , meaning and origin To cite this version : HAL Id : hal-03095794 The apprehensive in Kambaata (Cushitic): Form , meaning and origin,” 2021.

[2] Y. Treis, *(Cushitic Language Studies 26) Yvonne Treis - A Grammar of Kambaata, Part 1_ Phonology, Nominal Morphology and Non-verbal Predication-Rudiger Koppe Verlag (2008)*.

[3] P. Goyal, S. Pandey, and K. Jain, *Deep Learning for Natural Language Processing*.

[4] M. HAGIWARA, *Real-World Natural Language Processing*.

[5] H. He and D. Maia, “Application of Grammar Error Detection Method for English Composition Based on Machine Learning,” vol. 2022, 2022.

[6] D. Tesfaye, “A rule-based Afan Oromo Grammar Checker,” vol. 2, no. 8, pp. 126–130, 2011.

[7] A. T. Gebru, “Design and Development of Amharic Grammar Checker,” no. March, 2013.

[8] Y. DESSALEGNE, “DEEP LEARNING BASED AMHARIC GRAMMAR ERROR DETECTION,” no. August, 2020.

[9] Y. Zhong and X. Yue, “On the correction of errors in English grammar by deep learning,” no. 178, pp. 260–270, 2022.

[10] L. Wu and M. Pan, “English Grammar Detection Based on LSTM-CRF Machine Learning Model,” vol. 2021, 2021.

[11] N. Madi and H. S. Al-Khalifa, “A Proposed Arabic Grammatical Error Detection Tool Based on Deep Learning,” in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 352–355. doi: 10.1016/j.procs.2018.10.482.

[12] T. T. Kambata, L. Kambata, N. Testament, O. Testament, and I. Phonetic, “K

ambata," pp. 1–20.

[13] Y. Treis, "Motion Events in Kambaata *," vol. 5, no. July, pp. 197–226, 2007.

[14] A. Notes and W. L. Source, "Notes on Kambatta Additional of Southern," no. 1956, pp. 985–993, 2015.

[15] G. Ciardo and P. Darondeau, "Lecture Notes in Computer Science: Preface," *Lect. Notes Comput. Sci.*, vol. 3536, 2005.

[16] Y. Treis, "Categorial hybrids in Kambaata 1," vol. 33, no. 2, pp. 215–254, 2012, doi: 10.1515/jall-2012-0009.

[17] Y. Treis, Y. Treis, K. Cushitic, Y. Treis, and K. Cushitic, "The approximative derivation in Kambaata (Cushitic) To cite this version : HAL Id : hal-03914947," 2023, doi: 10.21248/zwjw.2023.1.88.

[18] J. S. SUMAMO, "DESIGNING A STEMMING ALGORITHM FOR KAMBAATA TEXT: A RULE BASED APPROACH," 2018.

[19] Y. Treis, "Treis2005_Kambaata Kinship Terms.pdf."

[20] H. Yang, L. Luo, L. P. Chueng, D. Ling, and F. Chin, "Deep Learning and Its Applications to Natural Language Processing," pp. 89–109.

[21] V. Henrich and T. Reuter, "LISGrammarChecker : Language Independent," no. February, 2009.

[22] A. G. GEBREEGZAABHER, "Development of Tigrigna grammar checker using hybrid approach," no. June, 2018.

[23] V. Domeij, Rickard and Knutsson, Ola and Carlberger, Johan and Kann, "Granska--an efficient hybrid system for Swedish grammar checking," *Proc. 12th Nord. Conf. Comput. Linguist. (NODALIDA 1999)*, pp. 49–56, 2000, [Online]. Available: https://aclanthology.org/W99-1005.pdf%0Ahttps://aclanthology.org/W99-1005

[24] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Comput. Sci.*, vol. 2, no. 6, pp. 1–20, 2021, doi: 10.1007/s42979-021-00815-1.

[25] C. Janiesch, P. Zschech, and K. Heinrich, “Machine learning and deep learning,” *Electron. Mark.*, vol. 31, no. 3, pp. 685–695, 2021, doi: 10.1007/s12525-021-00475-2.

[26] S. Alshahrani and E. Kapetanios, “Are deep learning approaches suitable for natural language processing?,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9612, no. June, pp. 343–349, 2016, doi: 10.1007/978-3-319-41754-7_33.

[27] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, “Deep Learning Based Text Classification: A Comprehensive Review,” no. April, 2020, [Online]. Available: http://arxiv.org/abs/2004.03705

[28] J. Brownlee, “Long Short-Term Memory Networks With Python,” *Mach. Learn. Mastery With Python*, vol. 1, no. 1, p. 228, 2017.

[29] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[30] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling,” no. December, 2014, [Online]. Available: http://arxiv.org/abs/1412.3555

[31] Z. Cui, R. Ke, Z. Pu, and Y. Wang, “Stacked bidirectional and unidirectional LSTM recurrent neural network for forecasting network-wide traffic state with missing values,” *Transp. Res. Part C Emerg. Technol.*, vol. 118, pp. 1–11, 2020, doi: 10.1016/j.trc.2020.102674.

[32] N. E. Helwig, S. Hong, and E. T. Hsiao-wecksler, *Advanced Data Mining and Applications*.

[33] B. Tekalign, “RULE BASED WOLAITA GRAMMAR CHECKER A Thesis Submitted to Department of Information Technology , School of Graduate Studies , Wolaita Sodo University In Partial Fulfillment of the Requirements for the Degree of Master in Information Technology Tekalgn Balc,” no. January, 2020.

[34] A. A. Patel and A. U. Arasanipalai, *Applied Natural Language Processing in the*

Enterprise.

[35] A. Alothman and A. M. Alsalman, “An Arabic Grammar Auditor Based on Dependency Grammar,” *Adv. Human-Computer Interact.*, vol. 2020, 2020, doi: 10.1155/2020/8856843.

[36] J. Zhu, X. Shi, and S. Zhang, “Machine Learning-Based Grammar Error Detection Method in English Composition,” vol. 2021, 2021.

[37] H. A. Mohamed Hassan, G. Sansonetti, F. Gasparetti, A. Micarelli, and J. Beel, “BERT, ELMo, use and infersent sentence encoders: The Panacea for research-paper recommendation?,” *CEUR Workshop Proc.*, vol. 2431, no. September, pp. 6–10, 2019.

[38] J. Wei and K. Zou, “EDA: Easy data augmentation techniques for boosting performance on text classification tasks,” *EMNLP-IJCNLP 2019 - 2019 Conf. Empir. Methods Nat. Lang. Process. 9th Int. Jt. Conf. Nat. Lang. Process. Proc. Conf.*, pp. 6382–6388, 2019, doi: 10.18653/v1/d19-1670.

[39] Y. Wang, Y. Hou, W. Che, and T. Liu, “From static to dynamic word representations: a survey,” *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 7, pp. 1611–1630, 2020, doi: 10.1007/s13042-020-01069-8.

[40] P. M. Brennan, J. J. M. Loan, N. Watson, P. M. Bhatt, and P. A. Bodkin, “Pre-operative obesity does not predict poorer symptom control and quality of life after lumbar disc surgery,” *Br. J. Neurosurg.*, vol. 31, no. 6, pp. 682–687, 2017, doi: 10.1080/02688697.2017.1354122.

[41] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, “Advances in pre-training distributed word representations,” *Lr. 2018 - 11th Int. Conf. Lang. Resour. Eval.*, no. 1, pp. 52–55, 2019.

[42] A. Aghaebrahimian and M. Cieliebak, “Hyperparameter tuning for deep learning in natural language processing,” *CEUR Workshop Proc.*, vol. 2458, 2019.

[43] S. Sharma, S. Sharma, and A. Anidhya, “Understanding Activation Functions in Neural Networks,” *Int. J. Eng. Appl. Sci. Technol.*, vol. 4, no. 12, pp. 310–316,

2020.

[44] J. Brownlee, “Better Deep Learning. Train Faster, Reduce Overfitting, and Make Better Predictions,” *Mach. Learn. Mastery With Python*, vol. 1, no. 2, p. 539, 2018.

[45] S. Joseph, “Australian Literary Journalism and ‘Missing Voices’: How Helen Garner finally resolves this recurring ethical tension,” *Journal. Pract.*, vol. 10, no. 6, pp. 730–743, 2016, doi: 10.1080/17512786.2015.1058180.

[46] J. C. Duchi, P. L. Bartlett, and M. J. Wainwright, “Randomized smoothing for (parallel) stochastic optimization,” *Proc. IEEE Conf. Decis. Control*, vol. 12, pp. 5442–5444, 2012, doi: 10.1109/CDC.2012.6426698.

[47] R. Hermawan, *Natural language processing with python*, vol. 1, no. 1. 2011. doi: 10.17509/ijal.v1i1.106.

[48] J. Feng and S. Lu, “Performance Analysis of Various Activation Functions in Artificial Neural Networks,” *J. Phys. Conf. Ser.*, vol. 1237, no. 2, 2019, doi: 10.1088/1742-6596/1237/2/022030.

[49] M. Vakili, M. Ghamsari, and M. Rezaei, “Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification,” no. January, pp. 0–13, 2020, [Online]. Available: http://arxiv.org/abs/2001.09636

[50] A. Shiri, “Introduction to Modern Information Retrieval (2nd edition),” *Libr. Rev.*, vol. 53, no. 9, pp. 462–463, 2004, doi: 10.1108/00242530410565256.

[51] M. L. Zhang and Z. H. Zhou, “ML-KNN: A lazy learning approach to multi-label learning,” *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, 2007, doi: 10.1016/j.patcog.2006.12.019.

[52] N. Xiong *et al.*, “Lecture Notes on Data Engineering and Communications Technologies 153,” vol. 53, no. 0, p. 6221, 2022.

[53] P. John, N. Woll, M. Gazaille, and W. Cardoso, “Using Grammar Checkers in an ESL Context : An Investigation of Automatic Corrective Feedback,” no. April, 2017.

[54] R. Rahman and G. Z. Islam, “An exploratory research on grammar checking of Bangla sentences using statistical language models,” no. February, pp. 3244–3252, 2020, doi: 10.11591/ijece.v10i3.pp3244-3252.

[55] J. Lee, “Deep Learning-Based Context-Sensitive Spelling Typing Error Correction,” pp. 152565–152578, 2020, doi: 10.1109/ACCESS.2020.3014779.

[56] M. P. Marcus, “Text Chunking using Transformation-Based Learning 1 Introduction,” pp. 82–94.

[57] L. Hu, Y. Tang, X. Wu, and J. Zeng, “Considering optimization of English grammar error correction based on neural network,” *Neural Comput. Appl.*, vol. 2, 2021, doi: 10.1007/s00521-020-05591-2.

[58] F. P. Putra, D. Enda, N. L. Baydikova, and Y. S. Davidenko, “A Research on Online Grammar Checker System Based on Neural Network Model A Research on Online Grammar Checker System Based on Neural Network Model,” 2020, doi: 10.1088/1742-6596/1651/1/012135.

[59] A. M. Gezmu, B. E. Seyoum, and A. Nürnberger, “Contemporary Amharic Corpus : Automatically Morpho-Syntactically Tagged Amharic Corpus,” pp. 65–70, 2018.

[60] V. Verma and S. K. Sharma, “Comparative analysis of Grammar Checkers of various Asian Languages,” no. 10, pp. 697–700, 2018.

[61] K. Amon, “Setswana Grammar Checker for Declarative Sentences using LSTM-Recurrent Neural Networks,” 2021.

[62] N. Bhirud, R. Bhavsar, and B. V. Pawar, “A Survey of Grammar Checkers for Natural Languages A S URVEY OF G RAMMAR C HECKERS F OR,” no. February 2019, 2017, doi: 10.5121/csit.2017.70905.

[63] W. Mulugeta and M. Gasser, “Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming”.

[64] L. Bopche, G. Dhopavkar, and M. Kshirsagar, “Grammar Checking System Using Rule Based Morphological Process for an Indian,” pp. 524–525.

[65] M. Gasser, “HornMorpho : a system for morphological processing of”.

[66] V. Goyal and G. S. Lehal, “Hindi Morphological Analyzer and Generator Hindi Morphological Analyzer and Generator,” no. January, 2008, doi: 10.1109/ICETET.2008.11.

APPENDICES

Appendix A: Kambaatissa part of speech's

Appendix Table 1. Kambaatissa Adjectives (Su'mcaakkisaanu)

Abba	Gaana	Muccuru	Hiilata	Abbata
Bareeda	Gamballa	Mutulu	Digibata	Bareedata
Biiza	Haraara	Naafa	Qeraa'rru	Biizata
Biishsha	Hifata	Ooga	Xilla	Biishshata
Busha	Hiyyeessa	Qallu	Kurata	Ellawwata
Dala	Igga	Wojju	Hemaasinchuta	Gaanata
Ellawwa	Kashala	Xaliga	Haraara	Dalata
Farra	Lunga	Xuma	Lanki	Bushata
Fayyisano	Lita	Daangoo	Bakkani	Gamballata
Maashshata	Baara	Qammaaxaa	Bashila	Farrata
Qoora	Barru	Daamaa	Bisima	Fayyata
Gida	Gadaalla	Qayixaara	Bitiraamit	Lungata
Geceeca	Kaara'a	Jii'ra	Shalalata	Kashalata
Qophphanaanchu	Qacu	Batinaashsha	Shokka	Iggata
Hila	Bulla	Hoolama	Horanka	Hiyyeessata
Qorqoraanchu	Gabbanchu	Qawunka	Danamurra	Hifatata
Muccuruta	Mutuluta	Naafata	Oogata	Qacuta
Qalluta	Wojjuta	Xaligata	Litata	Jii'rata

Appendix Table 2. Kambaatissa Verbs (Shoosawwakkata)

Aguxxamu	Agu	Birru	Wiitu	Geu
Aphphamu	Booradu	Buuhu	Soozu	Leu
Abbiccamu	Firiixu	Dassu	Kitimu	Sheeu
Aaqqaqqamu	Genu	Gaajju	Abbaasu	Qollo
Aassu	Gobu	Ootu	Kotu	Sholo
Daguxxamu	Itu	Qaafu	Zatu	Biixxu
Dooramu	Kaasu	Sutu	Galu	Heu
Eramu	Muddu	Anju	Baabu	Osa'llu

Hiiramu	Qasu	Cancu	Hoogu	Bau
Oo’llamu	Rosu	Darshu	Kulu	Ossau
Sarbamu	Tuuru	Gisu	Qooccu	Waalu
Sohu	Gobba	Gugu	Keenu	Anshu
Uujjaqqamu	Faisu	Hasu	Uumu	Qasu
Sohinuta	Oddiisu	Kushu	Ee’nnu	Qambu
Agudiisu	Uurrisu	Sazu	Gexu	Wixu
Aagisu	Caakkisu	Shuuru	Dooru	Tumu
Ballisu	Atootu	Yaaru	Goollu	Haxiidu
Booyyeesu	Bollchchu	Gassu	Foolooccu	Zarru

Appendix Table 3. Kambaatissa Adverb (Shoosawwicaakkisaannu)

Addaamoga	Bireechchi	Ekku	Wogge	Kaita
Leelan	Hikkae	Gibboomm	Teesu	Kaba
Ga’aata	Gidanoon	Hannuta	Xoonita	Hakkannenii
Higisa	Aaze	Ma	Barraha	Zakko
Higa	Shiina	Hakkada	Gassima	Qawunka
Bere	Aleen	Mii	Sozzima	Xabbeenaga
Kabar	Zakkoon	Hawanka	Soddarraron	Mooshsh
Aliinii woroonii	Korooxan	Iillanqaxee	Ankareen	Wudiin
Kazammanu	Sarbanchiin	Ali	Hezetu	Etaru
Bareen bareen	Dagudoon	Aazi	Aggana	Woruta
Baganka	Abbishsh	Aazeen	Wogga	Birita
Sakki nur	Horanii	Birita	Wollon wollo	Hawarron
Kanne	Horooman	Maraman	Hehezetoon	Hegeren
Hada	Higisata	Mereeroon	Horinka	Woro

Appendix B: Samples

correct_datakm

	text	classindex
0	Woqqa baee'u.	correct
1	Tiin ay galt?	correct
2	Sarbbeen wali!	correct
3	Aa wolo malax!	correct
4	Isu oo'aayyom.	correct
...	...	...
1313	Wona arrichchut xe□itata.	correct
1314	Rosaannu wonan hujatoou.	correct
1315	Wona xa□mmuta hujatant.	correct
1316	Daraartu wona beezzeechchuta.	correct
1317	An wonabbi roshsha minen fulloom.	correct

1318 rows × 2 columns

Appendix Figure 1. Correct sample dataset

	text	classindex
0	At kinu torreemm.	SVD
1	Ise buna kaasano.	SVD
2	At mini waalaamm.	SVD
3	Esa seelu yoonee.	SVD
4	An kanne egenaam.	SVD
...	...	...
1313	Woqqee ta□mmitoo caakkis!	SVD
1314	Qussi oddaqqii ta'mmoo yooba.	SVD
1315	Kuun ta□mmeennosi woqqaahaa.	SVD
1316	Egedo malaakka ta□mmitan kule'ee!	SVD
1317	Seera xabb asseenan ta□mme hasisano.	SVD

1318 rows × 2 columns

Appendix Figure 2. SVD sample dataset

	text	classindex
0	Abbat cilla ollo.	AND
1	Abban billawa murr!	AND
2	Abbati dubu murrema.	AND
3	Abbaati at esaachch.	AND
4	Abbaayyu abbaata laga.	AND
...	...	...
1313	Kambaatu minaadabiichch tordunqisi afoo ammana...	AND
1314	Danaamo haqqiichch danaamu illitii plaaneetaan...	AND
1315	Naoot yoommi minaadi gooni oosoo xalla abbata ...	AND
1316	Lamit biishshat harde oosuta kabar worqi bixxi...	AND
1317	Danaamu haqqiichch danaamit illit plaaneetaant...	AND

1318 rows × 2 columns

Appendix Figure 3. AND sample dataset

```
avd_datakm
```

	text	classindex
0	Aboni zakko wallo.	AVD
1	Cammelu zakku wallo.	AVD
2	Wo□lleeb zakki waallo.	AVD
3	Isu zakkoga waallee□u.	AVD
4	Buna zakkoon gaffeem.	AVD
...	...	...
1312	Afee insamuhaa maanaakkata caakkissaau.	AVD
1313	Maanaakkasi mahoomaga xaafantoo duuhaan.	AVD
1314	Barena hezzetti roshsha qaada qixxansoa.	AVD
1315	Xawaaqqaannu mato killihiichch waalltaau.	AVD

Appendix Figure 4. AVD sample dataset

```
       0.6392705 ,  0.07758418,  0.53851396, -1.030086  , -0.7801944 ],
      dtype=float32),
'leano': array([ 0.9683086 ,  0.8325378 , -0.3880957 ,  0.91662335,  0.82657206,
       0.35564616,  3.6520011 , -0.0820287 ,  0.6448017 , -1.7317939 ],
      dtype=float32),
'dagudu': array([ 0.05990609,  0.75869185,  0.94881666,  0.45544785,  0.85139817,
       2.136539  ,  1.0628406 ,  1.6261978 , -0.40177557, -1.4088486 ],
      dtype=float32),
'hikka': array([-0.40694535, -0.12432677,  0.00338644,  0.02719649, -0.17795824,
       0.8964655 ,  2.504012  , -1.079163  , -1.4455717 ,  0.39337903],
      dtype=float32),
'xaaf': array([ 0.09992545, -2.2485375 , -0.72820485,  0.76416326,  0.656265  ,
       0.10998404,  1.60717   , -0.13497153, -1.3362286 , -1.8565385 ],
      dtype=float32),
'woyyano': array([ 1.7451421e+00, -2.3075798e-01,  9.3491236e-03, -4.7241884e-01,
       2.4817152e-01, -3.1447315e-01,  1.7045097e+00,  7.5122304e-02,
       2.6198590e-04, -1.0654943e+00], dtype=float32),
'kul': array([ 0.19974162, -1.5137782 , -0.3741844 ,  0.47387996,  1.7760311 ,
      -0.7041481 ,  1.9136323 , -0.6984477 , -0.1795495 , -1.284715  ],
      dtype=float32),
'dudubu': array([ 0.00506582, -0.23918103,  1.2346426 ,  2.0765536 ,  0.65183145,
       0.89331234,  0.59383583, -0.32076967, -0.8534913 , -0.2338019 ],
      dtype=float32),
'gamba': array([ 0.12232224, -0.1495552 , -0.48295248,  0.6257177 , -0.48541456,
       1.0977491 ,  2.8558252 , -0.37205836, -0.39956626, -1.4880197 ],
      dtype=float32),
```

Appendix Figure 5. Sample word embedding