
**SCHOOL OF GRADUATE STUDIES**

# DESIGN SOCIAL EVENT EXTRACTION MODEL FROM AMHARIC TEXTS USING DEEP LEARNING APPROACHES

**MSc. THESIS**

**MIMI ADIMASU GODINE**

**NOVEMBER, 2025**

**WOLKITE, ETHIOPIA**

**Wolkite University**

**School of Graduate Studies**

# Design Social Event Extraction Model from Amharic Texts Using Deep Learning Approaches

**A Thesis Submitted to School of Graduate Studies, in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science and Engineering (Specialization: Computer Science)**

**Mimi Adimasu Godine**

**Major Advisor: Mesfin Abebe (Ph.D.)**

**Co-Advisor:Jemal A**

**November, 2025**

**Wolkite, Ethiopia**

# APPROVAL SHEET

## School of Graduate Studies

## Wolkite University

## Social Event Extraction Model from Amharic Text Using Deep Learning Approaches

Submitted by:

Mimi Adimasu Godine

Name of Student | Signature | Date

Approved by:

1. Dr. MESFIN ABEBE  02-04-2025

Major Advisors Name | Signature | Date

2. _____ _____ _____

Co- Advisors Name | Signature | Date

3. _____ _____ _____

Department Head | Signature | Date

4. _____ _____ _____

Name of Chairman, DGC | Signature | Date

5. _____ _____ _____

Name of Dean, SGS | Signature | Date

## WOLKITE UNIVERSITY

## SCHOOL OF GRADUATE STUDIES

We hereby certify that we have read and evaluated Thesis titled **Social Event Extraction Model from Amharic Text Using Deep Learning Approaches** Prepared under our guidance by Mimi Adimasu Godine.We recommend that the Thesis shall be submitted as fulfilling the requirements for the award of a MSc. degree in computer science and engineering.

Dr. MESFIN ABEBE 02-04-2025

Major Advisor signature date

_____ _____ _____

Co-Advisor signature date

As members of the Board of Examiner of the Master of Science Thesis open defense examination, we have read and evaluated this Thesis prepared by Mimi Adimasu Godine and examined the candidate. We hereby certify that; the thesis is accepted for fulfilling the requirements for the award of the degree of Master of Science (M.Sc.) in computer science and engineering.

1. Baye Y.(Ph.D.) _____ 02-04-2025
   Name of external examiner signature date
2. Worku .M( Assistant professor) _____ 02-04-2025
   Name of internal examiner signature date
3. _____ _____ _____
   Name of chairman signature date

Final approval and acceptance of the Thesis is contingent upon the submission of its final copy to the Council of Postgraduate Program (CPGS) through the candidate's department or school graduate committee (DGC or SGC).

# 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 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 provide that accurate and complete acknowledgement of the source is made. Request for permission for extended quotations from, or reproduction of, this thesis in whole or in part may be granted by the Head of the School or Department or the Dean of the School of Graduate Studies when in his or her judgement the proposed use of the material is in the interest of scholarship. In all other instances, however, permission must be obtained from the author of the thesis.

Name: Mimi Adimasu Godine  Signature: _____

Date:

School/Department: Computer Science and Engineering

# ACKNOWLEDGEMENT

Firstly, I would like to express my immense gratitude to the God for their guidance and blessings throughout this research endeavor. Words fail to capture the depth of my appreciation. Equally important, I extend my heartfelt thanks to Dr. Mesfin Abebe, my research advisor, whose invaluable support and guidance were instrumental in completing this research from beginning to end. My deepest gratitude goes to my family, whose unwavering support, prayers, patience, and encouragement were indispensable throughout this research journey. Furthermore, I am indebted to my friends, whose insights and information proved invaluable in the completion of this thesis.

Name: Mimi Adimasu Godine

Signature: _____

Date:

## LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| Bi-GRU | Bidirectional Gated Recurrent Unit |
| Bi-LSTM | Bidirectional Long Short-Term Memory |
| CBOW | Continuous Bag of Word |
| CNN | Convolution Neural Network |
| CRF | Conditional Random Fields |
| HAN | Hierarchical Attention Network |
| IE | Information Extraction |
| GNN | Graphical Neural Networks |
| GRU | Gated Recurrent Unit |
| LECM | Latent Event & Category Model |
| LSTM | Long Short-Term Memory |
| MLM | Masked Language Modeling |
| MRC | Machine Reading Comprehension |
| NER | Name Entity Recognition |
| NLP | Natural Language Processing |
| OIE | Open Information Extraction |
| POS | Part of Speech |
| RNN | Recurrent Neural Network |
| SMO | Sequential Minimal Optimization |
| SOV | Subject Object Verb |
| SVM | Support Vector Machine |
| TFIDF | Term Frequency Inverse Document Frequency |

## TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF EQUATIONS

# LIST OF FIGURES IN THE APPENDIX

# ABSTRACT

*Social events play a crucial role in capturing societal trends, public opinions, and cultural activities. Extracting and analyzing social events from Amharic text can provide valuable insights into various domains. The extraction of social events from Amharic text poses significant challenges due to the complexity of the language and the unstructured nature of user-generated content. This study aims to develop an effective social event extraction model for Amharic text using deep learning approaches. This study used Yem zone social event datasets, total dataset size is 4,738 event records. By evaluating various feature extraction techniques, including Fast Text, Bi-grams, and Tri-grams, we identify the most suitable methods for enhancing event extraction accuracy. We implement several deep learning models, including LSTM, Bi-LSTM, GRU, Bi-GRU, and Simple-RNN, and assess their performance in extracting event trigger words. The results indicate that the GRU and Bi-GRU models consistently outperform their LSTM and Bi-LSTM counterparts, particularly when utilizing Tri-gram features. Notably, the Bi-GRU model achieves the highest accuracy of 1.00, underscoring the benefits of a bidirectional approach in capturing contextual information. This research contributes to the advancement of Amharic language processing, offering insights that can support various applications such as cultural studies, disaster management, and crisis response. Additionally, we introduce a social event extraction corpus for the Amharic language, paving the way for future research in this area.*

*Keywords**:** Amharic text, social event extraction, deep learning, LSTM, Bi-LSTM, GRU, Bi-GRU*

# CHAPTER ONE

# INTRODUCTION

## 1.1. Background of the Study

In the era of digital communication and social media, the Amharic language has witnessed a significant surge in user-generated content, particularly on platforms like Face book, Twitter, and blogs. Amharic, the official language of Ethiopia, is widely spoken and written by millions of people[1]. Within this vast amount of Amharic text, social events play a crucial role in capturing societal trends, public opinions, and cultural activities. Extracting and analyzing social events from Amharic text can provide valuable insights into various domains, including social sciences, marketing, public opinion analysis, and disaster management[1].However, the extraction of social events from Amharic text poses significant challenges due to the complexity of the language and the unstructured nature of user-generated content. Traditional methods for event extraction often rely on manual annotation or rule-based approaches, which are time-consuming, labor-intensive, and limited in their ability to handle the nuances and variability of Amharic text. To overcome these limitations, researchers have turned to deep learning approaches, because deep learning can handle complicated patterns in huge datasets and automatically train hierarchical features, it is very beneficial for social event extraction in complex languages like Amharic. This eliminates the need for considerable human feature engineering. Deep learning architectures like LSTMs and GRU, in contrast to conventional machine learning models, are excellent at preserving context over longer sequences, which is essential for comprehending dependencies in natural language. Their excellent processing skills enable them to effectively handle unstructured and noisy data, and their scalability enables them to improve performance as data availability increases. Furthermore, training on small Amharic datasets is made effective by the use of pre-trained models like Fast text, which improve task-specific performance by utilizing existing knowledge of linguistic subtleties.

Deep learning models, such as recurrent neural networks (RNNs), convolution neural networks (LSTM), and transformer-based architectures, have demonstrated the ability to automatically learn complex patterns and hierarchical representations from textual data. Leveraging the power of these

deep learning techniques, researchers have started exploring their potential for social event extraction from Amharic text[3].

The objective of this research is to develop a social event extraction model using deep learning approaches that can automatically identify and extract relevant events from Amharic text. The proposed model will leverage the unique characteristics of the Amharic language, capturing its syntactic and semantic structures, as well as contextual information, to achieve accurate and comprehensive event extraction. By utilizing deep learning algorithms, the model aims to capture the dependencies and relationships between words, phrases, and entities in Amharic text, enabling researchers and practitioners to gain deeper insights into social phenomena within the Ethiopian context[4].This study investigated various deep learning architectures and techniques suitable for Amharic text processing, including recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) cells, Simple Recurrent Neural Network (RNN), Bidirectional Long Short Term Memory (Bi-LSTM), Gated Recurrent Unit (GRU), Bidirectional Gated Recurrent Unit (Bi-GRU) and Pre-trained feature extraction techniques for Amharic[5]. The model trained and evaluated on large-scale Amharic text datasets containing diverse social event types. The performance of the proposed model assessed using standard evaluation metrics such as precision, recall, and F1-score.The contributions of this research include advancing the state-of-the-art in social event extraction from Amharic text.

## 1.2. Statement of the Problem

The motivation of social event extraction from Amharic text using deep learning stems from the growing significance of Amharic social media. As Ethiopia's official language, Amharic has seen a surge in user-generated content across platforms like Face book and Twitter, providing valuable insights into cultural activities, public sentiment, and political dynamics. Despite this growth, there is limited research on natural language processing (NLP) tailored specifically for Amharic. Most existing NLP tools are designed for languages like English, creating a gap in effective text processing for Amharic. This study aims to address this gap by focusing on the extraction of social events from Amharic text through deep learning techniques, thereby contributing to the development of essential NLP resources. Moreover, Amharic's complex linguistic features present challenges for traditional event extraction methods. Deep learning approaches have shown promise in overcoming similar challenges in other languages. By applying these techniques, the research

aspires to create a robust model for social event extraction, which has practical applications in policy-making, cultural studies, and sentiment analysis, ultimately enhancing the understanding of Ethiopian society.

Social events are an important source of information about the activities, interactions, and relationships within a community. Extracting this information from text can provide valuable insights for social scientists, policymakers, and others. However, the task of automatically identifying and extracting or classifying social events from textual data, especially for low-resource languages like Amharic, poses significant challenges.

Amharic is the official working language of Ethiopia and it has 32.4 million native speakers[6]. Despite its importance, relatively little research has been done on natural language processing (NLP) for the Amharic language, including the extraction of structured information like social events from Amharic text. The lack of annotated datasets, specialized linguistic resources, and NLP tools for Amharic presents a major obstacle.

In recent years some researchers had done on event extraction such as Event Extraction from un structured Amharic Text done by Ephrem T et al. (2020) but, the researcher used traditional machine learning algorithm and rule-based techniques. But for deep extracting event from word feature deep learning techniques more preferable. In 2020 Platform for Event Extraction in Hindi done by Sovan.Kumar et at. But, didn't work event realis status classification and event coreference resolution. In 2020 Event Extraction and Representation Model from News Articles done by Abera Hordofa etal. These researchers used only machine learning algorithms. They did not used deep learning techniques. And in 2022, Balemlay Gebeyehu did Event Modeling from Orthodox Bible using Deep Learning Approach but he considered continuous bag of word (CBOW) feature engineering techniques. To take more feature engineering techniques like word2vec, TFID and fast text are very crucial to extract more features from the given text of words.

Furthermore, social events can be expressed in diverse ways in natural language, with varying levels of explicitness and contextual information. Developing robust methods to accurately identify and extract relevant events from Amharic text is an open research problem.

The goal of this research is to develop effective techniques for social event extraction from Amharic text. This will involve creating annotated datasets, designing and evaluating deep

learning models, and addressing the unique linguistic and resource challenges present in working with the Amharic language. The resulting system could enable new applications in social science research, event monitoring, and other domains for Amharic-speaking communities.

To find a solution to the above challenges, this study investigates and answers the following research questions.

**RQ1**: Which feature extraction technique is best fit for social event extraction?

**RQ2**: For the task of event extraction which deep learning model is suitable?

## 1.3. Objective of the Study

### 1.3.1. General Objective

The main objective of this research is to develop a deep learning model for identifying social events in Amharic text.

### 1.3.2. Specific Objectives

Creating a reliable model to extract social events from Amharic text is the goal of this research. Therefore, the following procedure must be followed to make this

- To collect dataset for the task of social event extraction.
- To annotate and prepare the collected dataset
- To preprocess the collected data and make suitable to the selected models
- To extract relevant feature from dataset using word embedding fast text and N-grams.
- To build event extraction model using LSTM, BI-LSTM, GRU, BI-GRU and Simple RNN.
- To measure the performance of the build model using accuracy, precession, recall and F1 score metrics

## 1.4. Significance of the Study

This study on social event extraction from Amharic text using deep learning approaches carries significance in several domains. Event extraction is a crucial task in natural language processing (NLP) with diverse applications across various domains. For instance, structured events can be utilized to expand knowledge bases, enabling logical reasoning and inference [7],[8]. Governments focus on event detection and monitoring to promptly respond to popular social events and manage public affairs effectively[9],[10] In the business and financial sector, event extraction aids companies in quickly identifying market responses to their products, enabling risk analysis and

trading suggestions[11].In the biomedical domain, event extraction plays a vital role in identifying changes in bimolecular state or interactions, as described in scientific literature, crucial for understanding physiological and pathogenesis mechanisms [12].

- ***Advancement of Amharic Language Processing:*** The development of a social event extraction model specifically tailored for Amharic text contributes to the advancement of Amharic language processing. By leveraging deep learning techniques, this study aims to address the linguistic complexities of Amharic, such as its rich morphology, syntactic phenomena, and unique linguistic features. The proposed model has the potential to serve as a foundation for future research and development of NLP tools and applications for the Amharic language, enabling more accurate and efficient processing of Amharic text in various domains.
- ***Enhanced Understanding of Amharic-Speaking Society:*** Extracting social events from Amharic text can provide valuable insights into societal dynamics, public sentiment, cultural trends, and other aspects of the Amharic-speaking community. The proposed model enables researchers, analysts, and decision-makers to gain deeper and more comprehensive understanding of the social landscape, facilitating evidence-based decision-making processes in various domains, including social sciences, marketing, public opinion analysis, disaster management, and policy-making.
- ***Enabling Cultural Studies and Preservation:*** Amharic language and culture hold immense historical and cultural significance. The extraction and analysis of social events from Amharic text contribute to the preservation and promotion of Amharic culture by providing insights into cultural activities, events, and traditions. Researchers can use the extracted social events to study cultural patterns, changes, and practices within the Amharic-speaking community, fostering a better understanding of Ethiopian society and facilitating cultural studies.
- ***Support for Disaster Management and Crisis Response:*** In times of crisis and disaster, social media platforms often become important sources of real-time information. The ability to extract social events from Amharic text using deep learning approaches can play a crucial role in disaster management and crisis response. By identifying and analyzing relevant events, such as emergency situations, public reactions, or calls for help, the proposed model can assist authorities and organizations in understanding the impact of

disasters, coordinating rescue efforts, and providing timely support to affected communities.

- ***Bridging the Language Processing Gap:*** The research on social event extraction from Amharic text contributes to bridging the gap in natural language processing techniques and resources for under-resourced languages. While most NLP research has focused on widely spoken languages, this study emphasizes the importance of developing effective NLP tools and models for languages like Amharic. The proposed deep learning-based approach can pave the way for further advancements in NLP research for under-resourced languages, promoting linguistic diversity and inclusivity in the field.
- ***Practical Applications and Industry Relevance:*** The successful development of a social event extraction model for Amharic text using deep learning approaches offers practical applications and industry relevance. The extracted social events can serve as a foundation for various downstream applications, including sentiment analysis, trend prediction, opinion mining, and targeted marketing. Industries can leverage these insights to make informed business decisions, understand consumer preferences, and tailor their products and services to the Amharic-speaking market.

## 1.5. Scope and Limitation of the Study

The scope of the study on social event extraction from Amharic text using deep learning approaches encompasses the development and evaluation of a model specifically designed to extract social events from Amharic text. The study aims to address the linguistic complexities of Amharic, leverage deep learning techniques, and provide valuable insights into societal dynamics within the Amharic-speaking community. However, there are certain limitations and boundaries that need to be considered. These include:

- ***Language Focus:*** The study focuses specifically on the Amharic language. While the proposed deep learning approaches and methodologies may have broader applicability to other languages, the evaluation and analysis are centered on the unique characteristics of Amharic text. The findings and conclusions of the study should be interpreted within the context of the Amharic language and may not be directly transferable to other languages without further adaptation and validation.

- ***Event Extraction Task:*** The study focuses on the task of social event extraction from Amharic text. While social events are an important aspect of text analysis, other NLP tasks, such as sentiment analysis, entity recognition, or topic modeling, are not directly addressed in this research. The scope is limited to the extraction and analysis of social events, neglecting other linguistic and semantic aspects that may coexist in the text.
- ***Deep Learning Approaches:*** The study specifically explores the application of deep learning approaches for event extraction from Amharic text. While deep learning has shown promising results in various NLP tasks, it is important to acknowledge that it may not be the only approach or the most suitable approach for every scenario. Other traditional machine learning techniques or rule-based methods may have their own advantages and could be explored as potential alternatives or in combination with deep learning approaches.
- ***Availability of Labeled Data:*** The scarcity of labeled data for event extraction from Amharic text is a significant challenge. The study seeks to address it by creating a new annotated dataset.

## 1.6. Organization of the Study

This study on the social event extraction model from Amharic text using deep learning approaches is organized into several chapters. Chapter 1 introduces the research topic and outlines its significance, objectives, and scope. Chapter 2 presents a comprehensive literature review covering social event extraction, natural language processing, deep learning approaches, and Amharic language processing. Chapter 3 describes the methodology employed, including data collection, preprocessing techniques, and the deep learning approaches used for event extraction. Chapter 4 focuses on the design and experimental implementation, discussing the setup, training process, and implementation details. Chapter 5 presents the experimental results and provides a thorough discussion of the findings. Finally, Chapter 6 concludes the study by summarizing the key findings, offering recommendations for future research, and discussing the broader implications of the study.

# CHAPTER TWO

# LITERATURE REVIEW

## 2.1. Overview of Natural Language Processing (NLP)

Since its inception in the 1970s, Natural Language Processing (NLP) has consistently evolved and expanded its applications [13]. As a subfield of Artificial Intelligence, NLP focuses on developing computational methods to automatically analyze and represent human language in its various forms. The accessibility and rapid implementation of powerful tools and techniques have fueled a surge in NLP research across diverse application areas in recent decades. These advancements enable computer systems to comprehend and manipulate natural language, allowing them to perform a wide range of tasks [14].The primary objective of natural language processing (NLP) is to enable computers and software to comprehend human language automatically. To achieve this, machines require a substantial amount of data known as a corpus to learn language rules and perform specific tasks. However, developing countries often lack the necessary resources and tools to adopt human language technologies, including NLP. Amharic, a Semitic language, is one such example of a resource language. Despite this, numerous researchers have made significant contributions in various NLP applications for the Amharic language, such as word stemming, document classification, part-of-speech tagging, information retrieval, and information extraction. To enhance tagging performance for Amharic, this is both under-resourced and linguistically complex. The approach involves segmenting words into morphemes with different parts-of-speech and exploring combinations of tag hypotheses[15].

## 2.2. Amharic Language

Amharic, a language with a semantic structure, is widely spoken across various regions of Ethiopia and holds official status as the working language for the Federal Democratic Republic of Ethiopia. In other words, Amharic is the recognized language used for official purposes by the Ethiopian government, and it is characterized by its emphasis on meaning and semantics.

Amharic, also known as አማርኛ, is the second most widely spoken Semitic language globally, after Arabic. It holds the status of being the official working language for the Ethiopian government. In Ethiopia, Amharic is spoken in various regions, including Amhara and the multi-

ethnic Southern Nations, Nationalities, and Peoples. Unlike Arabic and Hebrew, Amharic is written from left to right[16]. It serves as the primary language for communication within the Ethiopian Federal Government and some regional governments, and most official documents and news in the country are produced in Amharic. However, despite its significance, the Amharic language faces challenges in terms of electronic resources on the internet and limited work done for computer-based applications. The writing system used for Amharic is called Fidel or abugida (ፊደል), which was adapted from the extinct Geez language. The Fidel system consists of 33 orders, each with seven forms, resulting in a total of 231 unique Fidels (ፊደሎች). In computer systems, Amharic characters are represented using Unicode, which assigns a unique number to each character. However, Amharic is considered a low-resource language globally, lacking the necessary tools and resources for effective natural language processing techniques.

### 2.2.1. Morphology of Amharic Language

Morphology is a field of study that examines the structure and formation of words, encompassing processes such as inflection, derivation, and compound word formation. These processes involve the use of various affixes to create derivational and inflectional morphemes[17]. In the context of morphology, a morpheme refers to the smallest meaningful unit in a word, which can be a single phoneme or a combination of phonemes[18]. For instance, lets consider the noun ልጅ meaning child. By adding the morpheme -ነት, we can form the noun ልጅነት, which refers to childhood. Similarly, from the adjective ደግ meaning generous,we can derive the noun ደግነት, which signifies generosity. Additionally, by utilizing the root ሄደ,we can form the noun መሄድ, meaning go. Furthermore, from the infinitive verb መስበር meaning to break,we can derive the noun ስበር, which translates to break.

### 2.2.2. Word Classes of Amharic Language

The Amharic language exhibits a variety of word classes, including nouns, verbs, adverbs, pronouns, adjectives, and more.

- ***Amharic Nouns (ሰም):*** In Amharic, nouns can be classified as either simple or derived. Simple nouns include words like ወንበር(chair), ዛፍ(tree), and አፈር(soil).
  Derived nouns can be formed through compound words or by affixing specific vowels like ኛ and ኦ. Here are some examples:

- Noun + Noun =>ሸክላ+ ድስት =>ሸክላድስት
- Noun + Verbal Stems =>አየር+ ወለድ =>አየርወለድ

Nouns can also be derived by adding bound morphemes to existing nouns. For instance, ሀገር (country) can become ሀገረኛ by adding the morpheme ኛኛ.Within the Amharic noun class, there are masculine and feminine gender indicators. The masculine gender is used for words referring to males, while the feminine gender is used for words referring to females. In cases where the gender is not naturally male or female, the choice of gender tends to depend on the size or adorability of the entity. If the entity is small or adorable, the feminine gender suffix (-it or -yt) is used; otherwise, the masculine gender is employed. This expansion of gender markers helps to convey femininity in cases that would otherwise be masculine.Moreover, Amharic nouns can also be inflected for number, indicating singular or plural forms. For example:

- ***Singular:***ተማሪ(student), ሀኪም(doctors), እናት (mother)
- ***Plural:***ተማሪዎች(students), ሀኪሞች(doctors), እናቶች(mothers)

- ***Amharic Verb (ግሰ):*** In the Amharic language, any word that can be placed at the end of a sentence and can accept suffixes such as /ን/, /ና/, /ሽ/, and others. Verbs in Amharic are often derived from verbal roots by affixing the vowel -ኧ. For example, ን -ግ -ር can be transformed into ንኧግኧር, which is derived from the compound words of stems and verbs. Another example is ቀና + አለ = ቀናአለ, and ገደፍ + አደረገ = ገደፍአደረገ.
- ***Amharic Adjective (ቅፅል):***Adjectives in Amharic are a word class called ቅፅል. They are used to describe or qualify a noun and can also come before a noun, such as in the phrase ፈጣንልጅ (fast child), or after an adverb, like in በጣምፈጣን (very fast). Another characteristic of adjectives in Amharic is that when they are pluralized, the previous letter of the last letter of the word is repeated. For example, ቀጭንbecomes ቀጫጭን, ወፍራም becomes ውፋፍራምand ነጭbecomes ነጫጭ. Additionally, adjectives in Amharic can be derived from verbal roots by inserting vowels between consonants.

• ***Amharic Adverb (ተውሳከግሰ):***In the Amharic language, adverbs serve the purpose of enhancing the sentence by providing additional information or ideas to the verb[19]. For instance, words like በጣም, ሁልጊዜ, በትጠዋት, በፍጥነት, and በዝግታ, among others, can be used as examples[20].

• ***Amharic Preposition (መስተዋድድ):***In the Amharic language, a preposition is a word that comes before a noun and functions as an adverbial element expressing various relationships such as place, time, cause, and more. Prepositions cannot be modified by prefixes or suffixes, and they are never used to create new words. Examples of prepositions include ከ፣በ፣ላይ፣ስለ፣ጋር, and so on[21].

• ***Amharic Pronoun (ተውላጠስም):***In the Amharic language, pronouns, known as ተውላጠስም in Amharic, can be further categorized into different types. The first type is the deictic specifier, which includes words like ይህ, ያ, እሱ, እስዋ, እኔ, አንተ, and አንች. The second type is the quantitative specifier, which consists of words like አንድ, አንዳንድ, ብዙ, ጥቂት, and በጣም[21]. Lastly, there is the possession specifier, which includes words like የእኔ, የአንተ, የእሱ, and እነሱ, among others[22].

• ***Amharic Phrases (ሐረግ):***In the Amharic language, a phrase, known as ሐረግin Amharic, refers to a collection of words that conveys meaning but lacks complete sense when standing alone. It is always a part of a sentence, functioning as a group of words that often carries a specific idiomatic meaning. In linguistic analysis, a phrase is a group of words, which can even be a single word, such asና,that acts as a constituent within the syntax of a sentence, forming a single unit within the grammatical structure[23].

• ***Amharic Sentence (አረፍተነገር):***In Amharic, a sentence is referred to as አረፍተነገር and it consists of a group of words that conveys a complete meaning. The word order in Amharic sentences differs from English. Generally, the verb is placed at the end of the sentence, following the Subject / Object / Verb structure. For example, in the sentence Abebe [subject] eat [verb] his lunch [object], in Amharic it would be አበበ [subject] ምሳውን [object] በላ [verb].Sometimes, a sentence may resemble an incomplete phrase used to express something. Amharic sentences are categorized into two types: simple sentences and complex sentences. A simple sentence contains only one verb within a phrase. On the other hand, a complex sentence is formed by combining simple sentences with complex phrases[24].

The typical structure of an Amharic sentence follows a subject-object-verb (SOV) order. However, sentences may occasionally appear in an object-subject-verb (OSV) order. For instance, the sentence ውሻውድመቷንነከሳትis in SOV order, but it can also be written as ድመቷንውሻውነከሳትin OSV order. The placement of words in a sentence can alter its meaning,

except when the object of the sentence carries the object marker ን (-n).Amharic nouns do not have subject markers or affixes. However, the subject can be identified based on its position within the sentence. The suffix -ን (-n) is used as an object marker in Amharic.

## 2.3. Extraction of Information from Text

Information extraction (IE) is a domain within the realm of Natural Language Processing that focuses on identifying and extracting pertinent details from extensive textual content. This field involves the extraction of specific information from a vast collection of data, extending beyond just text to encompass various data formats, including speech. The objective of IE is to convert the input, be it text or other data forms, into structured and unambiguous data that adheres to a predetermined format[25].

While there are varying definitions of Information Extraction (IE) provided by different scholars, the core concept revolves around the extraction or mining of pertinent information from diverse application domains. The rationale behind this practice is that unstructured data presents challenges in terms of management, assessment, analysis, and utilization for decision-making purposes. The process of automatically extracting relevant information from extensive datasets involves several subtasks, including trigger identification and argument extraction (such as identifying people, places, companies, and physical objects). Moreover, the utility of Information Extraction extends beyond unstructured text processing and finds application in fields like media analysis, decision-making support, preventing data overload in the media, and education. In fact, Information Extraction can also be applied to other forms of unstructured sources, such as images and videos, due to its significance and broad scope.

Open-domain information extraction (Open IE) is an approach that facilitates the identification and extraction of domain-specific information from textual documents in an independent manner[26] Open-domain information extraction (Open IE) systems play a valuable role in tasks such as question answering, inference, and other information extraction tasks. These systems are developed using the traditional pattern-based method[27]. This study introduces a novel extraction paradigm that is dependent on domain indexing and can effectively handle the vast diversity and size of web corpora. The paradigm aims to facilitate the discovery of relations extracted from textual data that are specific to the domain being analyzed. To achieve automatic information extraction from a document collection, a rule-based approach is employed,

leveraging shallow syntax and lexical patterns. These patterns are hand-crafted based on parts-of-speech (POS) tagged texts[28].

The increasing availability of large volumes of online text presents new opportunities for natural language processing (NLP) systems to address the challenge of knowledge engineering. In this research, the focus was on developing an open information extraction system that can automatically extract relevant information from the web. The approach involved utilizing techniques such as morphology analysis, named entity recognition, part-of-speech tagging, and syntax analysis, which are considered subtasks within the fields of Chinese and English linguistics. The outcome of this study was the proposal of a model for open information extraction[29],[26].

In the Amharic language, multiple models for Information Extraction have been developed over different time periods. Information Extraction model specifically for Amharic texts[30]. Their research utilized 1200 news texts as training and testing data. The study involved various preprocessing steps, including character normalization and text categorization, along with the utilization of machine learning techniques to extract relevant information from unstructured Amharic texts. They employed different machine learning algorithms, including Naive Bayes and support vector machines, to achieve this goal. Ultimately, the sequential minimal optimization (SMO) classifier algorithm successfully categorized the information present in Amharic texts.

The extraction of information from unstructured and multidimensional big data presents significant challenges for information extraction (IE) techniques. As the volume of multifaceted, also known as multidimensional unstructured data, rapidly grows, traditional IE systems struggle to handle this overwhelming influx of unstructured big data. The sheer volume and diverse nature of big data necessitate improvements in the computational capabilities of IE systems. To effectively address these challenges, it is imperative to comprehend the strengths and limitations of existing IE techniques in terms of data pre-processing, data extraction and transformation, and representations suitable for handling vast volumes of multidimensional unstructured data. Numerous research studies have been conducted on IE, specifically addressing the challenges and issues associated with different data types, including text, image, audio, and video. These studies aim to overcome obstacles and refine IE techniques to cater to the unique characteristics of each data type within the context of big data[31]. Information extraction is closely associated

with event extraction, which involves the use of a template to identify and extract a set of identified incidents. In some cases, information extraction tasks begin with identifying text within a piece of information and then extracting the relevant information along with its corresponding type. Therefore, information extraction serves as a fundamental process for event extraction, as it enables the extraction of pertinent information from unstructured text, images, videos, and audio.

For instance, in a given text, it might be necessary to determine the types of events expressed and their respective timing. Name Entity Recognition (NER), Semantic Role Recognition, Entity Relation Recognition, Timex, and TimeLine Recognition are all essential components of event extraction[32].These tasks aim to identify and extract specific entities, roles, relations, temporal expressions, and timelines associated with events.

## 2.4. Extraction of Events from Text

On a daily basis and across different locations, numerous noteworthy events take place. However, these events are reported in diverse data formats such as images, text, videos, audio, and various unstructured forms. This lack of structure in the information poses challenges for users trying to comprehend and identify the relevant details within the reported events. It also requires media experts to invest significant time in analyzing and selecting valuable events to present to users.

Event extraction addresses this issue by automating the extraction of events from various media sources and creating a structured event framework for users. This process involves identifying and extracting relevant information from the different formats, organizing it into a coherent event structure, and presenting it in a more comprehensible manner to users. By automating this process, event extraction aims to make event-related information more accessible and efficiently deliver it to users.

Event extraction involves the application of natural language processing techniques to extract particular events, along with their types, participants, and attributes, from unstructured textual data[33]. This process enables the identification and extraction of specific events mentioned within the text, providing structured information about them. Event extraction finds applications in various tasks such as question answering, populating knowledge bases, information retrieval, document classification, and document summarization[34]. These applications leverage the

extracted event information to enhance tasks related to understanding, organizing, and summarizing textual content.

Scholars have categorized event extraction methods into two main groups: open domain and closed domain event extraction. Open-domain event extraction involves the identification and extraction of events from extensive unstructured text documents. It focuses on detecting events and extracting their associated arguments, such as words or phrases, without relying on a predefined schema that describes the structure or attributes of the events[35]. In other words, open-domain event extraction aims to capture events and their relevant details from text data without prior knowledge or constraints about the specific event types or their expected format.

## 2.5. Machine Learning and Deep Learning Techniques for Event Extraction

Event extraction extensively relies on the application of machine learning techniques to automatically detect and extract events from diverse data sources. These techniques enable the identification and extraction of events without manual intervention, contributing to the efficiency and accuracy of the process[36].

Scholars worldwide have categorized the existing event extraction methods into three main types: pattern-matching based[37],machine-learning based[38],and deep-learning based approaches[39]. The pattern-matching based methods rely on predefined patterns or rules to identify and extract events. Machine-learning based methods employ algorithms that learn from labeled data to automatically extract events. Deep-learning based methods utilize advanced neural network architectures to capture intricate patterns and dependencies for event extraction.

Event extraction and relation extraction often begin with the use of template matching, which involves the application of manually or automatically obtained templates. These templates are then employed in conjunction with different template matching algorithms to identify and extract information that satisfies the constraints defined by the templates[40].

In order to improve the effectiveness of multi-event type extraction systems, researchers introduced the concept of a cross-event model. This model leverages document-level information to enhance the pattern matching process. This approach was developed as a response to the limitations of existing event extraction systems, which primarily focused on extracting information at the phrase or sentence level. They used RBRB model, which enhances trigger word recognition

by incorporating inputs such as trigger words, sentence representations, and pattern features. This model employs regularization techniques to effectively extract trigger words, enabling the utilization of both pattern-based and representation-based information. As a result, the classification performance of the model shows significant improvement. Nonetheless, the process of manually constructing the template for this model is time-consuming, costly, and lacks portability[41].

The machine learning-based event extraction method involves selecting relevant features from textual information, constructing binary or multivariate classification models using machine learning algorithms, and subsequently performing event detection and argument extraction through classification. These methods typically have the ability to filter out non-event sentences from the text and employ various techniques to represent potential event instances through the fusion of multiple sources of knowledge[42]. Introduced a novel event extraction model that combines open information extraction and ontology to identify and extract events from social media texts written in the Portuguese language. The study employed a combination of rule-based and machine learning algorithms to detect and extract events from unstructured social media documents. A machine learning algorithm was utilized for event detection, while rules were employed for event extraction. The model aimed to effectively capture events within the social media domain by leveraging both automated learning and predefined rules specifically designed for extracting events from Portuguese texts[43].

As technology progresses and deep neural networks continue to evolve rapidly[44], the application of deep learning techniques has gained prominence in modeling intricate structures. These techniques have proven to be effective in numerous natural language processing (NLP) tasks, including event extraction. Specifically, event extraction methods utilizing convolutional neural networks (CNN), recurrent neural networks (RNN), and graphical neural networks (GNN) have emerged as popular research areas, capitalizing on the capabilities of deep learning models to tackle the complexities involved in event extraction[45].

The RNN model, introduced by the researchers, is a type of recurrent neural network (RNN) that incorporates bridge dependencies. This model leverages bidirectional RNNs to perform event extraction, utilizing relational graph information to identify event triggers and their associated elemental roles[46].

The researchers proposed a JMEE model that utilizes graph convolutional networks for event extraction. This model leverages entity mentions to aggregate convolutional vectors and combines graph-based convolutional vectors for entities mentioned within the current word and sentence. By incorporating this approach, the JMEE model aims to enhance the performance of event detection[47].

In addition, the researchers introduced Joint3EE, a deep learning model designed to address multiple tasks simultaneously[48]. This joint task model combines named entity recognition, trigger word recognition, and argument recognition. By sharing hidden representations among the three tasks, Joint3EE facilitates knowledge sharing and captures dependencies and interactions between tasks. This approach ultimately enhances the performance of event extraction by leveraging the benefits of joint learning[49].

Introduced the PLMEE model, which utilizes a pre-trained language model to extract event trigger words[50]. To handle the issue of role, overlap in role extraction tasks, the model employs a role prediction separation method. Additionally, the masked language modeling (MLM) functionality of the Bert model is employed to automatically generate labeled data, yielding promising results. While the pre-trained model captures more intricate interaction information, leading to reduced extraction errors, this approach overlooks the prior semantic information associated with element descriptions[51].

In 2020, Du et al. introduced a novel approach that transforms trigger word extraction and argument extraction into a question-and-answer format. By utilizing a question-and-answer task model, event extraction is completed without heavy reliance on entity class information from previous tasks, thereby mitigating the issue of error propagation in event detection. The study confirmed the advantages of the question-and-answer (Q&A) and machine reading comprehension (MRC) methods over traditional sequence annotation techniques. However, this approach overlooks the prior information embedded in historical question-and-answer pairs and the hierarchical dependencies within question-and-answer patterns. In this paper, we aim to enhance the performance of event extraction by developing an event extraction model based on multi-round question-and-answer patterns, building upon the findings of the aforementioned study[52].

## 2.6. Related Work

Several studies have explored event extraction and contributed valuable insights to the field. In June 2020, Abera Hordofa et al. presented a method for extracting events from Amharic news articles that combines machine learning techniques with an ontology-based approach. However, this study has limitations, such as not covering all sub-tasks in event extraction, including Argument Role Classification and Event Classification, and relying on handcrafted feature engineering, which can lead to inaccurate predictions. Similarly, Tadesse et al. (2020) employed a hybrid approach for event extraction from unstructured Amharic texts, combining machine learning with rule-based methods. However, this research lacked fundamental sub-tasks like event element extraction and event clustering, focusing primarily on temporal information.

In another study, Sahnoun et al. (2020) utilized open information extraction (OIE) and ontology to detect management change events, achieving favorable accuracy with an adaptive Conditional Random Fields (CRF) approach. Event extraction is influenced by language and domain, prompting researchers to target specific areas, such as biomedical texts. For instance, Ramponi et al. (2020) used a Support Vector Machine (SVM) to identify biomedical sentences containing events but faced challenges due to high data dimensionality.

Deep learning techniques, like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, have also been applied to event extraction. CNNs adapt well to text classification by utilizing word embeddings and convolutional layers. LSTMs, on the other hand, effectively capture long-term dependencies in sequential data, making them suitable for NLP tasks.

Studies by Xiang and Wang (2019) introduced unsupervised machine learning algorithms for clustering similar events in news articles, while Petroni et al. (2018) focused on disaster event extraction from social media and news reports. Other researchers, such as Kuila et al. (2018), combined CNN and Bidirectional LSTM for event detection and argument identification in Indian-language data. Overall, while significant progress has been made in event extraction methodologies, challenges remain, particularly in adapting techniques to low-resource languages like Amharic.

Table 2.1. Related Work

| ***Research Papers*** | ***Authors and Year*** | ***Methodology and Result Description*** | ***Research Gap*** |
|---|---|---|---|
| Event Extraction from Unstructured Amharic Text | Ephrem T et al. (2020) | Used supervised machine learning, handcrafted rules, and hybrid techniques. Total dataset consists of 659,848 words. Results: Naive Bayes (NB) achieved 91% weighted average, LIBSVM 80%, J48 89%, rule-based F1-score of 95%, and hybrid approach 97%. | The researchers employed traditional machine learning algorithms and rule-based techniques. For deeper event extraction from word features, deep learning techniques are more preferable. |
| Platform for Event Extraction in Hindi | Sovan Kumar et al. (2020) | Used BI-LSTM + CRF, BI-LSTM + Softmax, CNN, and Bi-GRU + CNN. Results: BI-GRU + CNN 86%, CNN 82%, BI-GRU + Self-Attention 87%. | Future research should investigate event realis status classification and event co reference resolution. |
| Event Extraction and Representation Model from News Articles | Abera Hordofa et al. (2020) | Used machine learning techniques, specifically a maximum entropy classifier. Achieved precision values of 67.1%, 69.1%, and 72% for event trigger identification, event element extraction, and event classification, respectively. | The researchers used only machine learning algorithms and did not employ deep learning techniques. |

| | | | |
|---|---|---|---|
| Event Extraction Based on Deep Learning in Food Hazard Arabic Text | Fouzi Harrag et al. (2018) | Algorithms used: RNN and LSTM. | The researchers used simple RNN and LSTM; they did not explore Bi-LSTM, GRU, or Bi-GRU. |
| Event Modeling from Orthodox Bible Using a Deep Learning Approach | Balemlay Gebeyehu (2022) | Used CNN and Bi-LSTM. Bi-LSTM achieved the best accuracy of 94.5%, CNN 91.5%, and combined CNN and Bi-LSTM 90.7%. The dataset consisted of 61,989 words. | The researcher considered continuous bag of words (CBOW) for feature engineering. However, other techniques such as Word2Vec and TF-IDF could enhance feature extraction. Additionally, to determine the best model for extracting events and triggers, more deep learning algorithms should be implemented. |

To address the identified research gaps, our study employs a comprehensive approach that utilizes advanced deep learning algorithms, including Simple RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU. Additionally, we incorporate feature extraction techniques such as Bigram, Trigram, and FastText pre-trained embeddings. By leveraging these methodologies, our research significantly enhances the extraction of event types and trigger words, outperforming traditional machine learning and rule-based techniques. This approach not only contributes to the existing body of knowledge but also provides a robust framework for future studies in event extraction across various languages and contexts. Due to the complexities of the Amharic language, along with the scarcity of datasets and the imbalance among the four classes, it is essential to select deep learning techniques to effectively extract meaningful insights from the limited available data.

Furthermore, choosing the appropriate feature extraction methods for event extraction is ideal; nevertheless, due to the intricacy of the language, the aforementioned feature extraction methods might not be able to handle terms that are currently out of vocabulary. FastText and the other well-known feature extraction methods are therefore more appealing as the model is hampered by overfitting.

# CHAPTER THREE

# METHODOLOGY

## 3.1. Introduction

This study focuses on developing social event extraction model from Amharic text using deep learning approach, it follows experimental approach and the tasks in this chapter include data collection, data preprocessing, relevant feature extraction, model building with deep learning technique and the last task in this chapter is model performance evaluation and experimental tools. Here is the general flow diagram depicted in figure 3.1.

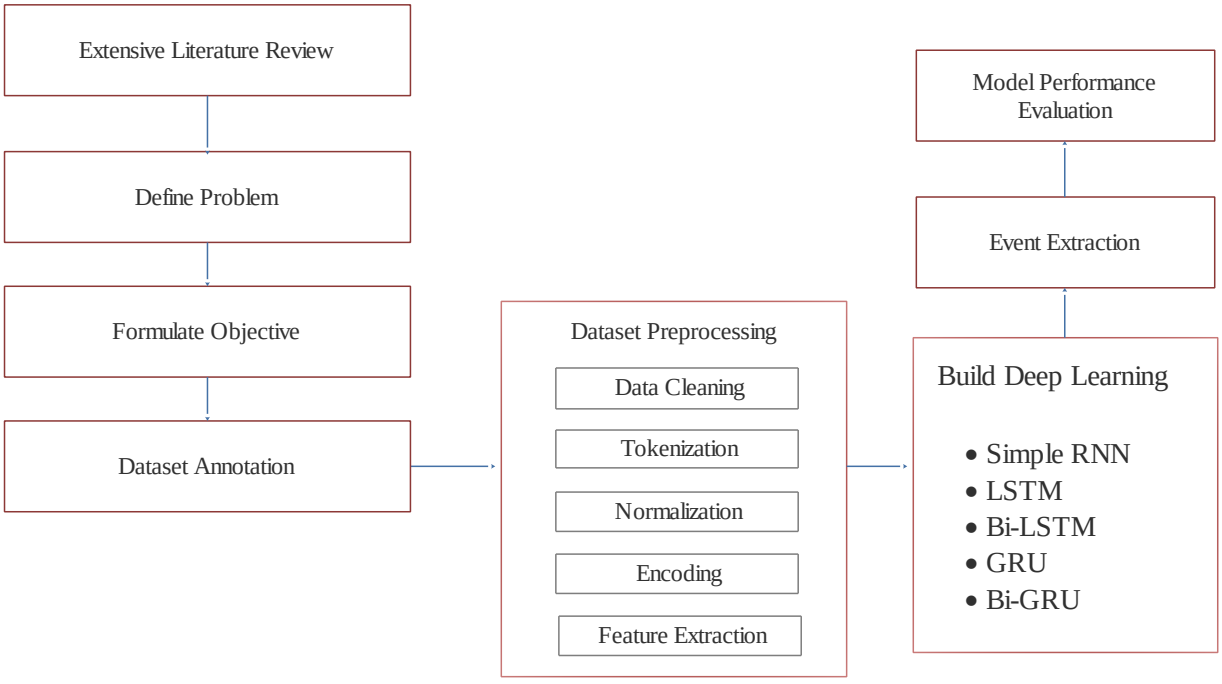



Figure 3.1. Research Flow Diagram

Figure 3.1 illustrated the research flow of the research process. In the initial sub-section, we conducted an extensive literature review, examining various research papers on event extraction. The second sub-section highlights the problems we identified based on the literature review. Moving on to the third sub-section, we established the research objectives, outlining both the general and specific goals of the study. The fourth sub-section involved the collection of relevant datasets from kebele social event registered data, which were used to create the Event dataset. In the fifth sub-section, we performed data preprocessing on the collected dataset to facilitate easy model training. Subsequently, in the sixth sub-section, we constructed deep learning models such as LSTM, BILSTM, Simple-RNN, GRU and BI-GRU. The seventh sub-section focused on event classification and extraction using the built models. Finally, in the eighth sub-section, we evaluated the performance of our models using metrics such as F1-score, precision, recall, and accuracy. Each step of the research process is discussed in detail as follows.

## 3.2. Data Collection

The social event extraction data was collected from the Yem Zone, located in the northwestern apex of the Southern Nations, Nationalities and Peoples Regional State (SNNPR) of Ethiopia. It is situated between the geographical coordinates of 7°37'N to 8°02'N and 37°40'E to 37°61'E. Covering a surface area of approximately 724.5 km², Yem is characterized by a diverse landscape that includes elevations ranging from 920 to 2,939 meters above sea level[66]. A large number of data collected from manually recorded books of the zone. When an individual is married, birth, death, and divorce their social event recorded by the zone officer. We also conducted this research by taking this filtered and recorded data. Upon registering the record, the individual provides information regarding the event type, location, date of the Event. Total size of data that we have collected for this study is 4,738 event records. the information, it was in the register book, the work was tiring as it required re-entering it into the computer and even the recorded data has not structure. First, we collected manually recorded event data and then entered the data to the computer using the Microsoft excel application. This task took us a very long time to prepare a lot of event data.

As shown in figure 3.2 below, the social event extraction data reveals a complex pattern of documentation across different years. In the late 1920s, there were 18 records in 1928 and 6 records in 1929, indicating a gradual increase in activity. The 1930s exhibited fluctuating numbers, peaking

in 1933 with 25 records, while 1939 and 1940 saw only 1 record each, suggesting a significant decline during this period. In the early 1940s, the years 1942 and 1943 experienced a rise in activity, with 14 and 18 records, respectively. However, this was followed by a notable drop to just 2 records in 1944, reflecting a decrease in documentation. The post-war years from 1945 to 1959 showed varied counts, ranging from 1 record in 1949 to a peak of 29 records in 1958, indicating a slow recovery or resurgence in event documentation. The data from the 1960s and 1970s is particularly rich, with the highest count of 94 records recorded in 1969 and a peak of 112 records in 1978, demonstrating increasing engagement in social events during this time. This upward trend continued into the 1980s, culminating in 126 records in 1988. The dataset reached remarkable heights with 943 records in 2014, marking one of its highest points. In 2012, there were 386 records, followed by 598 in 2013, and the all-time high of 943 in 2014, illustrating a significant surge in documentation. However, this count dropped significantly in the subsequent years, with only 1 record in 2015 and 7 in 2016, indicating a possible decline in event documentation or reporting. Lastly, the year 8181 contains 2 records, which may suggest a data entry error or an outlier, warranting further investigation. Overall, this analysis highlights the dynamic nature of social event documentation over time, reflecting both community engagement and potential reporting challenges.

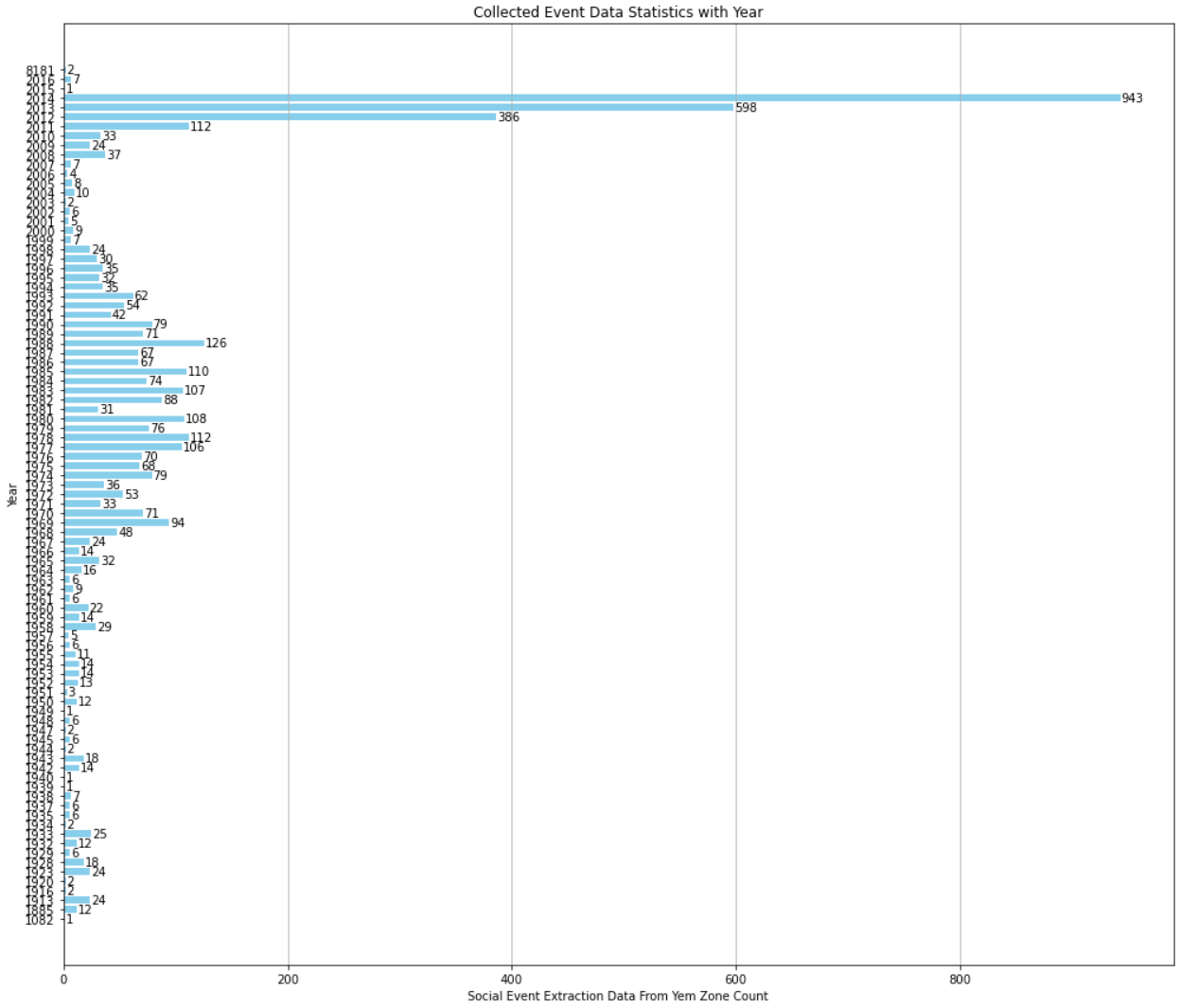



Figure 3.2. Social Event Extraction Dataset Statistics

### 3.2.1. Dataset Annotation

Dataset annotation plays a crucial role in event extraction tasks, as it involves labeling specific attributes or data features that are relevant to the extraction of events. During the annotation process, human annotators assign labels or tags to different elements within the dataset, indicating their role in event extraction. These annotations typically include attributes such as event triggers, arguments, event types, temporal information. By annotating these key data features, the dataset becomes a valuable resource for training and evaluating event extraction models, enabling the development of accurate and effective algorithms for automated event detection and classification.

## 3.3. Dataset Pre-Processing

Dataset preprocessing for event extraction involves various necessary tasks to clean and prepare the data for further analysis and modeling. Some of the essential preprocessing steps for event extraction include: Removing irrelevant or noisy data, such as HTML tags, special characters, or non-textual elements, to ensure the dataset consists of clean and readable text. Splitting the text data into individual tokens, typically words, to create a tokenized representation that facilitates further processing. Removing punctuation marks from the text, as they often do not carry significant meaning for event extraction and can introduce noise in the data. Standardizing the text by converting it to a consistent format, such as converting all the same sound characters to the same format, to ensure uniformity in the dataset. From the text that do not contribute much to the event extraction process.

## 3.4. Proposed Deep Learning Models

In recent years deep learning is another investigation in the machine learning (ML) field. It can capture successfully the internal structures of data and to characterize the data, it uses powerful modeling competences. For the meantime 2006, deep structured learning, or typically known as deep learning, has arisen as a new area of machine learning (ML) study. Over the past many years, the methods developed from deep learning analysis have already had methods developed from deep learning analysis have already been impacting a large variety of signal an impact on a wide range of signals and information processing work inside the ancient and the new, broadened scopes together with key features of machine learning (Machine Learning) and artificial intelligence (AI) [21]. Algorithms that have demonstrated potential in low-resource environments were taken into consideration to make sure our selected models could efficiently train from the scant annotated data for Amharic. Because of the intricacies of the Amharic language, we concentrated on deep learning algorithms that are able to comprehend context and manage sequential data. Our focus was on Convolutional Neural Networks (GRU) and Long Short-Term Memory (LSTM) networks. Since we only have a little amount of annotated data available for Amharic, we also selected algorithms that have

### 3.4.1. Long Short-Term Memory (LSTM)

LSTM is a type of recurrent neural network (RNN) that is well-suited for processing sequential data. It addresses the vanishing gradient problem and can effectively capture long-term dependencies. LSTM has been widely used in event extraction tasks to model the context and relationships between words.

**Algorithm 1:** Train and evaluate Long Short-Term Memory Model

**Start**

***Input:*** Normalized sequenced training data

***Output:*** Extraction of Event type and Event Trigger Words

***Initialization:*** epoch, batch size, lstm_unit, dropout rate

***Step1:*** add LSTM Layer with defined lstm_units and input data
add dropout regularization
add dense layer with defined output dimension and sigmoid activation function

***Step2:*** Compile LSTM model with binary-cross entropy loss function and Adam optimizer

***Step3:*** Fit the training data to LSTM Model with initialized hyper parameter
***While***maximum iteration not reached***do***
Train LSTM Model

***Step4:*** Evaluate LSTM model with testing data

***Step5:*** Predict unseen data by LSTM Model

***Step6:*** Measure the LSTM Model by Accuracy, f1-score

**Stop**

### 3.4.2. Simple Recurrent Neural Network (Simple-RNN)

A Simple Recurrent Neural Network (RNN) is a type of neural network designed for processing sequential data, where the output from previous steps is fed as input to the current step. It maintains a hidden state that captures information about previous inputs, enabling it to recognize patterns

and dependencies over time. Simple RNNs are commonly used in tasks like time series prediction, natural language processing, and event extraction.

**Algorithm 2:** Train and evaluate Simple Recurrent Neural Network (RNN) Model

**Start**

***Input:*** Normalized sequenced training data

***Output:*** Extraction of Event type and Event trigger words

***Initialization:*** epoch, batch size, rnn_unit, dropout rate

***Step1:*** add RNN Layer with defined rnn_units and input data
add dropout regularization
add dense layer with defined output dimension and Softmax, relu activation function

***Step2:*** Compile RNN model with binary-cross entropy function and Adam optimizer

***Step3:*** Fit the training data to RNN Model with initialized hyper parameter
***While***maximum iteration not reached***do***
Train RNN Model

***Step4:*** Evaluate RNN model with testing data

***Step5:*** Predict unseen data by RNN Model

***Step6:*** Measure the RNN Model by Accuracy, f1-score

**Stop**

### 3.4.3. Bidirectional LSTM (BiLSTM)

BiLSTM extends the capabilities of LSTM by incorporating information from both past and future contexts. It processes the input sequence in both forward and backward directions, capturing dependencies from both temporal directions. BiLSTM has shown effectiveness in capturing contextual information for event extraction.

**Algorithm 2:** Train and evaluate Bidirectional Long Short Term Memory Model

**Start**

***Input:*** Normalized sequenced training data

***Output:*** Extraction of Event type and Event trigger words

***Initialization:*** epoch, batch size, bilstm_unit, dropout rate

***Step1:*** add Bi-LSTM Layer with defined bilstm_units and input data
add dropout regularization
add dense layer with defined output dimension and Softmax, reluactivation function

***Step2:*** Compile Bi-LSTM model with binary-cross entropy function and Adam optimizer

***Step3:*** Fit the training data to Bi-LSTM Model with initialized hyper parameter
***While***maximum iteration not reached***do***
Train Bi-LSTM Model

***Step4:*** Evaluate Bi-LSTM model with testing data

***Step5:*** Predict unseen data by Bi-LSTM Model

***Step6:*** Measure the LSTM Model by Accuracy, f1-score

**Stop**

### 3.4.4. Gated Recurrent Unit (GRU)

The Gated Recurrent Unit (GRU) is a component within a particular type of recurrent neural network designed to establish connections across a sequence of nodes, primarily for tasks involving memory and grouping, such as in speech recognition. GRUs are instrumental in addressing the vanishing gradient problem, a prevalent challenge encountered in recurrent neural networks, by adapting input weights within the neural network.

**Algorithm 3:** Train and evaluate Gated Recurrent Unit Model

**Start**

***Input:*** Normalized sequenced training data

***Output:*** Extraction of Event type and Event trigger words

***Initialization:*** epoch, batch size, gru_unit, dropout rate

***Step1:*** add GRU Layer with defined gru_units and input data
add dropout regularization

add dense layer with defined output dimension and Softmax, relu activation function

***Step2:*** Compile GRU model with binary-cross entropy function and Adam optimizer

***Step3:*** Fit the training data to GRU Model with initialized hyper parameter

***While*** maximum iteration not reached ***do***

Train GRU Model

***Step4:*** Evaluate GRU model with testing data

***Step5:*** Predict unseen data by GRU Model

***Step6:*** Measure the GRU Model by Accuracy, f1-score

**Stop**

### 3.4.5. Bidirectional Gated Recurrent Unit

The Bidirectional Gated Recurrent Unit (Bi-GRU) is a component within a particular type of recurrent neural network designed for tasks in machine learning, utilizing connections across a sequence of nodes. What sets Bi-GRU apart from GRU is its ability to grasp context from both the forward and backward directions, enhancing its understanding of sequences.

**Algorithm 4:** Train and evaluate Bidirectional Gated Recurrent Unit Model

**Start**

***Input:*** Normalized sequenced training data

***Output:*** Extraction of Event type and Event trigger words

***Initialization:*** epoch, batch size, bi-gru, dropout rate

***Step1:*** add Bi-GRU Layer with defined bi-gru and input data

add dropout regularization

add dense layer with defined output dimension and Softmax, relu activation function

***Step2:*** Compile Bi-GRU model with categorical cross entropy loss function and Adam optimizer

***Step3:*** Fit the training data to Bi-GRU Model with initialized hyper parameter

***While*** maximum iteration not reached ***do***

Train Bi-GRU Model

***Step4:*** Evaluate Bi-GRU model with testing data

***Step5:*** Predict unseen data by Bi-GRU Model

***Step6:*** Measure the Bi-GRU Model by f1 score, precision and accuracy

**Stop**

## 3.5. Classification Metrics

To assess how well our model is performing, various classification metrics are available, including confusion matrices, accuracy, precision, recall, and the F1-score.

### 3.5.1. Confusion Metrics

The confusion matrix is employed to depict the accuracy of the model. Basic terms in confusion metrics described below.

- ***True Positive (TP):*** The actually correct Event label classified or predicted as correctly.
- ***True Negative (TN):*** Actually, non-event label predicted as Real non language label.
- ***False Positive (FP):*** Incorrect event label classified as positively.
- ***False Negative (FN):***The correct event label predicted as negatively.

### 3.5.2. Accuracy

Accuracy is determined by adding the number of true positives and true negatives and then dividing this sum by the total of true positives, true negatives, false negatives, and false positives[67].

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \quad 3.1$$

### 3.5.3. Precision

Precision is computed by dividing the number of true positives by the total of true positives and false positives[67].

$$Precision = \frac{TP}{TP + FP} \quad 3.2$$

### 3.5.4. Recall

Recall is determined by dividing the number of true positives by the total of true positives and false negatives[67].

$$Recall = \frac{TP}{TP + FN} \quad 3.3$$

### 3.5.5. F1-Score

The F1-score is computed by taking two times the product of precision and recall and dividing it by the sum of precision and recall[67].

$$\text{F1} - \text{Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad 3.4$$

## 3.6. Experimentation Tools

### 3.6.1. Python

Python is a versatile programming language that is object-oriented, interpreted, and suitable for a wide range of purposes. It has powerful natural language processing packages and text processing. It has a huge standard library, which provides rich functions and modules. We used Python programming language to implement the social event extraction model.

### 3.6.2. Anaconda

Anaconda is enterprise platform for data science, which contains python for machine learning, deep learning and data science tasks. This is a distribution of the R and Python programming languages tailored for applications in predictive analysis, machine learning, deep learning, and data science.

### 3.6.3. Jupiter Notebook

Jupyter Notebook is a web-based application used for generating and distributing computational documents. It contains live code, equations, narrative text and visualizations. It is additionally employed for tasks such as data cleansing, numerical simulation, data transformation, statistical modeling, data visualization, machine learning, and various other functions.

### 3.6.4. Libraries

First, we decided to install all necessary libraries and packages. These Libraries are described as follow:

- ***Numerical Python (NumPy)*:** is popular for scientific computing tasks. This library provides multi-dimensional metrics and arrays with a collection of mathematical functions. NumPy is an essential tool for numerical tasks in Python.
- ***Pandas***: stands as an open-source Python library that serves as a valuable resource for conducting data analysis and handling data manipulation tasks. Notably, it offers a versatile array of data analysis tools and presents a user-friendly data structure capable of accommodating diverse data types.
- ***Scikit-learn***: scikit-learn is a machine learning library that is open-source and built using Python that offers a comprehensive suite of algorithms and utilities catering to diverse machine learning tasks such as classification, clustering, regression, model selection, and data preprocessing. Notably, this library is underpinned by foundational Python scientific libraries like SciPy, Matplotlib, and NumPy. In essence, scikit-learn not only streamlines the creation of robust and efficient machine learning models but also streamlines the entire machine learning workflow, simplifying the intricate processes involved in model development and evaluation.
- ***TensorFlow*:** It is a machine learning tool that is open-source and deep learning framework, can be utilized for code-mixed language modeling. It offers a range of tools and APIs for building and training neural network models.
- ***Keras*:** Keras is a Python-based open-source deep learning framework that offers a high-level and user-friendly interface for creating and training machine learning and deep

learning models. It is designed to be easy to use, modular, and flexible, enabling researchers to swiftly experiment and iterate on their machine learning projects.

- ***Matplotlib*:**is widely used in the systematic, data analysis, and visualization communities due to its functionality, flexibility and integration with different python packages. For creating professional quality visualization and plots matplotlib provides powerful toolset.
- ***Seaborn*:**is popular in statistical analysis communities and data science. It has the ability to produce aesthetically informative visualizations. It is powerful toolkit for meaningful visualizations. Seaborn complements matplotlib by simplifying statistical plots and offers additional styles.
- **Genism:** is a term that refers to discrimination or prejudice based on genetic traits or characteristics. It can manifest in various forms, including societal biases against individuals with certain genetic conditions, perceived genetic superiority or inferiority, or the ethical implications of genetic engineering and modification. The concept raises important discussions about the moral and social responsibilities surrounding genetic information and its potential impact on identity, health care, and societal norms.
- **Word Cloud:** is a visual representation of text data, where the size of each word indicates its frequency or importance in a given context. Words that appear more frequently are displayed in larger fonts, while less common words are shown in smaller fonts. Word clouds are often used to summarize key themes from a body of text or to visualize responses in surveys.
- **A regular Expression (regex)**: is a collection of functions and tools that allow you to work with regular expressions for pattern matching in strings. Regular expressions are powerful for searching, extracting, and manipulating text based on specific patterns.

# CHAPTER FOUR

# 4. DESIGN AND IMPLEMENTATION OF SOCIAL EVENT EXTRACTION MODELS

In this chapter shows how the proposed model designed and implemented with actual implementation code and description.

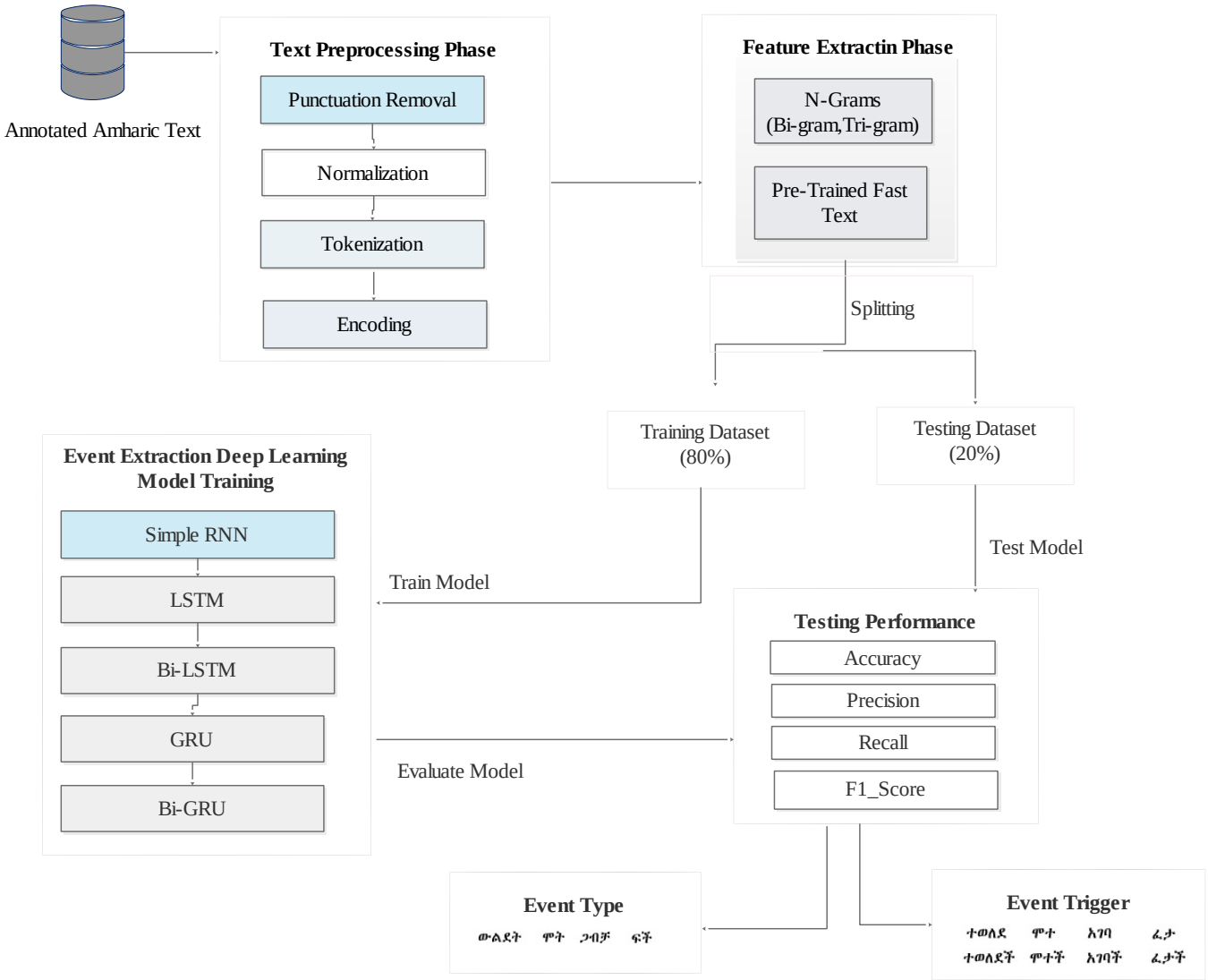



Figure 4.1. Proposed Social Event Extraction Modeling

In the social event extraction process, as depicted in Figure 4.1, the first step is to collect the dataset from Yem Zone, the collected data un structured, it needs processing. This study used different

pre-processing techniques to pre-process the data like cleaning, normalizing, tokenizing and encoding techniques. This involves removing punctuation, normalizing the Amharic characters, tokenizing the sentences into words, and removing stop words. Additionally, encoding using a Label encoder is performed to handle the nuances of the Amharic language, where certain characters have the same sound, such as {ሃኀኃሐኻሀ}, {ሠሰ}, {ዓአዐኣ}, {ጸፀ}.so, we normalized in to one uniform character, this process depicted in Figure 4.5.After the preprocessing stage, feature engineering is carried out using techniques like word embedding using pre-trainedfast-text, Ngrams (Bigram and Trigram). The preprocessed and engineered data is then divided into training (80%) and testing (20%) sets, and the input is converted into a three-dimensional shape that can be easily fed into the models. The next step involves training various Recurrent Neural Network (RNN) models, including Long Short-Term Memory (LSTM), Bidirectional-Long Short-Term Memory (BiLSTM), Gated Recurrent Unit (GRU), Simple Recurrent Neural Network(SimpleRNN), and Bidirectional Gated Recurrent Unit (Bi-GRU), using the training dataset. These models are then used to make predictions on the test dataset. Finally, the performance of the trained models is evaluated using metrics such as accuracy, F1-score, precision, and recall. This evaluation helps assess the model's effectiveness in extracting relevant social events from the input data. The key steps in this process are data collection, preprocessing, feature engineering, model training, and model evaluation, all of which are aimed at developing robust and accurate social event extraction capabilities.

## 4.1. Data Loading

Before doing any preprocessing first the dataset load or available to the python environment. To do so pandas DataFrame helps to load the dataset.

```
# Load the dataset
dataset = pd.read_csv('event/extract/event_data/event.csv')
dataset = shuffle(dataset)
dataset.head()
```

Figure 4.2. Dataset Loading

As shown in the above Figure 4.2. loading the dataset using pandas' data frame, in the panda's data frame there is a method read csv used to load the data by taking parameter of the dataset location path.

Table 4.1. Sample Dataset

| # | *Text* | *event_Triger* | *event_Type* |
|---|---|---|---|
| 1414 | ተወዳጅንጉሴገዝሙየካቲት 28 2014 ዓ.ምየምተወለደ | ተወለደ | ውልደት |
| 4575 | ጊሣፎጊድጋሐምሌ 10 1942 ዓ.ምትግርአረፈ | አረፈ | ሞት |
| 2624 | ዳዊትዘካናኖጥቅምት 25 2013 ዓ.ምየምአረፈች | አረፈች | ሞት |
| 2904 | ሙላቱሙላቱቀልብመጋቢት 12 1966 ዓ.ምየምተወለደ | ተወለደ | ውልደት |
| 643 | መስፍንዘውጋሚልክግንቦት 7 1979 ዓ.ምየምተወለደ | ተወለደ | ውልደት |
| 4027 | ሰለሞንመጫዱላጥቅምት 1 2014 ዓ.ምየምተወለደ | ተወለደ | ውልደት |

### 4.1.1. Dataset Description

Data collection technique briefly discussed in chapter 3, but this section about to present dataset features high lightly. The sample dataset depicted in the figure 4.1.The social event extraction datasets have the following features.

- ***Event Name:*** The name of social event name
- ***Event Date-Time:*** Timestamp of the event took place
- ***Event Trigger:*** The words that indicate the event
- ***Event Type:*** The type or the class of event
- ***Event Location:*** it shows where the event took place


Figure 4.3. Frequency of words in the dataset

However, the visualized data in the word cloud contain stop words, it is evident that stop words are prevalent in authentic event text, which might serve as a crucial feature for enhancing the performance of deep learning models in extraction tasks. In light of this observation, to mitigate the potential negative impact of certain preprocessing methods, we opted to exclusively utilize alternative techniques such as data cleaning, normalization, and tokenization.

## 4.2. Dataset Pre-Processing

In dataset preprocessing phase any missing values, noises cleaned, and transform the raw data in to machine understandable format mean that the categorical values change into numbers for this, we used Label encoder.

### 4.2.1. Cleaning

Cleaning is the process of removing special characters, emojis, extra white spaces, English words, Arabic numerals, and Ethiopic numbers from the event extraction dataset.

```
def process_text(text):
    special_chars_pattern = r'[,".:)(\-!|;\'$&/\[\]>%=#*+\\•~@£·_{©^®`<→°€™›♥←xÿ"'Â▒½]'
    emojis_pattern = r'😁|😲|🏇|🥳😁|🤢|😥|🤠|🤡'
    tabs_spaces_pattern = r'\s+'
    amharic_pattern = r'[\u1200-\u137F]'
    # Remove special characters
    text = re.sub(special_chars_pattern, '', text)
    # Remove emojis
    text = re.sub(emojis_pattern, '', text)
    # Remove tabs and extra white spaces
    text = re.sub(tabs_spaces_pattern, ' ', text)
    # Remove Amharic characters
    text = re.sub(amharic_pattern, '', text)
    return text.strip()
```

Figure 4.4. Dataset Cleaning Sample Code

The above sample code shows to remove special characters, emojis, extra white spaces, English words, Arabic numerals, and Ethiopic numbers from event extraction dataset.

### 4.2.2. Character Normalization

Amharic language is morphologically complex language, specifically in verb construction it adds in expected prefix and suffixes. For instance, ሰርቱዋል and ሰርቱአል should be normalized to ሰርቷል.

besides that, another thing like the words have the same sound but use different letters likeጸሀይ,ጸሃይ, and ጸሓይ should be normalzed to ፀሀይ.

```
def normalize_char_level_missmatch(input_token):
    if isinstance(input_token, str):
        rep1=re.sub('[ሃኅኃሐሓኻ]','ሀ',input_tok
        rep2=re.sub('[ሑኁዅ]','ሁ',rep1)
        rep3=re.sub('[ኂሒኺ]','ሂ',rep2)
        rep4=re.sub('[ኌሔኼ]','ሄ',rep3)
        rep5=re.sub('[ሕኅ]','ህ',rep4)
        rep6=re.sub('[ኆሖኾ]','ሆ',rep5)
        rep7=re.sub('[ሠ]','ሰ',rep6)
        rep8=re.sub('[ሡ]','ሱ',rep7)
        rep9=re.sub('[ሢ]','ሲ',rep8)
        rep10=re.sub('[ሣ]','ሳ',rep9)
        rep11=re.sub('[ሤ]','ሴ',rep10)
        rep12=re.sub('[ሥ]','ስ',rep11)
        rep13=re.sub('[ሦ]','ሶ',rep12)
        rep14=re.sub('[ዓኣዐ]','አ',rep13)
        rep15=re.sub('[ዑ]','ኡ',rep14)
        rep16=re.sub('[ዒ]','ኢ',rep15)
        rep17=re.sub('[ዔ]','ኤ',rep16)
        rep18=re.sub('[ዕ]','እ',rep17)
        rep19=re.sub('[ዖ]','ኦ',rep18)
        rep20=re.sub('[ጸ]','ፀ',rep19)
        rep21=re.sub('[ጹ]','ፁ',rep20)
        rep22=re.sub('[ጺ]','ፂ',rep21)
        rep23=re.sub('[ጻ]','ፃ',rep22)
        rep24=re.sub('[ጼ]','ፄ',rep23)
        rep25=re.sub('[ጽ]','ፅ',rep24)
        rep26=re.sub('[ጸ]','ፆ',rep25)
        rep27=re.sub('(ሉ[ዋአ])','ሏ',rep26)
        rep28=re.sub('(ሙ[ዋአ])','ሟ',rep27)
        rep29=re.sub('(ቱ[ዋአ])','ቷ',rep28)
        rep30=re.sub('(ሩ[ዋአ])','ሯ',rep29)
        rep31=re.sub('(ሱ[ዋአ])','ሷ',rep30)
        rep32=re.sub('(ሹ[ዋአ])','ሿ',rep31)
        rep33=re.sub('(ቁ[ዋአ])','ቋ',rep32)
        rep34=re.sub('(ቡ[ዋአ])','ቧ',rep33)
        rep35=re.sub('(ቹ[ዋአ])','ቿ',rep34)
        rep36=re.sub('(ሁ[ዋአ])','ኋ',rep35)
        rep37=re.sub('(ኑ[ዋአ])','ኗ',rep36)
        rep38=re.sub('(ኙ[ዋአ])','ኟ',rep37)
        rep39=re.sub('(ኩ[ዋአ])','ኳ',rep38)
        rep40=re.sub('(ዙ[ዋአ])','ዟ',rep39)
        rep41=re.sub('(ጉ[ዋአ])','ጓ',rep40)
        rep42=re.sub('(ደ[ዋአ])','ዷ',rep41)
        rep43=re.sub('(ጡ[ዋአ])','ጧ',rep42)
        rep44=re.sub('(ጩ[ዋአ])','ጯ',rep43)
        rep45=re.sub('(ጹ[ዋአ])','ጿ',rep44)
        rep46=re.sub('(ፉ[ዋአ])','ፏ',rep45)
        rep47=re.sub('[ቊ]','ቁ',rep46)
        rep48=re.sub('[ኵ]','ኩ',rep47)
        return rep48
    else:
        return input_token
```

Figure 4.5. Sample code character normalization

So, above sample code shows to normalize Amharic characters that have the same sound but different structure, a character normalization function, normalize_char_level_missmatch, aimed at enhancing the preprocessing of text data for social event extraction tasks. By systematically replacing various orthographic variations of characters in the Ge'ez script with their canonical forms, this function ensures consistency in the text, which is crucial for accurate analysis. When applied to the Processed Text column of a dataset, it generates a cleaner representation of the text, thereby improving the reliability of downstream tasks such as event type identification and trigger word detection. The resulting Data Frame, which includes both the normalized text and corresponding event types, facilitates structured data access, enhancing the model's ability to accurately extract and classify social events. This preprocessing step is vital for minimizing errors

caused by character mismatches, ultimately leading to better performance in natural language processing tasks related to event extraction.

### 4.2.3. Tokenization

Tokenization the process of text tokenization is being carried out using the Keras Tokenizer class, which is a fundamental step in event extraction process. Firstly, an instance of the Tokenizer class is initialized to facilitate the conversion of textual data into numerical sequences. By calling the fit_on_texts () method on the datasets Text column, the tokenizer builds an internal vocabulary based on the words present in the text data.

```
# Create an instance of Tokenizer
tokenizer = Tokenizer()
tokenizer.fit_on_texts(combined_data)
```

Figure 4.6. Tokenization Sample Code

### 4.2.4. Encoding

Encoding the utilization of label encoding to convert non-integer categorical labels into integer representations, a pivotal preprocessing step in deep learning tasks dealing with categorical data. Through the instantiation of a Label Encoder object, the categorical labels in the event Type column of the dataset are transformed into corresponding integer values via the fit_transform () method. This process assigns a unique integer to each distinct label present in the column, enabling the conversion of qualitative data into a numerical format that deep learning algorithms can effectively interpret and utilize for tasks such as classification and regression. By converting non-integer labels into integers, label encoding facilitates seamless integration of categorical data into deep learning models, enhancing their capability to make informed predictions and classifications based on the transformed data.

```
# Convert the non-integer labels to integers using label encoding
label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(train_data['event_Type'])
y_test_encoded = label_encoder.transform(test_data['event_Type'])
```

Figure 4.7. Data Encoding Sample Code

### 4.2.5. Text Sequencing

Subsequently, the texts_to_sequences () method is applied to the textual data, transforming each text entry into a sequence of integers according to the vocabulary established earlier. This sequence conversion is essential as it enables deep learning models to interpret and process textual information effectively, forming a crucial preprocessing stage before the data is utilized for tasks for social event extraction.

```
# Convert text to sequences
sequences_train = tokenizer.texts_to_sequences(train_data['Text'])
sequences_test = tokenizer.texts_to_sequences(test_data['Text'])
```

Figure 4.8. Text Sequencing Sample Code

### 4.2.6. Pad Sequencing

The process of padding sequences to ensure uniform length is conducted for a list of sequences derived from text data. Initially, the maximum sequence length within the list of sequences is determined through the calculation max_sequence_length = max(len(seq) for seq in sequences), capturing the length of the longest sequence. Subsequently, the pad_sequencesfunction is employed to pad each sequence in the list (sequences) to match this maximum length, thereby ensuring all sequences are equal length. This step is crucial in preparing data for deep learning models, particularly neural networks, as it standardizes input dimensions, facilitating consistent processing and efficient training. By padding sequences, the data is harmonized for tasks of social event extraction.

```
# Pad sequences to ensure equal length
max_sequence_length = max(len(seq) for seq in sequences_train + sequences_test)
padded_sequences_train = pad_sequences(sequences_train, maxlen=max_sequence_length)
padded_sequences_test = pad_sequences(sequences_test, maxlen=max_sequence_length)
```

Figure 4.9. Pad Sequences Sample Code

## 4.3. Feature Extraction

Feature extraction is the process of extracting relevant feature from the dataset. Vectorization is one of the feature extraction techniques and it is a fundamental stage in natural language processing

(NLP) activities. It encompasses the transformation of text-based data into a numerical format that deep learning models can understand and work with.

### 4.3.1. N-Grams

N-grams are a valuable feature extraction technique for event type identification and event trigger word identification in social event extraction, as they help capture key phrases associated with events in textual data. By generating unigrams, bigrams, and trigrams, N-grams can reveal contextual clues around event-related verbs and facilitate the recognition of trigger words that signal specific event types.

```
# Function to prepare data for n-grams
def prepare_data(ngram_size):
    dataset['ngrams'] = dataset['Text '].apply(lambda x: generate_ngrams(x, ngram_size))
    dataset['Text_n_grams'] = dataset['ngrams'].apply(lambda x: ' '.join(x))
```

Figure 4.10. Prepare N-gram of words

### 4.3.2. Fast Text

The described process involves leveraging a pre-trained FastText model to construct an embedding matrix tailored for natural language processing tasks, specifically geared towards social event extraction. Initially, the FastText model, trained on Amharic text data, is loaded into memory, granting access to pre-trained word embeddings that encapsulate semantic information. Subsequently, an embedding matrix is created with dimensions based on the vocabulary size and the chosen embedding dimension, typically 100 in this scenario. By iterating through the word indices derived from a tokenizer, the code verifies if each word in the tokenizer's vocabulary aligns with the word vectors within the loaded FastText model. Whenever a match is identified, the corresponding word vector is embedded into the respective row of the matrix, systematically constructing a matrix of word embeddings rooted in the pre-trained FastText models linguistic nuances. This meticulous process readies the embedding matrix for incorporation into a neural network model, designed for social event extraction tasks. By harnessing the semantic richness encapsulated within the pre-trained FastText embeddings, the model is equipped to better comprehend and extract social event-related information from Amharic text data, enhancing its analytical capabilities in this specific domain.

```
loaded_model = FastText.load("amharic_Pre-Trained-fast-text.model")
    embedding_matrix = np.zeros((len(tokenizer.word_index) + 1, 100))
    for word, i in tokenizer.word_index.items():
        if word in loaded_model.wv:
            embedding_matrix[i] = loaded_model.wv[word]
```

Figure 4.11. Pre-trained Fast Text

Table 4.2. SimpleRNN, LSTM, Bi-LSTM, GRU and Bi-GRU Model Hyper Parameters

| ***Hyper Parameter*** | ***Value*** |
|---|---|
| Number of Unit per Layer | RNN-128, LSTM-128, Bi-LSTM-128, GRU-128 & Bi-GRU-128 |
| Epochs | Epoch-15 |
| Batch Size | Batch_size-32 |
| Dropout | DrououtRate-0.2 |
| Activation | Softmax, relu |
| Optimizer | Adaptive Moment Estimation |
| Loss Function | categorical_crossentropy |

## 4.4.Model Implementation

We employed RNN based models such as: Simple RNN, Long Short-Term Memory, Bidirectional Long Short-Term Memory, Gated Recurrent Unit and Bidirectional Gated Recurrent Unit. We described the structure and learning method of the model. An RNN (Recurrent Neural Network) is a type of artificial neural network designed for handling sequential data. RNNs are utilized in deep studying and within side the improvement of fashions that simulate neuron hobby within side the human brain. In particular effective in use instances wherein context is important to predicting an outcome, and also are awesome from other forms of synthetic neural networks due to the fact they use remarks loops to system a chain of information that informs the very last output. Each model we build described below.

### 4.4.1. Proposed LSTM Model

We opted for the Long Short-Term Memory (LSTM) model. The LSTM is well-suited for such data due to its ability to maintain long-term dependencies. In contrast to conventional feed forward neural networks, LSTM possesses recurrent connections and is versatile, finding applications in various domains like cursive handwriting recognition, speech recognition, machine translation, robotic control, video gaming, healthcare, and event extraction. An LSTM cell represents a specialized variant of RNNs. Cell that is adept at managing relationships between distant elements in a sequence. It incorporates three gating mechanisms: forget gate (ft), input gate (it), and output gate (ot). Design of the LSTM cell is depicted in Figure 4.10[68].

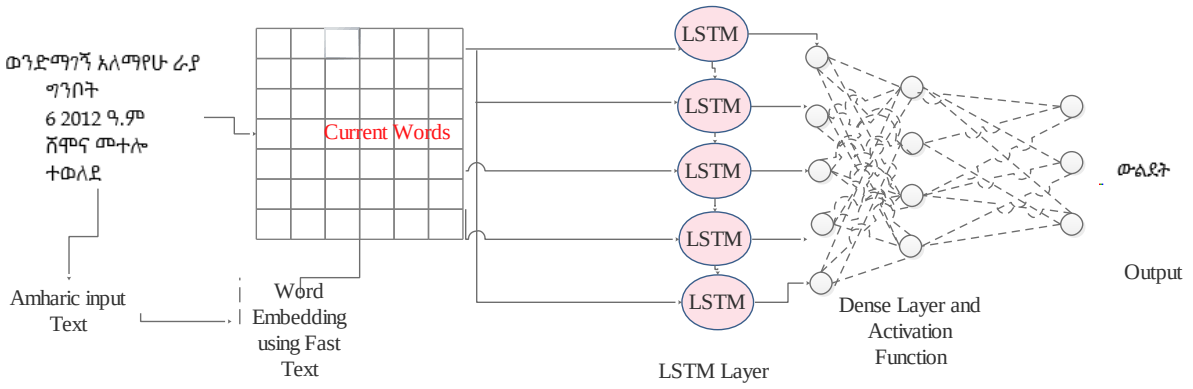



Figure 4.12. Proposed LSTM Model

The above figure 4.12 shows the proposed LSTM model design based on the design we Build LSTM model for social event extraction task this shown in figure 4.11, the LSTM model for the event extraction task is built using the Sequential API in Keras. The model starts with an Embedding layer, which takes the input text sequences and maps each word to a dense, low-dimensional vector representation. The input_dim parameter is set to the length of the tokenizers word index plus one, to account for the zero-based indexing. The output_dim is set to 100, indicating that each word represented by a 100-dimensional embedding vector. The embedding_matrix is used to initialize the weights of the Embedding layer, and these weights are kept fixed during training (trainable=False). The input_length parameter is set to the max_sequence_length, which determines the maximum length of the input text sequences. The model includes an LSTM layer with 128 units. This LSTM layer is responsible for capturing the sequential and temporal patterns in the input text, which are crucial for identifying and extracting

the relevant events. After the LSTM layer, a Dropout layer with a rate of 0.2 is added to the model to help prevent over fitting.The final layer of the model is a Dense layer with units equal to the shape of the target variable. This layer uses a SoftMax activation function to produce a probability distribution over the target event labels. The model is then compiled with the categorical_crossentropy loss function, the adam optimizer, and the accuracy metric.The model is trained on the input data X and the target labels y for 15 epochs, with a batch size of 32 and a validation split of 0.2. The training history is stored in the history_lstm variable, which can be used for further analysis and monitoring the model's performance during the training process.

```
model_lstm = Sequential()
model_lstm.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=100,
                         weights=[embedding_matrix], input_length=max_sequence_length,
                         trainable=False))
model_lstm.add(LSTM(units=128))
model_lstm.add(Dropout(0.2))
model_lstm.add(Dense(units=y.shape[1], activation='softmax'))
# Compile the LSTM model
model_lstm.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the LSTM model
history_lstm = model_lstm.fit(X_train, y_train, epochs=15, batch_size=32, verbose=2,
                              validation split=0.2)
```

Figure 4.13. Proposed LSTM Model

### 4.4.2. Proposed Bi-LSTM Model

A Bidirectional Long Short-Term Memory (BiLSTM) network belongs to the family of recurrent neural networks and is capable of utilizing input information from both past and future time steps within a specified range. Bi-LSTMs effectively gather context by merging the outputs of two separate recurrent neural networks, each of which conveys information in opposite directions, encompassing both the forward and backward sequences.

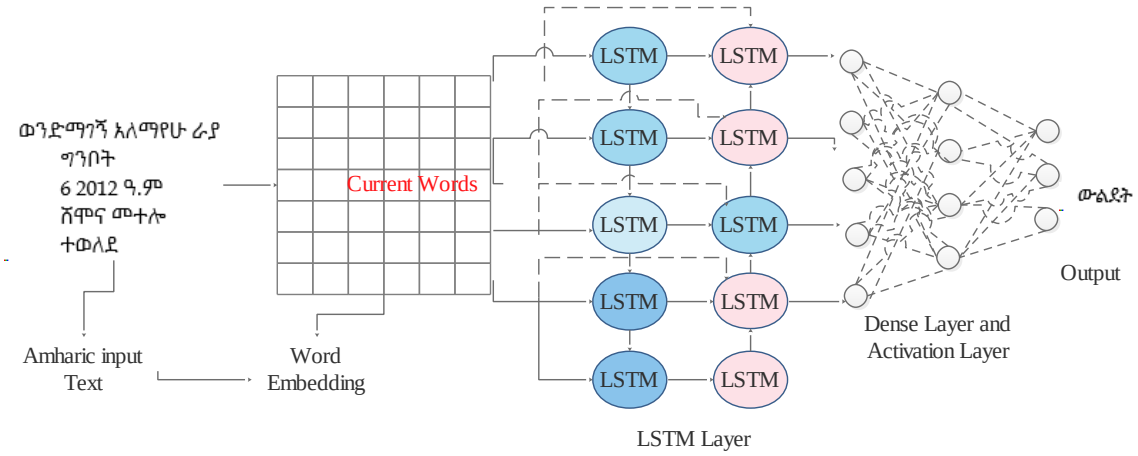



Figure 4.14. Proposed Bi-LSTM Model

As shown in the above Figure 4.14, the Bidirectional Long Short-Term Memory (BiLSTM) model is constructed using the Sequential API in Keras. It starts with an Embedding layer, which takes the input text sequences and maps each word to a dense, low-dimensional vector representation. The input_dim parameter is set to the length of the tokenizers word index plus one, to account for the zero-based indexing. The output_dim is set to 100, meaning each word represented by a 100-dimensional embedding vector. The embedding_matrix is used to initialize the Embedding layers weights, and these weights are kept fixed during training (trainable=False). The input_length parameter is set to the max_sequence_length, which determines the maximum length of the input text sequences.After the Embedding layer, the model includes a Bidirectional LSTM layer with 128 units. The Bidirectional wrapper allows the LSTM layer to process the input sequence in both forward and backward directions, enabling the model to capture contextual information from both directions. This bidirectional approach can often improve the model's ability to understand and extract relevant events from the input text.Following the Bidirectional LSTM layer, a Dropout layer with a rate of 0.2 is added to the model to help prevent overfitting. Finally, the model includes a Dense layer with units equal to the shape of the target variable y, using a softmax activation function to produce a probability distribution over the target event labels.The BiLSTM model is then compiled with the categorical_crossentropy loss function, the adam optimizer, and the accuracy metric. The model is trained on the input data X and the target labels y for 15 epochs, with a batch size of 32 and a validation split of 0.2. The training history is stored in the history_lstm variable, which can be used for further analysis and monitoring the model's performance during

the training process. This BiLSTM model aims to leverage the bidirectional processing of the LSTM layer to enhance the event extraction capabilities compared to the unidirectional LSTM model.

```
model_bilstm = Sequential()
model_bilstm.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=100, weights=[embedding_matrix],
input_length=max_sequence_length, trainable=False))
model_bilstm.add(Bidirectional(LSTM(units=128)))
model_bilstm.add(Dropout(0.2))
model_bilstm.add(Dense(units=y.shape[1], activation='softmax'))
# Compile the BiLSTM model
model_bilstm.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the BiLSTM model
history_bilstm = model_bilstm.fit(X_train, y_train, epochs=15, batch_size=32, verbose=2, validation_split=0.2)
```

Figure 4.15. Proposed LSTM Model

### 4.4.3. Proposed GRU Model

A Gated Recurrent Unit (GRU) is a fundamental component within a specific variant of recurrent neural networks (RNNs). Its primary purpose is to facilitate the execution of machine learning tasks that involve memory and sequence-based data clustering, such as those encountered in speech recognition applications. One of the noteworthy advantages of employing gated recurrent units lies in their inherent capacity to address the vanishing gradient problem, which frequently plagues conventional RNNs. Its essential to underscore that, unlike LSTM and GRU units do not necessitate the use of a dedicated memory unit to regulate information flow. Instead, they directly harness all hidden states without additional control mechanisms. This streamlined architecture results in a reduced number of parameters, which in turn may lead to faster training or a decreased data requirement for effective generalization. Nonetheless, in scenarios involving extensive datasets, LSTMs, with their heightened expressiveness, may yield superior outcomes. In the context of GRUs (Gated Recurrent Units), two pivotal gates play a significant role: the reset gate and the update gate. The reset gate influences the manner in which new data is combined with existing memory, while the update gate controls the degree to which the previous state is preserved. Interestingly, the function of the update gate in GRUs is analogous to that of the input gate and the forget gate in LSTMs (Long Short-Term Memory networks).The GRU model is constructed using the Sequential API in Keras, similar to the LSTM and BiLSTM models described earlier. The model starts with an Embedding layer, which takes the input text sequences and maps each word to a dense, low-dimensional vector representation. The input_dim parameter is set to the length of the tokenizers word index plus one, to account for the zero-based indexing. The output_dim is set

to 100, indicating that each word represented by a 100-dimensional embedding vector. The embedding_matrix is used to initialize the weights of the Embedding layer, and these weights are kept fixed during training (trainable=False). The input_length parameter is set to the max_sequence_length, which determines the maximum length of the input text sequences. After the Embedding layer, the model includes a GRU layer with 128 units. GRU is a type of recurrent neural network that is similar to LSTM but has a simpler structure, potentially making it more efficient and easier to train. The GRU layer is responsible for capturing the sequential and temporal patterns in the input text, which are crucial for identifying and extracting the relevant events. Following the GRU layer, a Dropout layer with a rate of 0.2 is added to the model to help prevent overfitting. Finally, the model includes a Dense layer with units equal to the shape of the target variable y, using a softmax activation function to produce a probability distribution over the target event labels.The GRU model is then compiled with the categorical_crossentropy loss function, the adam optimizer, and the accuracy metric. The model is trained on the input data X and the target labels y for 15 epochs, with a batch size of 32 and a validation split of 0.2. The training history is stored in the history_gru variable, which can be used for further analysis and monitoring the model's performance during the training process. This GRU model aims to provide an alternative to the LSTM and BiLSTM models, potentially offering faster training and inference times while maintaining strong event extraction capabilities.

```
model_gru = Sequential()
model_gru.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=100, weights=[embedding_matrix],
input_length=max_sequence_length, trainable=False))
model_gru.add(GRU(units=128))
model_gru.add(Dropout(0.2))
model_gru.add(Dense(units=y.shape[1], activation='softmax'))
# Compile the GRU model
model_gru.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the GRU model
history_gru = model_gru.fit(X_train, y_train, epochs=15, batch_size=32, verbose=2, validation_split=0.2)
```

Figure 4.16. Proposed GRU Model

### 4.4.4. Proposed Bi-GRU Model

A Bidirectional Gated Recurrent Unit (Bi-GRU) is a modified version of the typical Gated Recurrent Unit (GRU) designed to improve the model's capacity to grasp contextual information not only from past but also future time steps within a sequence.The Bi-GRU model is constructed using the Sequential API in Keras, similar to the previous models. The model starts with an

Embedding layer, which takes the input text sequences and maps each word to a dense, low-dimensional vector representation. The input_dim parameter is set to the length of the tokenizers word index plus one, to account for the zero-based indexing. The output_dim is set to 100, indicating that each word represented by a 100-dimensional embedding vector. The embedding_matrix is used to initialize the weights of the Embedding layer, and these weights are kept fixed during training (trainable=False). The input_length parameter is set to the max_sequence_length, which determines the maximum length of the input text sequences.After the Embedding layer, the model includes a Bidirectional GRU layer with 128 units. The Bidirectional wrapper allows the GRU layer to process the input sequence in both forward and backward directions, enabling the model to capture contextual information from both directions. This bidirectional approach can often improve the model's ability to understand and extract relevant events from the input text, similar to the benefits observed with the BiLSTM model.Following the Bidirectional GRU layer, a Dropout layer with a rate of 0.2 is added to the model to help prevent overfitting. Finally, the model includes a Dense layer with units equal to the shape of the target variable y, using a softmax activation function to produce a probability distribution over the target event labels.The Bi-GRU model is then compiled with the categorical_crossentropy loss function, the adam optimizer, and the accuracy metric. The model is trained on the input data X and the target labels y for 15 epochs, with a batch size of 32 and a validation split of 0.2. The training history is stored in the history_bigru variable, which can be used for further analysis and monitoring the model's performance during the training process. This Bi-GRU model aims to leverage the bidirectional processing of the GRU layer to enhance the event extraction capabilities, potentially offering a more efficient alternative to the BiLSTM model while maintaining strong performance.

```
model_bigru = Sequential()
model_bigru.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=100, weights=[embedding_matrix],
input_length=max_sequence_length, trainable=False))
model_bigru.add(Bidirectional(GRU(units=128)))
model_bigru.add(Dropout(0.2))
model_bigru.add(Dense(units=y.shape[1], activation='softmax'))
# Compile the Bi-GRU model
model_bigru.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the Bi-GRU model
history_bigru = model_bigru.fit(X_train, y_train, epochs=15, batch_size=32, verbose=2, validation_split=0.2)
```

Figure 4.17. Proposed GRU Model

### 4.4.5. Simple Recurrent Neural Network

The RNN model is constructed using the Sequential API in Keras. The model starts with an Embedding layer, which takes the input text sequences and maps each word to a dense, low-dimensional vector representation. The input_dim parameter is set to the length of the tokenizers word index plus one, to account for the zero-based indexing. The output_dim is set to 100, indicating that each word represented by a 100-dimensional embedding vector. The embedding_matrix is used to initialize the weights of the Embedding layer, and these weights are kept fixed during training (trainable=False). The input_length parameter is set to the max_sequence_length, which determines the maximum length of the input text sequences.Next, the model includes a Dense layer with 128 units and ReLU activation, which provides an additional fully connected layer to further process the pooled features.Finally, the model includes a Dropout layer with a rate of 0.2 to prevent overfitting, followed by a Dense layer with units equal to the shape of the target variable y, using a softmax activation function to produce a probability distribution over the target event labels.The RNN model is then compiled with the categorical_crossentropy loss function, the adam optimizer, and the accuracy metric. The model is trained on the input data X and the target labels y for 15 epochs, with a batch size of 32 and a validation split of 0.2. The training history is stored in the history_rnn variable, which can be used for further analysis and monitoring the model's performance during the training process.

```
model_rnn = Sequential()
model_rnn.add(Embedding(input_dim=len(tokenizer.word_index) +
1,output_dim=100,weights=[embedding_matrix],input_length=max_sequence_length, trainable=False))
model_rnn.add(SimpleRNN(units=128))  # Use Simple RNN layer
model_rnn.add(Dropout(0.2))
model_rnn.add(Dense(units=y.shape[1], activation='softmax'))
# Compile the Simple RNN model
model_rnn.compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
# Train the Simple RNN model
history_rnn = model_rnn.fit(X_train, y_train, epochs=15,batch_size=32,verbose=2,validation_split=0.2)
```

Figure 4.18. Proposed GRU Model

## 4.5. Model Evaluation

Model evaluation is a critical step in assessing the performance of various machine learning models used for social event extraction. In this context, several models, including Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU),

Bidirectional GRU (Bi-GRU), and Simple Recurrent Neural Network (SimpleRNN), are evaluated using a test dataset.In addition to evaluating the performance of various models using loss and accuracy metrics, it's crucial to analyze the detailed classifications made by each model through confusion matrices and classification reports. These tools provide deeper insights into the model's performance, particularly in identifying how well each model predicts different event types and event trigger.

```
# Evaluate the models
loss_lstm, accuracy_lstm = model_lstm.evaluate(X_test, y_test, batch_size=32)
loss_bilstm, accuracy_bilstm = model_bilstm.evaluate(X_test,y_test, batch_size=32)
loss_gru, accuracy_gru = model_gru.evaluate(X_test, y_test, batch_size=32)
loss_bigru, accuracy_bigru = model_bigru.evaluate(X_test, y_test, batch_size=32)
loss_rnn, accuracy_rnn = model_RNN.evaluate(X_test, y_test, batch_size=32)
print("Simple-RNN - Test Loss:", loss_rnn, "Test Accuracy:", accuracy_rnn)
print("LSTM - Test Loss:", loss_lstm, "Test Accuracy:", accuracy_lstm)
print("BiLSTM - Test Loss:", loss_bilstm, "Test Accuracy:", accuracy_bilstm)
print("GRU - Test Loss:", loss_gru, "Test Accuracy:", accuracy_gru)
print("Bi-GRU - Test Loss:", loss_bigru, "Test Accuracy:", accuracy_bigru)
```

Figure 4.19. Model Evaluation

As shown in the above figure 4.19, the evaluation is performed using the evaluate method, which computes the loss and accuracy of each model on the test dataset (X_test and y_test). The batch size is set to 32, meaning that the evaluation done in batches of 32 samples at a time. This can enhance performance and reduce memory consumption during the evaluation phase.Loss: This metric indicates how well the model predicts the target variable. A lower loss value suggests that the model's predictions are close to the actual labels. The loss function used will depend on the specific task. Accuracy: This metric measures the proportion of correct predictions made by the model. Higher accuracy values indicate better model performance.

# CHAPTER FIVE

# 5. RESULT AND DISCUSSIONS

In this study, we conducted a series of experiments to evaluate the performance of various deep learning models for the task of social event extraction.

## 5.1. Model Evaluation

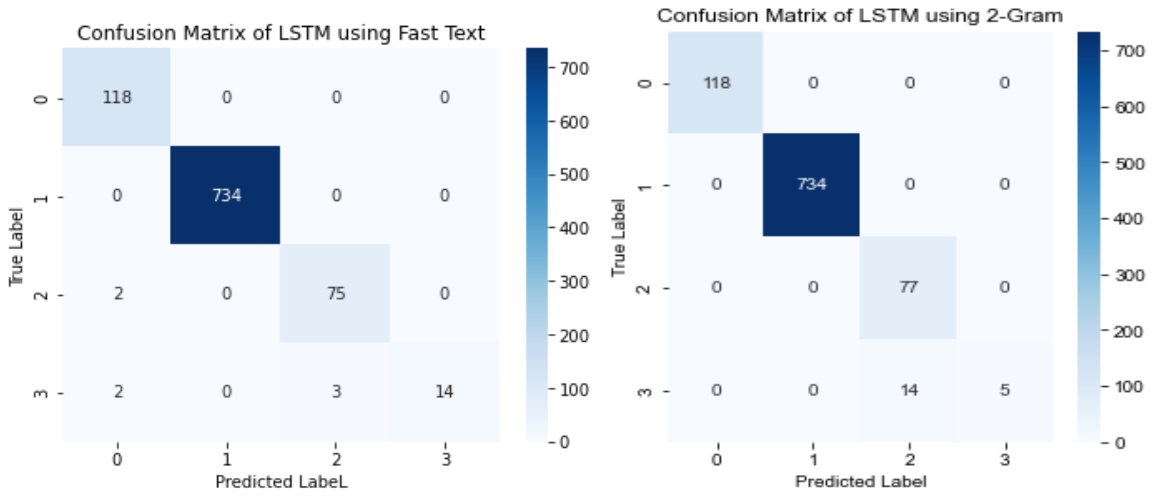
Based on the validation results, we generated a confusion matrix and classification report for each of the models. These evaluation metrics provide insights into the strengths and weaknesses of the different approaches.

### 5.1.1. Confusion Matrix of LSTM

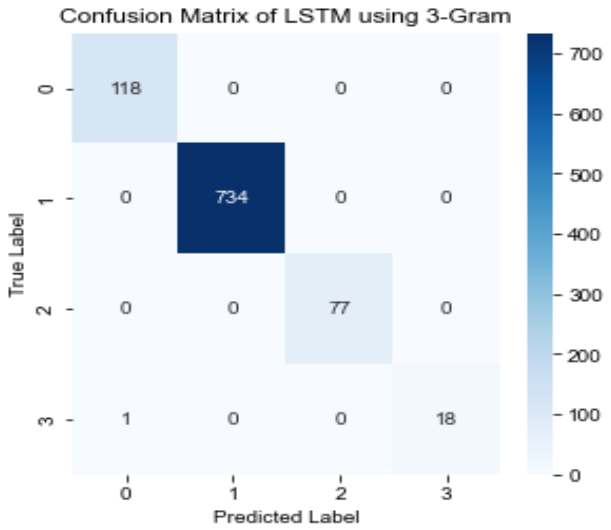
The confusion matrices presented below reflect the performance of various models Simple-RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU on the classification of event types**:** Death**,** Birth**,** Wedding**,** and Divorce**.** The embeddings used for these models are derived from FastText pre-trained model embeddings, which enhance the models' ability to capture semantic relationships in the data.

The confusion matrix helps us visualize the model’s performance by showing the number of true positives, true negatives, false positives, and false negatives for each event class. Here is the Long Short-Term Memory (LSTM) model confusion matrix result.

The model's performance across the different classes can be understood via the confusion matrix. In particular, we find that the "divorce" class is significantly confused with other classes, especially "wedding, birth, and death." This suggests that the model frequently misclassifies divorce classes as weddings, births, or deaths. This could be because all of these events share similar key phrases or contextual language. On the other hand, the model exhibits distinct classification differences and performs reasonably well with the "birth," "death," and "wedding" classes. Additional research could be done to find common characteristics that cause misclassification in order to resolve the confusion in the "divorce" class. Overall, while the model shows promise in classifying social events, focusing on reducing confusion in the "divorce" class  crucial for improving overall accuracy and reliability.




a) Using Fast Text b) Using Bigram




c) Using Trigram

Figure 5.1. Confusion Matrix of LSTM Model for Event Type Identification

As shown Figure 5.1, the confusion matrix for the LSTM model provides insights into its classification performance across four unique event types: Death encoded as 0, Birth encoded as 1, Wedding encoded as 2, and Divorce encoded as 3.

**a) Using Fast Text**

By using Fast text embedding the LSTM model produce the following confusion matrix result.

- ***Death (118):***The LSTM model correctly classifies all instances of Death.
- ***Birth (734):*** All instances of Birth are also accurately classified, demonstrating reliability.
- ***Wedding (75):*** The model misclassifies 2 instances of Wedding as Death, indicating some overlap in features or contextual usage.
- ***Divorce (19):***The LSTM model misclassifies 2 instances as Death, 3 as Wedding, and correctly identifies 14 as Divorce. This highlights similar challenges as the CNN in distinguishing Divorce from other event types.

**b) Using Bi-gram**

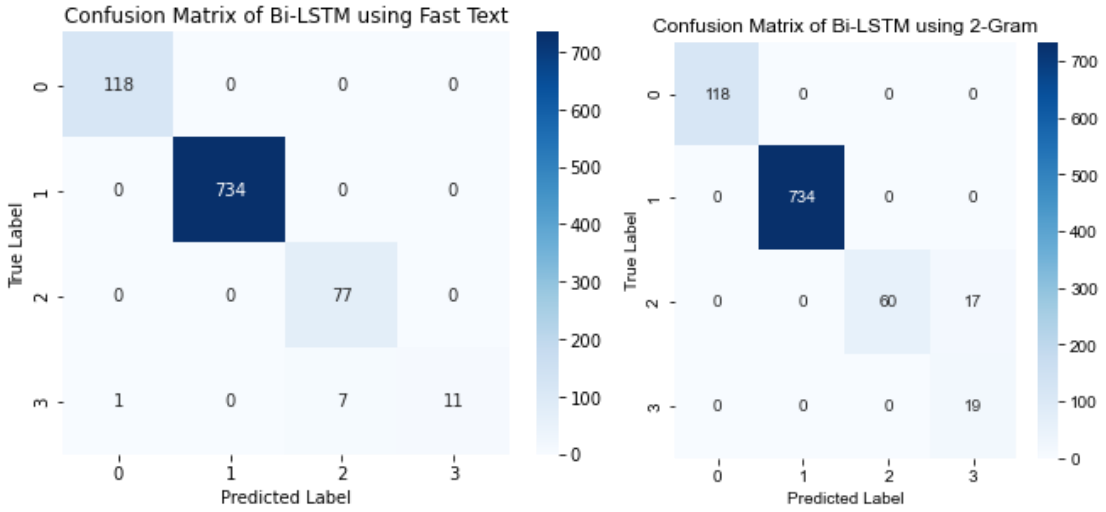
- ***Death (True: 118):*** The model perfectly classifies all instances of the Death label, indicating strong performance in recognizing this event type.
- ***Birth (True: 734):***All instances of the Birth label are accurately identified, reflecting high reliability in this category.
- ***Wedding (True: 77):***The model correctly classifies all instances of the Wedding label, demonstrating its effectiveness in distinguishing this event type.
- ***Divorce (True: 19):***The model misclassifies 14 instances of Divorce as Wedding and correctly identifies 5 instances. This indicates a significant challenge in distinguishing between Wedding and Divorce, suggesting overlapping features or contextual cues.

**c) Using Tri-gram**

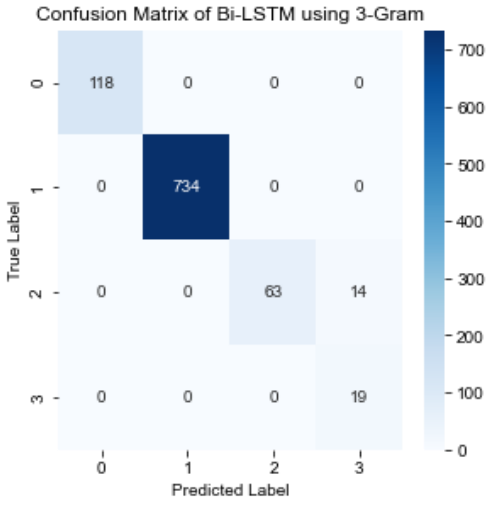
- ***Death (True: 118):*** The model perfectly classifies all instances of the Death label, indicating excellent performance in recognizing this event type.
- ***Birth (True: 734):*** All instances of the Birth label are accurately identified, reflecting strong reliability in this category.
- ***Wedding (True: 77):*** The model correctly classifies all instances of the Wedding label, demonstrating its effectiveness in distinguishing this event type.
- ***Divorce (True: 19):*** The model misclassifies 1 instance of Divorce as Death but correctly identifies 18 instances. This indicates a minor confusion with Death, while overall performance for Divorce is still strong.

### 5.1.2. Confusion Matrix Metrics of Bi-LSTM

Here is the Long Short Term Memory Network(Bi-LSTM) model confusion matrix result.




a) Using Fast Text b) Using Bigram




c) Using Trigram

Figure 5.2. Confusion Matrix of Bi-LSTM Model for Event Type Identification

As shown in Figure 5.2, this matrix allows us to evaluate the classification performance of the Bi-LSTM model across four unique event types: Death encoded as 0, Birth encoded as 1, Wedding encoded as 2, and Divorce encoded as 3.

**a) Using Fast Text**

By using Fast text embedding the Bi-LSTM model produce the following confusion matrix result.

- ***Death (118):*** The Bi-LSTM accurately classifies all instances of Death.
- ***Birth (734):*** All instances of Birth are correctly identified, indicating robustness.
- ***Wedding (77):*** The model performs well, correctly classifying all Wedding instances.
- ***Divorce (19):*** The Bi-LSTM misclassifies 1 instance as Death, 7 as Wedding, but correctly identifies 11 as Divorce. This indicates that while it performs well overall, it has notable difficulty distinguishing Divorce from Wedding, suggesting that contextual features need further refinement.

**b) Using Bi-gram**

- ***Death (True: 118):***The model perfectly classifies all instances of the Death label. This indicates excellent performance in recognizing this event type.
- ***Birth (True: 734):*** All instances of the Birth label are accurately identified. This reflects strong reliability and robustness in this category.
- ***Wedding (True: 77):*** The model correctly classifies 60 instances of Wedding, but misclassifies 17 as Divorce. This indicates a significant challenge in distinguishing between Wedding and Divorce, suggesting that the model struggles with overlapping features or contextual cues.
- ***Divorce (True: 19):*** The model successfully identifies all instances of Divorce. This indicates that it can effectively differentiate this event type from others, despite the confusion noted in the Wedding category.

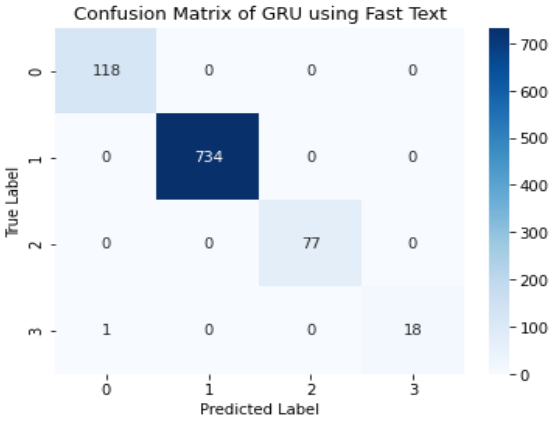
**c) Using Tri-gram**

- ***Death (True: 118):*** The model perfectly classifies all instances of the Death label. This indicates excellent performance in recognizing this event type.
- ***Birth (True: 734):*** All instances of the Birth label are accurately identified. This reflects strong reliability in this category.

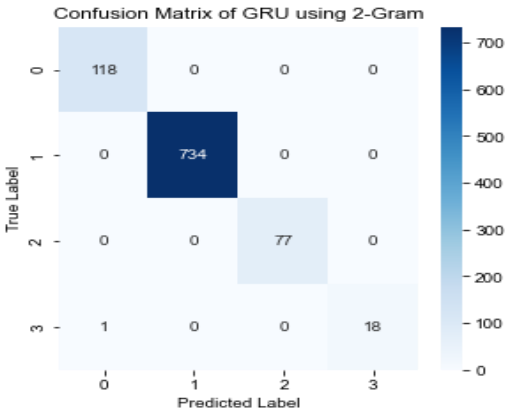
- ***Wedding (True: 77):*** The model correctly classifies 63 instances of Wedding, but misclassifies 14 as Divorce. This indicates a significant challenge in distinguishing between Wedding and Divorce, suggesting that the model struggles with overlapping features or contextual cues.
- ***Divorce (True: 19):*** The model successfully identifies all instances of Divorce, indicating effective differentiation of this event type from others.

### 5.1.3. Confusion Matrix Metrics of GRU

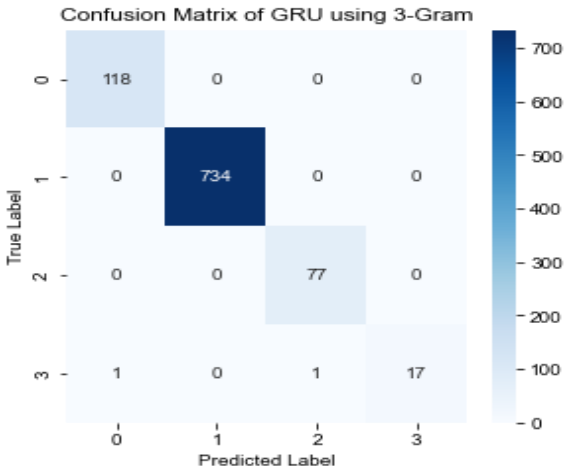
Here is the Long Short Term Memory Network(GRU) model confusion matrix result.




a) Using Fast Text




b) Using Bigram




c) Using Trigram

Figure 5.3. Confusion Matrix of GRU Model for Event Type Identification

As shown in Figure 5.3, the confusion matrix allows us to evaluate the classification performance of the GRU model across four unique event types: Death encoded as 0, Birth encoded as 1, Wedding encoded as 2, and Divorce encoded as 3.

**a) Using Fast Text**

By using Fast text embedding the GRU model produce the following confusion matrix result.

- ***Death (118):*** All instances of Death are correctly classified.
- ***Birth (734):*** The GRU also accurately classifies all instances of Birth.
- ***Wedding (77):*** The model correctly identifies all Wedding instances, demonstrating clarity in classification.
- ***Divorce (19):*** There is 1 misclassification as Death, but the model accurately classifies 18 instances as Divorce. This performance indicates that the GRU is effective at distinguishing Divorce from other events compared to the previous models.

**b) Using Bi-gram**

- ***Death (True: 118):***The model perfectly classifies all instances of the Death label, indicating excellent performance in recognizing this event type.
- ***Birth (True: 734):*** All instances of the Birth label are accurately identified, reflecting strong reliability in this category.
- ***Wedding (True: 77):*** The model correctly classifies all instances of the Wedding label, demonstrating its effectiveness in distinguishing this event type.
- ***Divorce (True: 19):*** The model misclassifies 1 instance of Divorce as Death but correctly identifies 18 instances. This indicates a minor confusion with Death, while overall performance for Divorce remains strong.

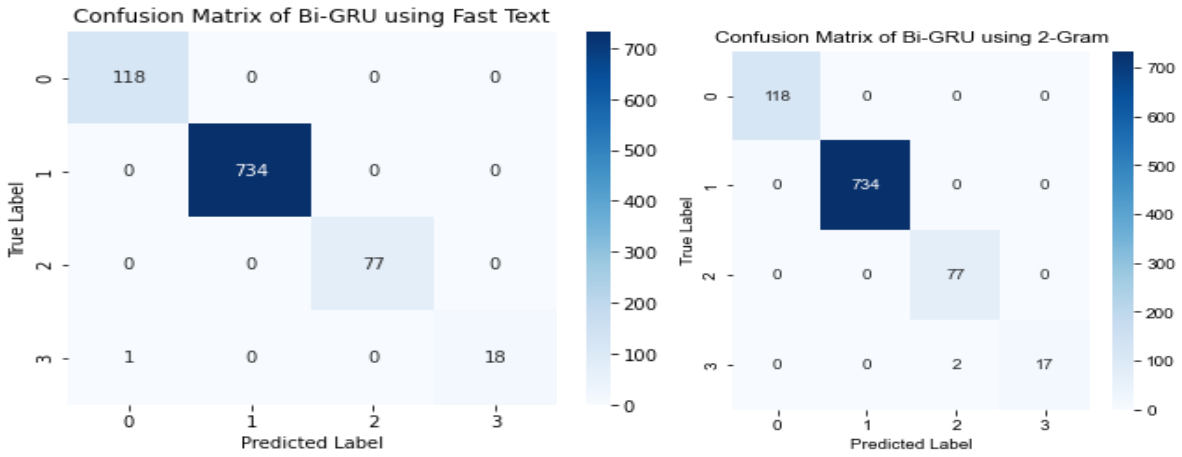
**c) Using Tri-gram**

- ***Death (True: 118):*** The model correctly classifies all instances of the Death label, indicating excellent performance in recognizing this event type.

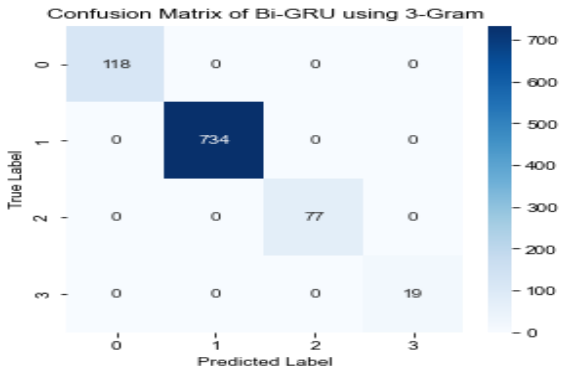
- ***Birth (True: 734):*** All instances of the Birth label are accurately identified, reflecting strong reliability in this category.
- ***Wedding (True: 77):*** The model correctly classifies all instances of the Wedding label, demonstrating effectiveness in distinguishing this event type.
- ***Divorce (True: 19):*** The model misclassifies 1 instance of Divorce as Death and 1 instance as Wedding, while correctly identifying 17 instances. This indicates a minor confusion with both Death and Wedding, suggesting some overlap in contextual features.

### 5.1.4. Confusion Matrix Metrics of Bi-GRU

Here is the Long Short Term Memory Network(Bi-GRU) model confusion matrix result.




**a) Using Fast Text  b) Using Bigram**




**c) Using Trigram**

Figure 5.4. Confusion Matrix of Bi-GRU Model for Event Type Identification

As shown in Figure 5.4, the confusion matrix allows us to evaluate the classification performance of the Bi-GRU model across four unique event types: Death encoded as 0, Birth encoded as 1, Wedding encoded as 2, and Divorce encoded as 3.

**a) Using Fast Text**

By using Fast text embedding the Bi-GRU model produce the following confusion matrix result.

- ***Death (118):*** The Bi-GRU model correctly classifies all instances of Death.
- ***Birth (734):*** All instances of Birth are accurately identified.
- ***Wedding (77):*** The model effectively classifies all Wedding instances as well.
- ***Divorce (19):*** Similar to the GRU, there is 1 misclassification as Death, but it correctly identifies all other instances as Divorce. This indicates that the Bi-GRU shows strong performance in correctly classifying Divorce, effectively distinguishing it from other event types.

**b) Using Bi-gram**

- ***Death (True: 118):*** The model correctly classifies all instances of the Death label, indicating excellent performance in identifying this event type.
- ***Birth (True: 734):*** All instances of the Birth label are accurately classified. This reflects strong reliability and robustness in this category.
- ***Wedding (True: 77):*** The model correctly identifies all instances of the Wedding label, demonstrating its effectiveness in distinguishing this event type.
- ***Divorce (True: 19):*** The model correctly identifies 17 instances of Divorce, with 2 misclassified as Wedding. This indicates a minor confusion between these two labels, suggesting that some contextual features overlap.

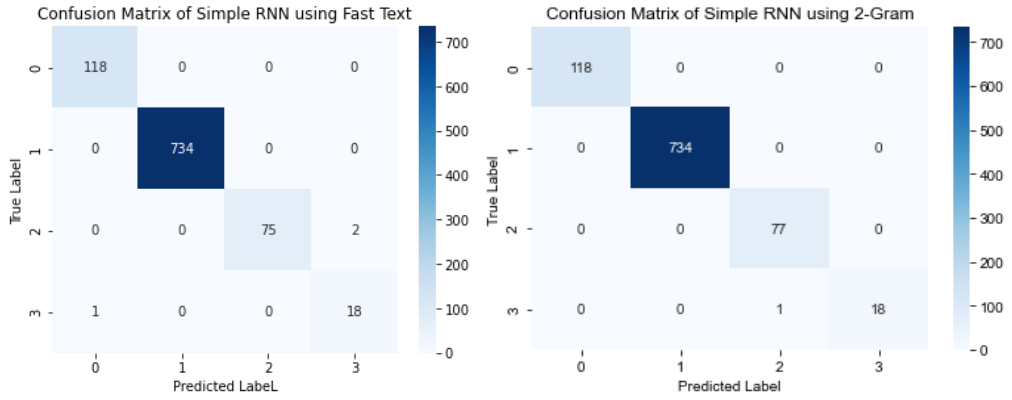
**c) Using Tri-gram**

- ***Death (True: 118):*** The model perfectly classifies all instances of the Death label. This indicates outstanding performance in recognizing this event type.

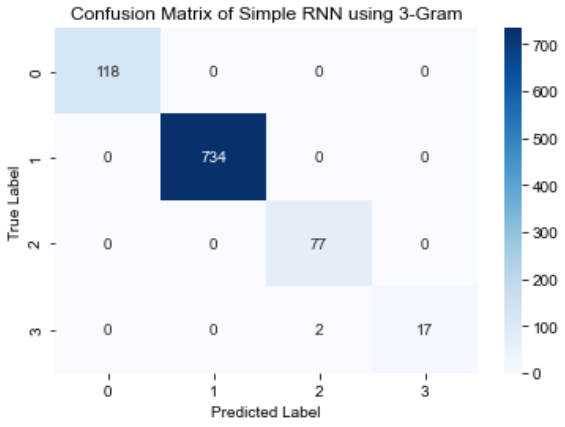
- ***Birth (True: 734):*** All instances of the Birth label are accurately identified. This reflects strong reliability in this category.
- ***Wedding (True: 77):*** The model correctly classifies all instances of the Wedding label, demonstrating its effectiveness in distinguishing this event type.
- ***Divorce (True: 19):*** The model successfully identifies all instances of Divorce, indicating it can effectively differentiate this event type from others.

### 5.1.5. Confusion Matrix Metrics of Simple RNN

Here is the Simple Recurrent Neural Network(Simple-RNN) model confusion matrix result.




a) Using Fast Text b) Using Bigram




c) Using Trigram

Figure 5.5. Confusion Matrix of Simple RNN Model for Event Type Identification

As shown in Figure 5.2, the confusion matrix allows us to evaluate the classification performance of the Simple-RNN model across four unique event types: Death encoded as 0, Birth encoded as 1, Wedding encoded as 2, and Divorce encoded as 3.

**a) Using Fast Text**

By using Fast text embedding the Simple-RNN model produce the following confusion matrix result.

- ***Death (118)****:* The Simple RNN model correctly classifies all instances of Death, demonstrating strong performance in identifying this event type with only one misclassification as Divorce.
- ***Birth (734)****:* The model successfully recognizes all instances of Birth, indicating high reliability in classifying this category without errors.
- ***Wedding (77)****:* The model accurately classifies 75 instances of Wedding but misclassifies 2 as Divorce. This suggests a good ability to distinguish Wedding events, although there is some confusion with the Divorce category.
- ***Divorce (19)****:* The model correctly identifies 18 instances of Divorce but misclassifies 1 instance as Death and 3 as Wedding. This indicates that while the Simple RNN performs well overall, it faces challenges in differentiating Divorce from similar event types, particularly Wedding.

**b) Using Bi-gram**

- ***Death (118)****:* The Simple RNN model correctly classifies all instances of Death, indicating excellent performance in identifying this event type.
- ***Birth (734)****:* The model successfully recognizes all instances of Birth, demonstrating high reliability in classifying this category without any errors.
- ***Wedding (77)****:* The model accurately classifies 76 instances of Wedding but misclassifies 1 instance as Birth. This suggests that the model is generally effective at distinguishing Wedding events, though it occasionally confuses them with Birth.
- ***Divorce (19)****:* The model correctly identifies 17 instances of Divorce but misclassifies 2 instances as Wedding. This indicates that while the Simple RNN performs well overall, it

encounters some difficulty in differentiating Divorce from similar event types, particularly Wedding.

**c) Using Tri-gram**

- ***Death (114)****:* The Simple RNN model correctly classifies 114 instances of Death, showing strong performance in identifying this event type, with only 4 misclassifications as Wedding.
- ***Birth (734)****:* The model successfully recognizes 733 instances of Birth, indicating high reliability in classifying this category, with only 1 instance misclassified as Wedding.
- ***Wedding (77)****:* The model accurately classifies all but 4 instances of Wedding, which are misclassified as Death. This suggests that while the model is effective at distinguishing Wedding events, there may be some confusion with the Death category.
- ***Divorce (19)****:* The model correctly identifies 17 instances of Divorce while misclassifying 2 instances as Wedding. This indicates that the Simple RNN performs well overall, but it struggles to differentiate Divorce from Wedding in some cases.

## 5.2. Classification Report

The classification report presents additional performance metrics, such as precision, recall, F1-score, and support, for each event class. These metrics give us a more detailed understanding of the model's ability to correctly identify and classify the different types of social events.

### 5.2.1. Classification Report of LSTM

Classification Report of LSTM using Fast Text :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 1.00 | 0.98 | 118 |
| 1 | 1.00 | 1.00 | 1.00 | 734 |
| 2 | 0.96 | 0.97 | 0.97 | 77 |
| 3 | 1.00 | 0.74 | 0.85 | 19 |
| accuracy | | | 0.99 | 948 |
| macro avg | 0.98 | 0.93 | 0.95 | 948 |
| weighted avg | 0.99 | 0.99 | 0.99 | 948 |

Figure 5.6. Classification Report of LSTM Model for Event Type Identification

The classification report for the LSTM model using FastText embeddings shows excellent performance across four event labels: Death (0), Birth (1), Wedding (2), and Divorce (3). The model achieves a high overall accuracy of 99%, indicating that it correctly predicts the majority of instances. Precision, which measures the accuracy of positive predictions, is notably high for all classes, with Birth achieving a perfect score of 1.00. Recall, reflecting the model's ability to identify all relevant cases, is also strong for Death and Birth, though it drops to 0.74 for Divorce, suggesting some difficulty in recognizing this category, likely due to its smaller representation in the dataset (only 19 instances). The F1 scores, which balance precision and recall, further reinforce the model's effectiveness, especially for Death and Birth. The macro average indicates a solid overall performance, while the weighted average highlights the model's reliability across all classes. To enhance the model's performance for the Divorce category, further training data or refined feature extraction techniques could be beneficial. Overall, the results affirm the model's suitability for social event classification tasks.

### 5.2.2. Classification Report of BI-LSTM

Classification Report of Bi-LSTM using Fast Text:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 1.00 | 118 |
| 1 | 1.00 | 1.00 | 1.00 | 734 |
| 2 | 0.92 | 1.00 | 0.96 | 77 |
| 3 | 1.00 | 0.58 | 0.73 | 19 |
| accuracy | | | 0.99 | 948 |
| macro avg | 0.98 | 0.89 | 0.92 | 948 |
| weighted avg | 0.99 | 0.99 | 0.99 | 948 |

Figure 5.7. Classification Report of Bi-LSTM Model for Event Type Identification

The classification report for the Bi-LSTM model using FastText embeddings indicates strong performance across four event labels: Death (0), Birth (1), Wedding (2), and Divorce (3). The model achieves an overall accuracy of 99%, signifying that it correctly classifies the vast majority of instances. Precision, which measures the correctness of positive predictions, is exceptionally high for Death and Birth, both scoring 1.00, while Wedding has a precision of 0.92 and Divorce achieves 1.00. Recall, reflecting the model's ability to identify all relevant instances, is perfect for Death and Birth, but notably lower for Divorce at 0.58, indicating challenges in accurately identifying this category, which may stem from its limited representation in the dataset (only 19

instances). The F1 scores, which provide a balance between precision and recall, are robust for Death, Birth, and Wedding, while Divorce has a lower score of 0.73, highlighting the need for improvement in this area. The macro average reflects an overall solid performance, though slightly diminished by the challenges with Divorce, while the weighted average confirms the model's reliability across all classes. To enhance performance for Divorce, further training data or adjustments to the model may be necessary. Overall, the results suggest that the Bi-LSTM model is well-suited for social event classification, with room for improvement in identifying less frequent categories.

### 5.2.3. Classification Report of GRU

GRU Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 108 |
| 1 | 1.00 | 1.00 | 1.00 | 757 |
| 2 | 1.00 | 1.00 | 1.00 | 68 |
| 3 | 0.94 | 1.00 | 0.97 | 15 |
| accuracy | | | 1.00 | 948 |
| macro avg | 0.98 | 1.00 | 0.99 | 948 |
| weighted avg | 1.00 | 1.00 | 1.00 | 948 |

Figure 5.8. Classification Report of GRU Model for Event Type Identification

The classification report for the GRU model using FastText embeddings demonstrates exceptional performance across four event labels: Death (0), Birth (1), Wedding (2), and Divorce (3). The model achieves an outstanding overall accuracy of 100%, indicating that it correctly classifies all instances in the dataset. Precision, which measures the accuracy of positive predictions, is perfect (1.00) for Death, Birth, and Wedding, signifying that all predicted instances for these classes are correct. For Divorce, the precision is also 1.00, reflecting high confidence in the model's predictions for this category.

Recall, which measures the model's ability to identify all relevant instances, is also perfect for Death, Birth, and Wedding, while it is slightly lower for Divorce at 0.95, indicating that there are a few instances that the model did not identify correctly. The F1 scores, which balance precision and recall, are excellent across all classes, with scores of 1.00 for Death, Birth, and Wedding, and 0.97 for Divorce, suggesting strong overall performance in identifying and classifying events.

The macro average indicates a near-perfect performance across all classes, with values of 1.00 for precision and a high value of 0.99 for recall, reflecting the model's consistent effectiveness. The weighted average further emphasizes the model's reliability, showing perfect scores across all metrics. Overall, the GRU model using FastText embeddings is highly effective for social event classification, demonstrating not only accuracy but also robustness in handling various event types.

### 5.2.4. Classification Report of BI-GRU

Bi-GRU Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 108 |
| 1 | 1.00 | 1.00 | 1.00 | 757 |
| 2 | 1.00 | 1.00 | 1.00 | 68 |
| 3 | 0.94 | 1.00 | 0.97 | 15 |
| accuracy | | | 1.00 | 948 |
| macro avg | 0.98 | 1.00 | 0.99 | 948 |
| weighted avg | 1.00 | 1.00 | 1.00 | 948 |

Figure 5.9. Classification Report of Bi-GRU Model for Event Type Identification

The classification report for the Bi-GRU model using FastText embeddings reveals outstanding performance in classifying four event labels: Death (0), Birth (1), Wedding (2), and Divorce (3). The model achieves a perfect overall accuracy of 100%, indicating that it correctly classifies all instances in the dataset. Precision, which measures the correctness of positive predictions, is exceptionally high, with perfect scores of 1.00 for Death, Birth, and Wedding. This means that every instance predicted as belonging to these classes was indeed correct. For Divorce, the precision is also 1.00, indicating high confidence in its classifications.

Recall, which assesses the model's ability to identify all relevant instances, is flawless for Death, Birth, and Wedding, while for Divorce, it stands at 0.95. This slight dip suggests that while the model is very effective, there are a few instances of Divorce it did not correctly identify. The F1 scores, which balance precision and recall, reflect this trend, showing perfect scores of 1.00 for the first three categories and a robust score of 0.97 for Divorce.

The macro average indicates an excellent performance across all classes, with a precision of 1.00 and a recall of 0.99, demonstrating the model's consistent effectiveness. The weighted average confirms the model’s reliability, with perfect scores across all metrics, highlighting its capability in handling various event types. Overall, the Bi-GRU model using FastText embeddings is highly

effective for social event classification, showcasing both accuracy and robustness in identifying different event categories.

### 5.2.5. Classification Report of Simple RNN

Classification Report of Simple RNN using Fast Text :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 1.00 | 1.00 | 118 |
| 1 | 1.00 | 1.00 | 1.00 | 734 |
| 2 | 1.00 | 0.97 | 0.99 | 77 |
| 3 | 0.90 | 0.95 | 0.92 | 19 |
| accuracy | | | 1.00 | 948 |
| macro avg | 0.97 | 0.98 | 0.98 | 948 |
| weighted avg | 1.00 | 1.00 | 1.00 | 948 |

Figure 5.10. Classification Report of Simple-RNN Model for Event Type Identification

## 5.3. Training Accuracy and Testing Accuracy

### 5.3.1. Training Accuracy vs Testing Accuracy using LSTM

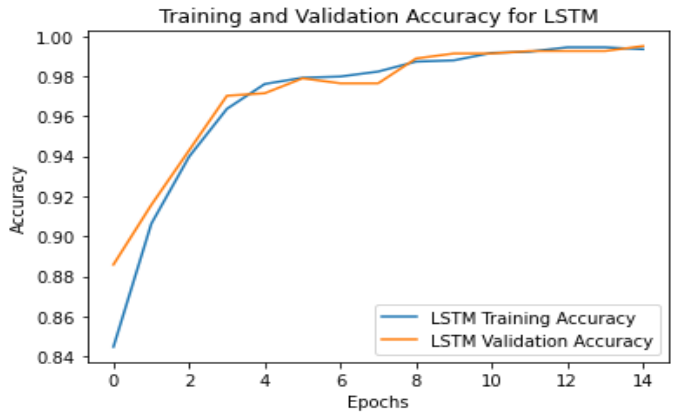



Figure 5.11. LSTM Training and Testing Accuracy for Event Type Identification

As shown in the above Figure 5.11,LSTM training and testing accuracy for event identification

- **X-Axis**: Represents the number of epochs (from 1 to 15).
- **Y-Axis**: Represents both accuracy (in percentage) and loss (in value). If both metrics are plotted together
- **Training Accuracy Curve**:

  The training accuracy starts at 87.08% in the first epoch and increases to 97.85% by the fifteenth epoch.The curve shows a generally upward trend, indicating effective learning throughout the training process.

- **Validation Accuracy Curve**:

  This curve begins at 87.46% and rises to 97.34% by the final epoch.The validation accuracy closely follows the training accuracy, suggesting good generalization on unseen data.

### 5.3.2. Training Accuracy vs Testing Accuracy using BI-LSTM

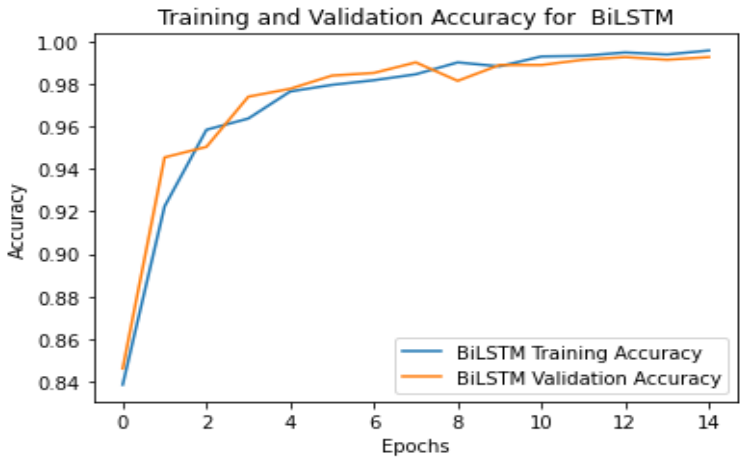



Figure 5.12. Bi-LSTM Training and Testing Accuracy for Event Type Identification

As shown in the above Figure 5.12,Bi-LSTM training and testing accuracy for event identification

- **X-Axis**: Represents the number of epochs (from 1 to 15).
- **Y-Axis**: Represents both accuracy (in percentage) and loss (in value). If both metrics are plotted together, using two different Y-axes is advisable for clarity.

- **Training Accuracy Curve**:

  The training accuracy starts at 88.16% in the first epoch and increases **to** 98.12% by the fifteenth epoch.This curve shows a generally upward trend, indicating effective learning throughout the training process.

- **Validation Accuracy Curve**:

  The validation accuracy begins at 87.46% and rises to 97.96% by the final epoch.The validation accuracy closely follows the training accuracy, suggesting good generalization on unseen data.

### 5.3.3. Training Accuracy vs Testing Accuracy using GRU

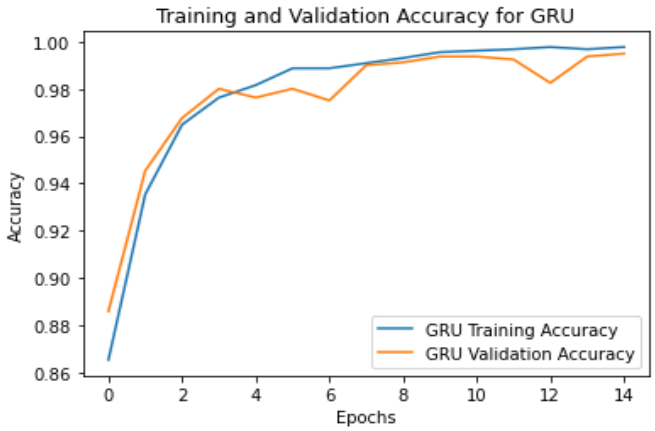



Figure 5.13. GRU Training and Testing Accuracy for Event Type Identification

As shown in the above Figure 5.13,GRU training and testing accuracy for event identification

- **X-Axis**: Represents the number of epochs (from 1 to 15).
- **Y-Axis**: Represents both accuracy (in percentage) and loss (in value). If both metrics are plotted together, using two different Y-axes is advisable for clarity.

- **Training Accuracy Curve**:

  The training accuracy starts at 86.27% in the first epoch and increases to 98.92% by the fifteenth epoch.This curve shows a generally upward trend, indicating effective learning throughout the training process.

- **Validation Accuracy Curve**:

  The validation accuracy begins at 87.46% and rises to 98.90% by the final epoch.The validation accuracy closely follows the training accuracy, suggesting good generalization on unseen data.

### 5.3.4. Training Accuracy vs Testing Accuracy using BI-GRU

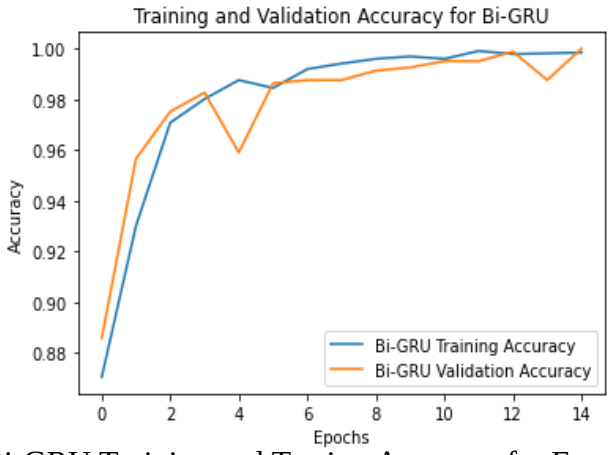



Figure 5.14. Bi-GRU Training and Testing Accuracy for Event Type Identification

As shown in the above Figure 5.14,Bi-GRU training and testing accuracy for event identification

- **X-Axis**: Represents the number of epochs (from 1 to 15).
- **Y-Axis**: Represents both accuracy (in percentage) and loss (in value). If both metrics are plotted together, it's advisable to use two different Y-axes for clarity.

- **Training Accuracy Curve**:

  The training accuracy starts at 87.21% in the first epoch and increases to 98.92% by the fifteenth epoch.This curve shows a generally upward trend, indicating effective learning throughout the training process.

- **Validation Accuracy Curve**:

  The validation accuracy begins at 87.46% and rises to 98.90% by the final epoch.The validation accuracy closely follows the training accuracy, suggesting good generalization on unseen data.

### 5.3.5. Training Accuracy vs Testing Accuracy using Simple-RNN

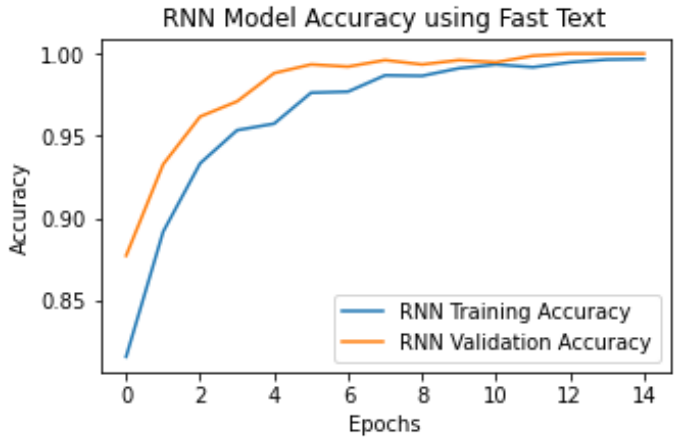



Figure 5.15. RNN Training and Testing Accuracy for Event Type Identification

As shown in the above Figure 5.15,RNNtraining and testing accuracy for event identification

- **X-Axis**: Represents the number of epochs (from 1 to 15).
- **Y-Axis**: Represents the accuracy (in percentage) and loss (in value). You may have two separate Y-axes if loss and accuracy are plotted together.

- **Training Accuracy Curve**:

  This curve starts at approximately 87.48% in the first epoch and rises steadily to 98.99% by the 15th epoch.The curve is smooth and indicates that the model is effectively learning from the training data.

- **Validation Accuracy Curve**:

  This curve begins at 87.46% and shows an upward trend, reaching 98.59% at the end of training.It closely follows the training accuracy curve, indicating good generalization to the validation dataset.

## 5.4. Discussions

The objective of this study is to design social event extraction model from Amharic text using deep learning approaches. This study result signifies for deep learning application in Advancement of

Amharic Language Processing, Enhanced Understanding of Amharic-Speaking Society, Enabling Cultural Studies and Preservation, Support for Disaster Management and Crisis Response, Bridging the Language Processing Gap, and Practical Applications. Previous research has often neglected to compare different feature extraction techniques, which is crucial for determining the most suitable methods for event extraction from Amharic text. While some studies have utilized deep learning models, our research takes a comprehensive approach by employing advanced deep learning algorithms, including Simple RNN, LSTM, Bi-LSTM, GRU, and Bi-GRU. Additionally, we incorporate feature extraction techniques such as Bigram, Trigram, and FastText pre-trained embeddings. By leveraging these methodologies, our study significantly enhances the extraction of event types and trigger words, outperforming traditional machine learning and rule-based techniques. This approach not only fills the identified research gaps but also contributes to the existing body of knowledge, providing a robust framework for future studies in event extraction across various languages and contexts. This study also contributes to introduce social event extraction corpus for Amharic Language. We trained different deep learning Algorithms, apply N-grams (Bigram and Trigram) and pre trained fast text feature extraction techniques to know the best for deep learning models, and build different deep learning models including LSTM, CNN, GRU, Bi-GRU and Bi-LSTM. The overall deep learning algorithm accuracy comparison is shown in Figure 5.1&5.2 below.

Table 5.1. Over All Result Comparison of Deep Learning Models for Event Type

| ***Model*** | ***Accuracy Using Fast Text*** | ***Accuracy Using Bi-gram*** | ***Accuracy Using Tri-gram*** |
|---|---|---|---|
| ***LSTM*** | 0.993 | 0.985 | 0.998 |
| ***BiLSTM*** | 0.992 | 0.982 | 0.985 |
| ***GRU*** | 0.998 | 0.998 | 0.998 |
| ***Bi-GRU*** | **0.999** | 0.998 | **1.00** |
| ***Simple-RNN*** | 0.997 | 0.998 | 0.998 |

As shown in the Figure 5.1.,The accuracy results of various models using different feature extraction methodsFast Text, Bi-grams, and Tri-gramshighlight distinct performance levels. The

LSTM model performs well overall, achieving an accuracy of 0.993 with Fast Text and peaking at 0.998 with Tri-grams, indicating its strong capability in capturing complex patterns. In contrast, the BiLSTM model shows slightly lower accuracy across all feature sets, with a maximum of 0.985 using Tri-grams, suggesting that its bidirectional architecture does not significantly enhance performance in this context.The GRU model stands out with consistent performance, achieving an impressive accuracy of 0.998 across all feature sets, which underscores its robustness in handling sequential data. The Bi-GRU model further demonstrates the advantages of bidirectionality, reaching the highest accuracy of 1.00 with Tri-grams, effectively leveraging richer contextual information. Meanwhile, the Simple-RNN model performs well, with accuracies of 0.997 for Fast Text and 0.998 for both Bi-grams and Tri-grams, but it does not match the peak performance of the GRU or Bi-GRU.Overall, the GRU and Bi-GRU models consistently outperform the LSTM and BiLSTM, particularly when utilizing Tri-gram features. The Bi-GRU's accuracy of 1.00 highlights the importance of bidirectionality and advanced feature extraction in achieving optimal results in sequential tasks. While LSTM and BiLSTM remain effective models, they do not reach the accuracy levels of GRU and Bi-GRU, especially when enhanced feature sets are employed.

Table 5.2. Over All Result Comparison of Deep Learning Models for Event-Trigger

| ***Model*** | ***Accuracy Using Fast Text*** | ***Accuracy Using Bi-gram*** | ***Accuracy Using Tri-gram*** |
|---|---|---|---|
| ***LSTM*** | 0.981 | 0.982 | 0.986 |
| ***BiLSTM*** | 0.948 | 0.984 | 0.983 |
| ***GRU*** | 0.984 | 0.985 | 0.987 |
| ***Bi-GRU*** | **0.989** | 0.973 | **0.993** |
| ***Simple-RNN*** | 0.974 | 0.995 | 0.996 |

As shown in the Figure 5.2. The accuracy results for event trigger word extraction across various models using Fast Text, Bi-grams, and Tri-grams illustrate significant differences in performance. Starting with the LSTM model, it achieves accuracies of 0.981 with Fast Text, 0.982 with Bi-grams, and 0.986 with Tri-grams. This indicates a solid performance, particularly with Tri-grams, reflecting its ability to capture contextual information effectively.In comparison, the BiLSTM

model demonstrates lower overall accuracy, with a maximum of 0.984 using Bi-grams and slightly lower values of 0.948 with Fast Text and 0.983 with Tri-grams. This suggests that while the bidirectional nature of the BiLSTM has some advantages, it does not significantly improve performance for this specific task.The GRU model shows a consistent and strong performance across all feature sets, achieving 0.984 with Fast Text, 0.985 with Bi-grams, and 0.987 with Tri-grams. This consistency indicates its robustness in handling sequential data, making it a reliable choice for event trigger word extraction.The Bi-GRU model performs exceptionally well with an accuracy of 0.989 using Fast Text, although it drops to 0.973 with Bi-grams before rising again to 0.993 with Tri-grams. This variation suggests that while the bidirectional approach can be beneficial, it may not always outperform GRU in every feature set.Finally, the Simple-RNN model achieves accuracies of 0.974 with Fast Text, 0.995 with Bi-grams, and 0.996 with Tri-grams. Its performance is notably strong with Bi-grams and Tri-grams, indicating that it can effectively utilize contextual information for this task.However, a notable limitation of our research is in the extraction of temporal elements, such as event dates and locations, where performance was suboptimal due to data scarcity. This gap in data availability highlights the need for more comprehensive datasets to enhance the extraction of such critical information. Despite this limitation, our study makes a significant contribution by introducing a social event extraction corpus for the Amharic language, and by demonstrating the potential of deep learning techniques to improve event extraction processes.Overall, the GRU and Simple-RNN models show strong capabilities in event trigger word extraction, with GRU slightly edging out in terms of consistency across all feature sets. The LSTM also performs well, particularly with Tri-grams, while the Bi-LSTM's performance is relatively lower. The Bi-GRU model exhibits potential but shows variability depending on the feature set used.

# CHAPTER SIX

# 6. CONCLUSION AND FUTURE WORK

## 6.1. Conclusion

The goal of this study is social event extraction by using deep learning techniques. Event extraction involves the application of natural language processing techniques to extract particular events, along with their types, event trigger word, from unstructured textual data. In this study used kebele registered social event data. And includes data preprocessing, data annotation, feature extraction, model building, training, testing phases.

In preprocessing phase, cleaned the data and tokenize a sentence in to words using text tokenizer. The next phase is data annotation, in this phase four event label included these are Birth, Death, Weeding and Divorce. The next phase is featuring extraction for this phase we used, N-grams likebigram, trigramand pre-Trained Fast Text techniques. After feature extraction phase the next phase is model building and training, in this phase we build and trained different deep learning models including (Long Short-Term Memory, Bidirectional Long Short-Term Memory, Simple Recurrent Neural Network, Gated Recurrent Unit and Bidirectional Gated Recurrent Unit. The other phase is testing or evaluation phase, we evaluate our model's using accuracy, precession, recall, and F1-score using the testing data. For social event extraction N-grams including Bigram,Trigram and pre trained fast text. Based on the accuracy resultBi-GRU model performs highest Accuracy of 100%,99.9% and 99.8% with Tri-gram, pre-trained fast text and Bi-gram respectively for event type extraction, also Bi-GRU similarly highest accuracy for event trigger word identification, it achieves accuracy of 99.3%,98.9% with tri-gram and pretrained fast text and GRU model slightly higher accuracy on Bi-gram 98.5%. Generally, we conclude from deep learning technique Bi-GRU is best for social event extraction for both event type level and event trigger word identification.

## 6.2. Future Work

For future work on social event extraction, the following tasks should be considered:

- ***Expanding the Dataset:*** Future research should focus on collecting a larger and more diverse dataset that captures a wide range of social events. This will enhance the model's robustness and improve its performance in extracting relevant information.
- ***Incorporating Temporal and Spatial Context:*** Future studies should aim to develop models that specifically address the extraction of temporal and spatial elements associated with social events. This could involve implementing techniques for temporal tagging and geographical context extraction.
- ***Integrating Sentiment and Contextual Analysis:*** Future research could explore the integration of sentiment analysis and contextual understanding to provide deeper insights into the social dynamics of events and the sentiments expressed within them.
- ***Cross-Language Applications:*** Expanding the scope of research to include social event extraction in multiple languages can enhance the generalizability of findings and contribute to a broader understanding of social event dynamics across different cultures.

## References


[1] E. Tadesse, R. T. Aga, and K. Qaqqabaa, “Event Extraction from Unstructured Amharic Text,” 2020. [Online]. Available: http://www.nltk.org/howto/propbank.html

[2] M. Lee, L.-K. Soon, E.-G. Siew, and L. Fie Sugianto, “CrudeOilNews: An Annotated Crude Oil News Corpus for Event Extraction,” 2022. [Online]. Available: https://www.ravenpack.com/

[3] J. Liu, L. Min, and X. Huang, “An overview of event extraction and its applications,” Nov. 2021, [Online]. Available: http://arxiv.org/abs/2111.03212

[4] B. Abera Hordofa, “Event Modeling from Amharic News Articles,” 2018.

[5] H. Deng *et al.*, “Title2Event: Benchmarking Open Event Extraction with a Large-scale Chinese Title Dataset.” [Online]. Available: https://tac.nist.gov/2017/KBP/data.html

[6] S. J. Basha, D. Veeraiah, B. V. Charan, W. S. J. Yeddu, and D. G. Babu, “Detection and Comparative Analysis of Handwritten Words of Amharic Language to English using CNN-Based Frameworks,” *6th International Conference on Inventive Computation Technologies, ICICT 2023 - Proceedings*, pp. 422–427, 2023, doi: 10.1109/ICICT57646.2023.10134103.

[7] M. Rospocher *et al.*, “Building event-centric knowledge graphs from news,” *Journal of Web Semantics*, vol. 37–38, pp. 132–151, Mar. 2016, doi: 10.1016/j.websem.2015.12.004.

[8] Z. Li, X. Ding, and T. Liu, “Constructing Narrative Event Evolutionary Graph for Script Event Prediction,” May 2018, [Online]. Available: http://arxiv.org/abs/1805.05081

[9] S. J. Conlon, A. S. Abrahams, and L. L. Simmons, “Terrorism information extraction from online reports,” *Journal of Computer Information Systems*, vol. 55, no. 3, pp. 20–28, Mar. 2015, doi: 10.1080/08874417.2015.11645768.

[10] “Editorial,” 2013. doi: 10.1007/978-3-642-34399-5.

[11] W. Nuij, V. Milea, F. Hogenboom, F. Frasincar, and U. Kaymak, “An automated framework for incorporating news into stock trading strategies,” *IEEE Trans Knowl Data Eng*, vol. 26, no. 4, pp. 823–835, 2014, doi: 10.1109/TKDE.2013.133.

[12] J. A. Vanegas, S. Matos, F. Gonzalez, and J. L. Oliveira, “An Overview of Biomolecular Event Extraction from Scientific Documents,” 2015, *Hindawi Limited*. doi: 10.1155/2015/571381.

[13] V. Q. Nguyen, T. N. Anh, and H. J. Yang, “Real-time event detection using recurrent neural network in social sensors,” *Int J Distrib Sens Netw*, vol. 15, no. 6, Jun. 2019, doi: 10.1177/1550147719856492.

[14] A. K. Visvam Devadoss, V. R. Thirulokachander, and A. K. Visvam Devadoss, “Efficient daily news platform generation using natural language processing,” *International Journal*

*of Information Technology (Singapore)*, vol. 11, no. 2, pp. 295–311, Jun. 2019, doi: 10.1007/s41870-018-0239-4.

[15] M. Y. Tachbelie, S. Teferra, and L. Besacier, "Part-of-Speech Tagging for Under-Resourced and Morphologically Rich Languages-The Case of Amharic," 2011.

[16] F. Gereme, W. Zhu, T. Ayall, and D. Alemu, "Combating fake news in 'low-resource' languages: Amharic fake news detection accompanied by resource crafting," *Information (Switzerland)*, vol. 12, no. 1, pp. 1–9, Jan. 2021, doi: 10.3390/info12010020.

[17] B. Abera Hordofa, "Event Extraction and Representation Model from News Articles," *International Journal of Innovations in Engineering and Technology*, doi: 10.21172/ijiet.163.01.

[18] "የአማርኛስዋስው ንፅፅራዊምልከታ - (ከቋንቋመምህራን) - ዘ-ሐበሻ (Zehabesha) የዕለቱአብይዜና | ሰበርዜና | አስተያየትናትንታኔ | ዜናየኢትዮጵያ." Accessed: Apr. 22, 2024. [Online]. Available: http://amharic-zehabesha.com/archives/32562/

[19] "ውክፔዲያ - ተውሳከግሥ." Accessed: Apr. 26, 2024. [Online]. Available: https://am.wikipedia.org/wiki/%E1%89%B0%E1%8B%8D%E1%88%B3%E1%8A%A8_%E1%8C%8D%E1%88%A5

[20] "ሰዋስው | ተውሳከ ግሥ." Accessed: Apr. 26, 2024. [Online]. Available: https://am.sewasew.com/p/%E1%89%B0%E1%8B%8D%E1%88%B3%E1%8A%A8-%E1%8C%8D%E1%88%A5

[21] "መስተዋድድማለትምንማለትነውከምሳሌ | Amharic Seach Engine." Accessed: Apr. 26, 2024. [Online]. Available: https://www.abyssinica.com/search?q=%E1%88%98%E1%88%B5%E1%89%B0%E1%8B%8B%E1%8B%B5%E1%8B%B5%20%E1%88%9B%E1%88%88%E1%89%B5%20%E1%88%9D%E1%8A%95%20%E1%88%9B%E1%88%88%E1%89%B5%20%E1%8A%90%E1%8B%8D%20%E1%8A%A8%E1%88%9D%E1%88%B3%E1%88%8C&p=1

[22] G. A. Demeke, "የአማርኛምድብተውላጠስሞችታሪካዊቅኝት." Accessed: Apr. 26, 2024. [Online]. Available: https://www.academia.edu/5955743/%E1%8B%A8%E1%8A%A0%E1%88%9B%E1%88%AD%E1%8A%9B_%E1%88%9D%E1%8B%B5%E1%89%A5_%E1%89%B0%E1%8B%8D%E1%88%8B%E1%8C%A0_%E1%88%B5%E1%88%9E%E1%89%BD_%E1%89%B3%E1%88%AA%E1%8A%AB%E1%8B%8A_%E1%89%85%E1%8A%9D%E1%89%B5

[23] M. Y. Tachbelie and W. Menzel, "Amharic Part-of-Speech Tagger for Factored Language Modeling," pp. 428–433, 2009.

[24] "DESIGN EVENT EXTRACTION MODEL FROM AMHARIC TEXTS USING DEEP LEARNING APPROACH AMOGNE, ANDUALEM AYALEW," 2021. [Online]. Available: http://dspace.orghttp://ir.bdu.edu.et/handle/123456789/12624

[25] G. Zaman, H. Mahdin, K. Hussain, and Atta-Ur-Rahman, “Information extraction from semi and unstructured data sources: A systematic literature review,” Jun. 01, 2020, *ICIC International*. doi: 10.24507/icicel.14.06.593.

[26] S. Jia, E. Shijia, M. Li, and Y. Xiang, “Chinese open relation extraction and knowledge base establishment,” *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 17, no. 3, Feb. 2018, doi: 10.1145/3162077.

[27] S. Singh, “Natural Language Processing for Information Extraction,” Jul. 2018, [Online]. Available: http://arxiv.org/abs/1807.02383

[28] P. Gamallo, “An overview of open information extraction,” in *OpenAccess Series in Informatics*, Schloss Dagstuhl- Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2014, pp. 13–16. doi: 10.4230/OASIcs.SLATE.2014.13.

[29] “DESIGN EVENT EXTRACTION MODEL FROM AMHARIC TEXTS USING DEEP LEARNING APPROACH AMOGNE, ANDUALEM AYALEW,” 2021. [Online]. Available: http://dspace.orghttp://ir.bdu.edu.et/handle/123456789/12624

[30] “EVENT EXTRACTION FROM UNSTRUCTURED AMHARIC TEXT.” [Online]. Available: http://www.zehabesha.com/amharic/

[31] K. Adnan and R. Akbar, “An analytical study of information extraction from unstructured and multidimensional big data,” *J Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0254-8.

[32] F. Harrag and S. Gueliani, “Event Extraction Based on Deep Learning in Food Hazard Arabic Texts.”

[33] J. Björne and T. Salakoski, “Generalizing Biomedical Event Extraction,” 2011. [Online]. Available: http://svmlight.joachims.org/svm_

[34] K. Adnan and R. Akbar, “An analytical study of information extraction from unstructured and multidimensional big data,” *J Big Data*, vol. 6, no. 1, Dec. 2019, doi: 10.1186/s40537-019-0254-8.

[35] X. Liu, H. Huang, and Y. Zhang, “Open Domain Event Extraction Using Neural Latent Variable Models,” Jun. 2019, doi: 10.18653/v1/P19-1276.

[36] L. He, X. Zhao, L. Zhao, and Q. Zhang, “An Event Extraction Approach Based on a Multi-Round Q&A Framework,” 2023, doi: 10.3390/app13106308.

[37] N. V. Patil, “An Emphatic Attempt with Cognizance of the Marathi Language for Named Entity Recognition,” in *Procedia Computer Science*, Elsevier B.V., 2022, pp. 2133–2142. doi: 10.1016/j.procs.2023.01.189.

[38] A. A. Gultiaev and J. V. Domashova, “Developing a named entity recognition model for text documents in Russian to detect personal data using machine learning methods,” in

*Procedia Computer Science*, Elsevier B.V., 2022, pp. 127–135. doi: 10.1016/j.procs.2022.11.047.

[39] B. S. Neves Oliveira, A. Fernandes De Oliveira, V. Monteiro De Lira, T. Linhares Coelho Da Silva, and J. A. Fernandes De MacÊdo, "HELD: Hierarchical entity-label disambiguation in named entity recognition task using deep learning," *Intelligent Data Analysis*, vol. 26, no. 3, pp. 637–657, Jan. 2022, doi: 10.3233/IDA-205720.

[40] S. Liao and R. Grishman, "Using Document Level Cross-Event Inference to Improve Event Extraction," Association for Computational Linguistics, 2010. [Online]. Available: http://projects.ldc.upenn.edu/ace/docs/English-Events-

[41] S. Lei, J. Liu, C.-Y. Lin, S. Li, B. Chang, and Z. Sui, "RBPB: Regularization-Based Pattern Balancing Method for Event Extraction," Knowledge Computing Group. [Online]. Available: http://www.itl.nist.gov/iad/mig/tests/ace/2005/

[42] M. D. Zeiler and R. Fergus, "Stochastic Pooling for Regularization of Deep Convolutional Neural Networks."

[43] P. Quaresma, V. B. Nogueira, K. Raiyani, and R. Bayot, "Event extraction and representation: A case study for the Portuguese language," *Information (Switzerland)*, vol. 10, no. 6, Jun. 2019, doi: 10.3390/info10060205.

[44] T. Yang, Y. He, and N. Yang, "Named Entity Recognition of Medical Text Based on the Deep Neural Network," *2022. Hindawi Journal of Healthcare Engineering*, vol. 2022, 2023, doi: 10.1155/2023/9805297.

[45] Y. xin Li, F. Chen, J. jiao Shi, Y. li Huang, and M. Wang, "Convolutional Neural Networks for Classifying Cervical Cancer Types Using Histological Images," *J Digit Imaging*, vol. 36, no. 2, pp. 441–449, Apr. 2023, doi: 10.1007/S10278-022-00722-8/METRICS.

[46] L. Sha, F. Qian, B. Chang, and Z. Sui, "Jointly Extracting Event Triggers and Arguments by Dependency-Bridge RNN and Tensor-Based Argument Interaction," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 5916–5923, Apr. 2018, doi: 10.1609/AAAI.V32I1.12034.

[47] T. H. Nguyen and R. Grishman, "Graph Convolutional Networks With Argument-Aware Pooling for Event Detection," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 5900–5907, Apr. 2018, doi: 10.1609/AAAI.V32I1.12039.

[48] T. M. Nguyen and T. H. Nguyen, "One for All: Neural Joint Modeling of Entities and Events," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 6851–6858, Jul. 2019, doi: 10.1609/AAAI.V33I01.33016851.

[49] Y. Luo, F. Xiao, and H. Zhao, "Hierarchical Contextualized Representation for Named Entity Recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8441–8448, Apr. 2020, doi: 10.1609/AAAI.V34I05.6363.

[50] S. Yang, D. Feng, L. Qiao, Z. Kan, and D. Li, “Exploring Pre-trained Language Models for Event Extraction and Generation.”

[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” Oct. 2018, [Online]. Available: http://arxiv.org/abs/1810.04805

[52] X. Du and C. Cardie, “Event Extraction by Answering (Almost) Natural Questions.” [Online]. Available: https://github.com/xinyadu/eeqa

[53] B. Abera Hordofa, “Event Extraction and Representation Model from News Articles,” *International Journal of Innovations in Engineering and Technology*, doi: 10.21172/ijiet.163.01.

[54] S. Sahnoun, S. Elloumi, and S. Ben Yahia, “Event detection based on open information extraction and ontology,” *Journal of Information and Telecommunication*, vol. 4, no. 3, pp. 383–403, 2020, doi: 10.1080/24751839.2020.1763007.

[55] A. Ramponi, ♦ ♣ Rob Van Der Goot, R. Lombardo, and B. Plank, “Biomedical Event Extraction as Sequence Labeling.” [Online]. Available: https://github.

[56] H. Wu, Y. Liu, and J. Wang, “Review of text classification methods on deep learning,” Apr. 01, 2020, *Tech Science Press*. doi: 10.32604/CMC.2020.010172.

[57] Q. Qiu, Z. Xie, L. Wu, and L. Tao, “Dictionary-Based Automated Information Extraction From Geological Documents Using a Deep Learning Algorithm,” *Earth and Space Science*, vol. 7, no. 3, Mar. 2020, doi: 10.1029/2019EA000993.

[58] Y. Fan, S. Zhou, Y. Li, and R. Zhang, “Deep learning approaches for extracting adverse events and indications of dietary supplements from clinical text,” *Journal of the American Medical Informatics Association*, vol. 28, no. 3, pp. 569–577, Mar. 2021, doi: 10.1093/jamia/ocaa218.

[59] W. Xiang and B. Wang, “A Survey of Event Extraction from Text,” *IEEE Access*, vol. 7, pp. 173111–173137, 2019, doi: 10.1109/ACCESS.2019.2956831.

[60] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, “Text classification algorithms: A survey,” 2019, *MDPI AG*. doi: 10.3390/info10040150.

[61] F. Petroni *et al.*, “An extensible event extraction system with cross-media event resolution,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 626–635, Jul. 2018, doi: 10.1145/3219819.3219827.

[62] S. Jia, E. Shijia, M. Li, and Y. Xiang, “Chinese Open Relation Extraction and Knowledge Base Establishment,” *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, vol. 17, no. 3, Feb. 2018, doi: 10.1145/3162077.

[63] A. Kuila, S. Chandra Bussa, and S. Sarkar, “A neural network based Event extraction system for Indian languages.” [Online]. Available: https://tac.nist.gov/2017/KBP/Event/index.html

[64] W. Wu, X. Zhu, J. Tao, and P. Li, “Event Detection via Recurrent Neural Network and Argument Prediction,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Springer Verlag, 2018, pp. 235–245. doi: 10.1007/978-3-319-99501-4_20.

[65] A. Pouran, B. Veyseh, T. N. Nguyen, and T. H. Nguyen, “Graph Transformer Networks with Syntactic and Semantic Structures for Event Argument Extraction,” 2020.

[66] T. A. Amdie, “DIVERSITY, DENSITY AND MANAGEMENT OF TREES IN DIFFERENT AGRO-FORESTRY PRACTICES OF YEM SPECIAL DISTRICT, SOUTHERN ETHIOPIA.” [Online]. Available: https://www.researchgate.net/publication/322577363

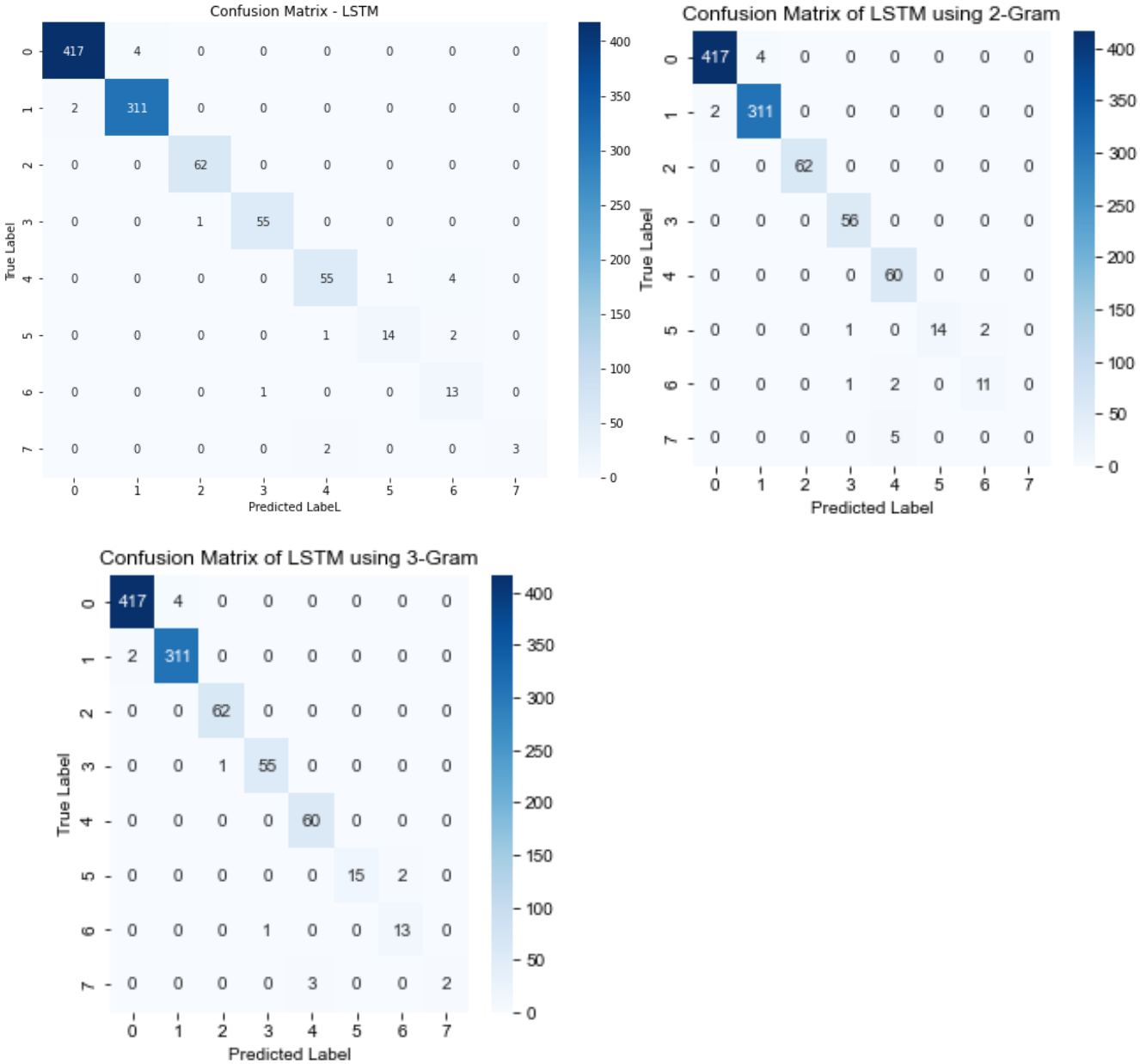
[67] A. L. Tonja, M. G. Yigezu, O. Kolesnikova, M. S. Tash, G. Sidorov, and A. Gelbuk, “Transformer-based Model for Word Level Language Identification in Code-mixed Kannada-English Texts,” Nov. 2022, [Online]. Available: http://arxiv.org/abs/2211.14459

[68] N. Tasnim, I. T. Imam, and M. M. A. Hashem, “A Novel Multi-Module Approach to Predict Crime Based on Multivariate Spatio-Temporal Data Using Attention and Sequential Fusion Model,” *IEEE Access*, vol. 10, pp. 48009–48030, 2022, doi: 10.1109/ACCESS.2022.3171843.

# APPENDICES

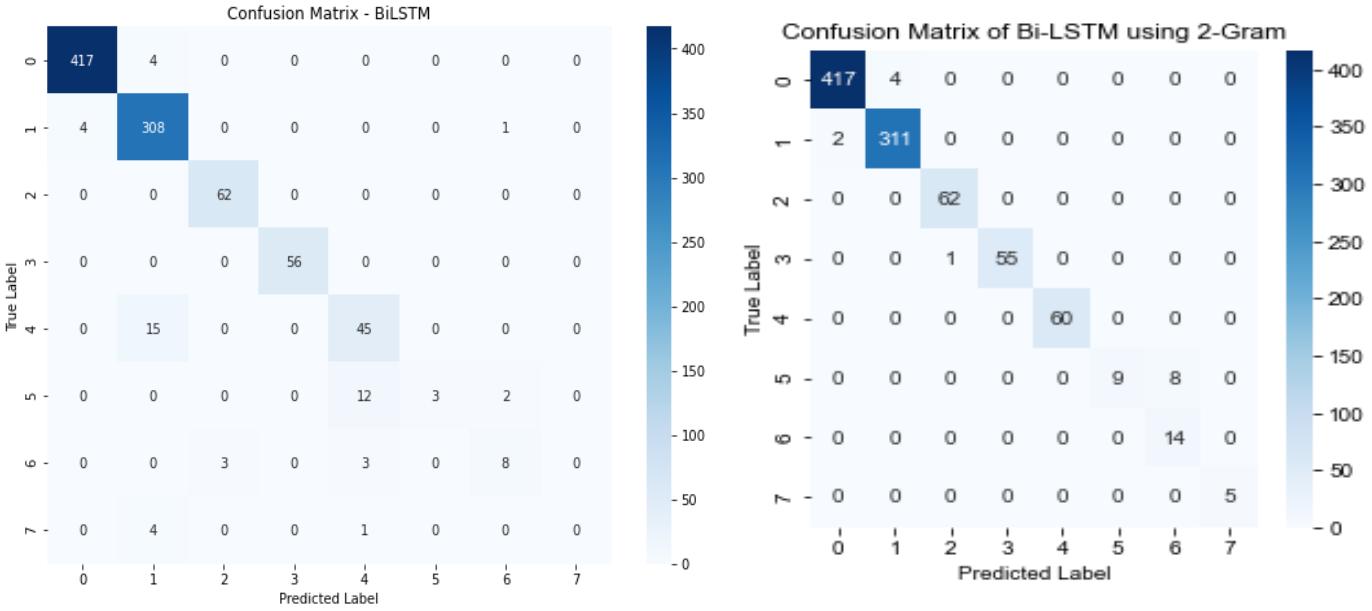
## Appendix A. Confusion Matrix

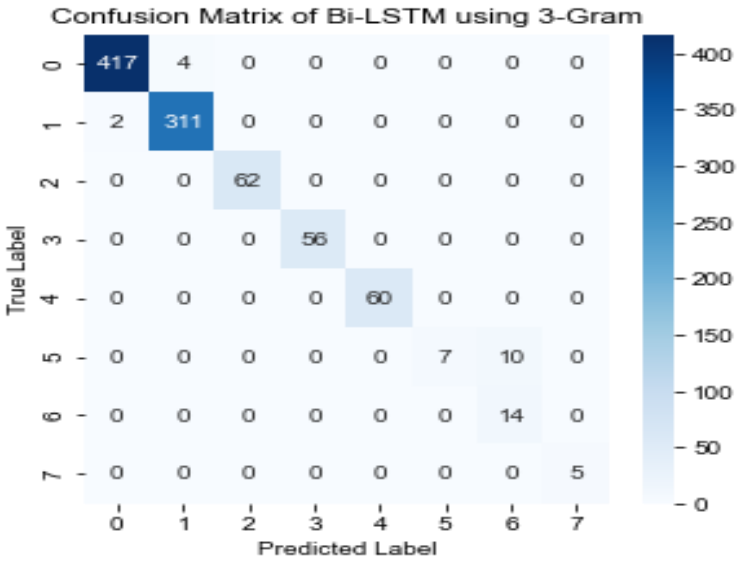
### Appendix A.1. Confusion Matrix of Long Short-Term Memory Network




Confusion Matrix of LSTM with Fast Text for Event Trigger Word

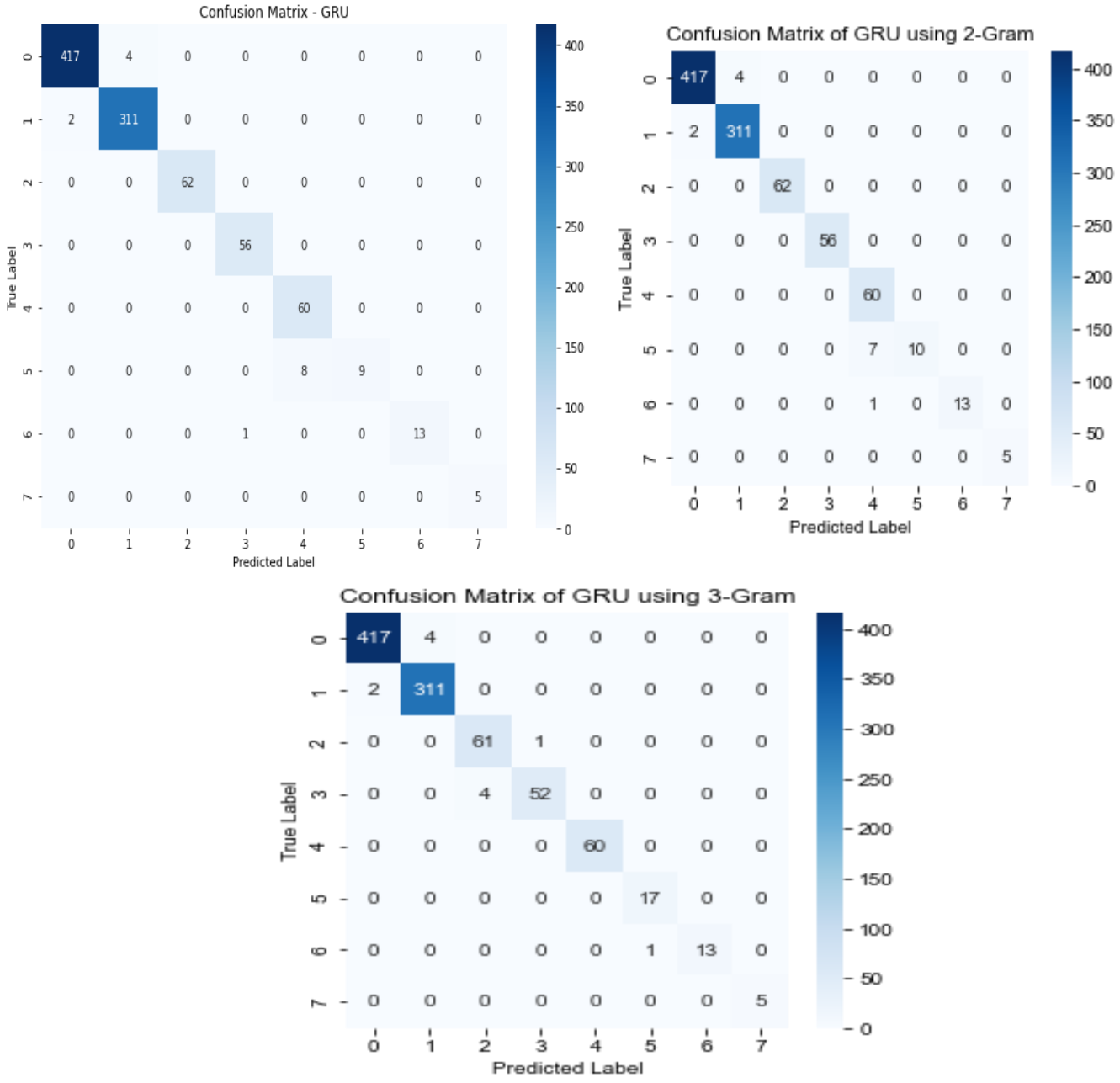
## Appendix A.2. Confusion Matrix of Bidirectional Long Short-Term Memory Network



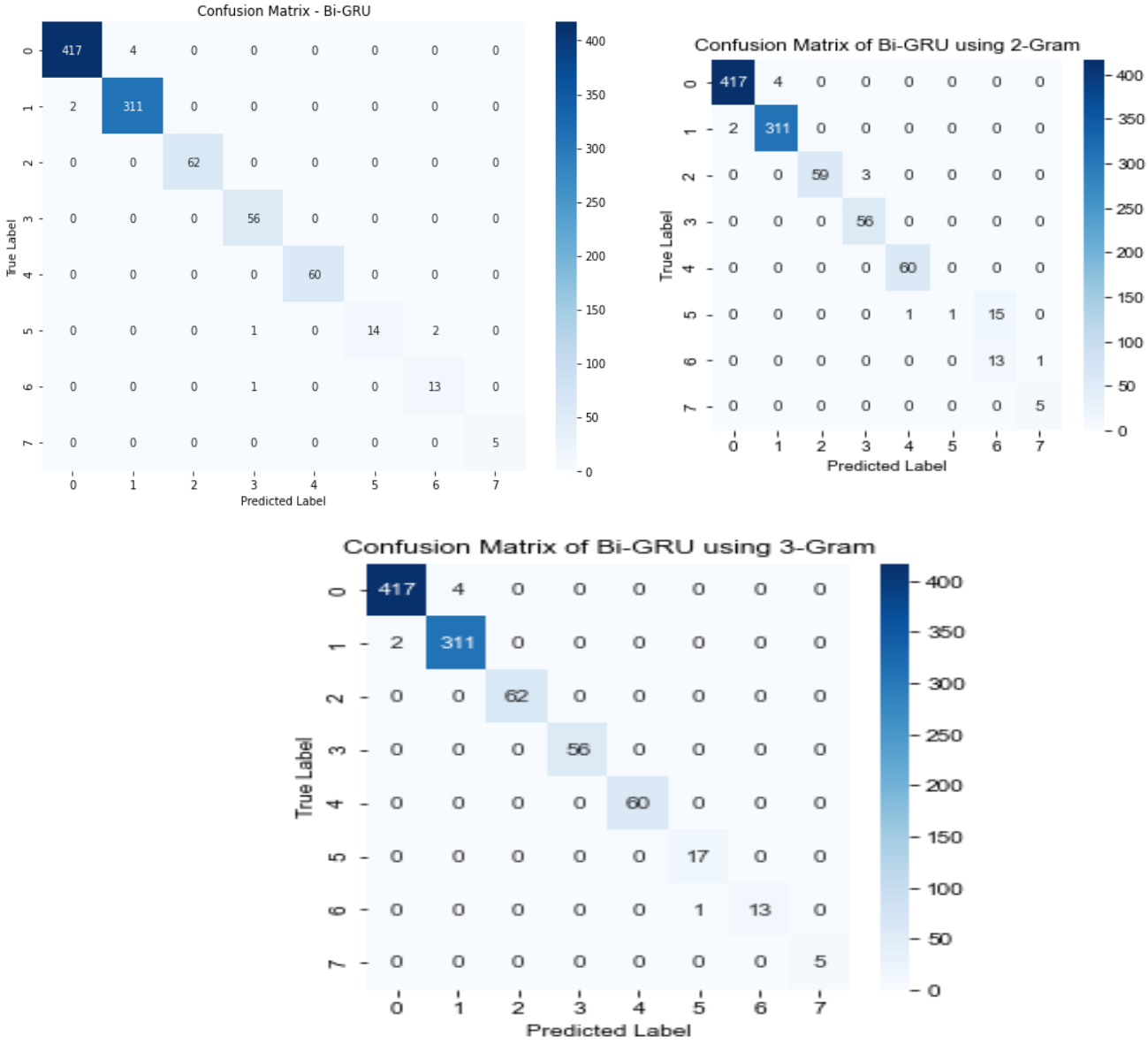

Confusion Matrix of Bi-LSTM with Fast Text for Event Trigger Word

## Appendix A.3. Confusion Matrix of Gated Recurrent Unit



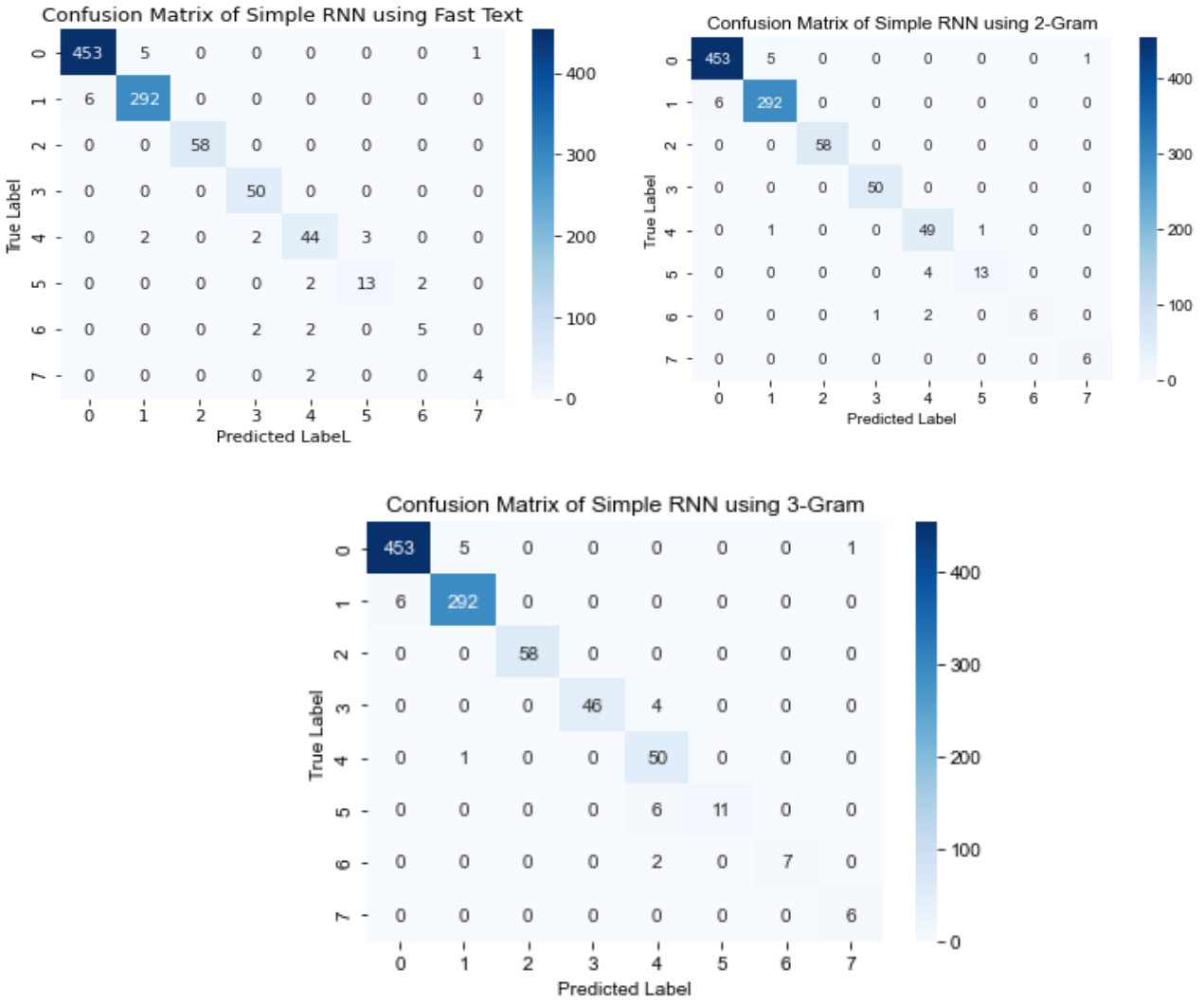

Confusion Matrix of Gated Recurrent Unit with Fast Text for Event Trigger Words

## Appendix A.4. Confusion Matrix of Bidirectional Gated Recurrent Unit




Confusion Matrix of Bidirectional Gated Recurrent Unit with Fast Text for Event Trigger Words

## Appendix A.5. Confusion Matrix of Simple Recurrent Neural Network




Confusion Matrix of Simple Recurrent Neural Network with Fast Text for Event Trigger Words

# Appendix B. Classification Report

## Appendix B.1. Classification Report of LSTM for Event Trigger

LSTM Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.99 | 0.99 | 421 |
| 1 | 0.99 | 0.99 | 0.99 | 313 |
| 2 | 0.98 | 1.00 | 0.99 | 62 |
| 3 | 0.98 | 0.98 | 0.98 | 56 |
| 4 | 0.95 | 0.92 | 0.93 | 60 |
| 5 | 0.93 | 0.82 | 0.87 | 17 |
| 6 | 0.68 | 0.93 | 0.79 | 14 |
| 7 | 1.00 | 0.60 | 0.75 | 5 |
| accuracy | | | 0.98 | 948 |
| macro avg | 0.94 | 0.90 | 0.91 | 948 |
| weighted avg | 0.98 | 0.98 | 0.98 | 948 |

Classification Report of LSTM with Fast Text for Event Trigger Words

**Appendix B.2. Classification Report of Bi-LSTM for Event Trigger**

BiLSTM Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 421 |
| 1 | 0.93 | 0.98 | 0.96 | 313 |
| 2 | 0.95 | 1.00 | 0.98 | 62 |
| 3 | 1.00 | 1.00 | 1.00 | 56 |
| 4 | 0.74 | 0.75 | 0.74 | 60 |
| 5 | 1.00 | 0.18 | 0.30 | 17 |
| 6 | 0.73 | 0.57 | 0.64 | 14 |
| 7 | 0.00 | 0.00 | 0.00 | 5 |
| accuracy | | | 0.95 | 948 |
| macro avg | 0.79 | 0.68 | 0.70 | 948 |
| weighted avg | 0.94 | 0.95 | 0.94 | 948 |

Classification Report of  Bi-LSTMwith Fast Text for Event Trigger Words

**Appendix B.3. Classification Report of GRU for Event Trigger**

GRU Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.99 | 0.99 | 421 |
| 1 | 0.99 | 0.99 | 0.99 | 313 |
| 2 | 1.00 | 1.00 | 1.00 | 62 |
| 3 | 0.98 | 1.00 | 0.99 | 56 |
| 4 | 0.88 | 1.00 | 0.94 | 60 |
| 5 | 1.00 | 0.53 | 0.69 | 17 |
| 6 | 1.00 | 0.93 | 0.96 | 14 |
| 7 | 1.00 | 1.00 | 1.00 | 5 |
| accuracy | | | 0.98 | 948 |
| macro avg | 0.98 | 0.93 | 0.95 | 948 |
| weighted avg | 0.99 | 0.98 | 0.98 | 948 |

Classification Report of GRUwith Fast Text for Event Trigger Words

**Appendix B.4. Classification Report of Bi-GRU for Event Trigger**

Bi-GRU Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.99 | 0.99 | 421 |
| 1 | 0.99 | 0.99 | 0.99 | 313 |
| 2 | 1.00 | 1.00 | 1.00 | 62 |
| 3 | 0.97 | 1.00 | 0.98 | 56 |
| 4 | 1.00 | 1.00 | 1.00 | 60 |
| 5 | 1.00 | 0.82 | 0.90 | 17 |
| 6 | 0.87 | 0.93 | 0.90 | 14 |
| 7 | 1.00 | 1.00 | 1.00 | 5 |
| accuracy | | | 0.99 | 948 |
| macro avg | 0.98 | 0.97 | 0.97 | 948 |
| weighted avg | 0.99 | 0.99 | 0.99 | 948 |

Classification Report of Bi-GRUwith Fast Text for Event Trigger Words

**Appendix B.5. Classification Report of Simple-RNN for Event Trigger**

Classification Report of Simple RNN using Fast Text :

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.98 | 0.99 | 459 |
| 1 | 0.98 | 0.98 | 0.98 | 298 |
| 2 | 1.00 | 1.00 | 1.00 | 58 |
| 3 | 0.98 | 1.00 | 0.99 | 50 |
| 4 | 0.80 | 0.96 | 0.88 | 51 |
| 5 | 0.92 | 0.65 | 0.76 | 17 |
| 6 | 1.00 | 0.56 | 0.71 | 9 |
| 7 | 1.00 | 0.67 | 0.80 | 6 |
| accuracy | | | 0.97 | 948 |
| macro avg | 0.96 | 0.85 | 0.89 | 948 |
| weighted avg | 0.97 | 0.97 | 0.97 | 948 |

Classification Report of Simple RNNwith Fast Text for Event Trigger Words