Deep Learning Judicial Decision Prediction for Afaan Oromo Multi-Defendant Cases

Leti Wakuma Turi

A Thesis Submitted to the Department of Computer Science and Engineering
School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2022

Adama, Ethiopia

Deep Learning Judicial Decision Prediction for Afaan Oromo Multi-Defendant Cases

Leti Wakuma Turi

Advisor: Teklu Urgessa (Ph.D)

A Thesis Submitted to Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

September 2022

Adama, Ethiopia

DECLARATION

I hereby declare that this Master Thesis entitled "**Deep Learning Judicial Decision Prediction for Afaan Oromo Multi-Defendant Cases**" is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Leti Wakuma Turi _____ _____

Name of student Signature Date

RECOMMENDATION OF ADVISORS/ SUPERVISORS

I, the major advisor/supervisor of this research thesis, hereby certify that I have closely advised/supervised the student while developing this thesis and read the draft thesis/dissertation entitled "**Deep Learning Judicial Decision Prediction for Afaan Oromo Multi-Defendant Cases"** prepared under my guidance by Leti Wakuma Turi. Therefore, I recommend the submission of the thesis to the department for further review and evaluation.

_____	_____	_____
Major Advisor/Supervisor	Signature	Date
_____	_____	_____
Co-advisor/Co-supervisor	Signature	Date

APPROVAL PAGE

We, the undersigned, members of the Board of Reviewers of the thesis by **Leti Wakuma Turi** have read and evaluated the thesis/dissertation entitled "**Deep Learning Judicial Decision Prediction for Afaan Oromo Multi-Defendant Cases**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science and Engineering.

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
External examiner	Signature	Date
_____	_____	_____
Internal examiner	Signature	Date

Finally, approval and acceptance of the thesis are contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____	_____	_____
Departement Head	Signature	Date
_____	_____	_____
School Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

ACKNOWLEDGMENT

First and foremost, I would like to express my gratitude to my LORD, Jesus Christ, for letting me through all the difficulties and making this research possible, as well as for creating the beautiful environment in which I could do it.

Next, I would like to convey my profound gratitude to Dr. Teklu Urgessa, my research adviser, for his tremendous guidance, support, and supervision throughout my research study. Working and studying under his guidance was a wonderful honor and privilege.

Additionally, I would like to thank my father, Wakuma Turi, who inspired and encouraged me to choose the legal field as my research area. I'd also want to thank him for sharing his experience with me and making the legal system understandable. A special thanks to my mother, Kume Dereje, for her unwavering prayers and support.

Last but not least, I want to thank every one of my family members and friends for sharing their wisdom with me. They are an important source of guidance and inspiration for me.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATION

AI	Artificial Intelligence
ANN	Artificial Neural Network
BiGRU	Bidirectional Gated Recurrent Unit
BiLSTM	Bidirectional Long Short-Term Memory Network
CBOW	Continuous Bag-of-Words
CNN	Convolutional Neural Networks
CSV	Common Separated Value
CPU	Central Processing Unit
DL	Deep Learning
FDRE	Federal Democratic Republic of Ethiopia
GRU	Gated Recurrent Unit
IDF	Inverse Document Frequency
JDP	Judicial Decision Prediction
JOC	Judgment on Conviction
LR	Gated Recurrent Unit
LSTM	Long Short-Term Memory Networks
ML	Machine Learning
NLP	Natural Language Processing
OCR	Optical Character Recognition
OSC	Oromia Supreme Court
RNN	Recurrent Neural Networks
SVM	Support Vector Machine
SCV	Stratified Cross Validation
TF-IDF	Term Frequency–Inverse Document Frequency

ABSTRACT

The goal of Judicial Decision Prediction (JDP) is to predict court rulings based on the accusations in a criminal case. Criminal cases may feature a single defendant or numerous defendants; the latter type of case is referred to as a "multiple defendant case". Multi-defendant cases are difficult since many defendants are accused of a single offense. By automating the process using a real dataset, the use of a deep learning prediction model can reduce complexity and boost the quality of decision-making. Using a newly collected dataset of 1005 criminal cases from the Oromia Supreme Court (OSC), we constructed a deep learning-based model for predicting court outcomes. Our dataset comprises 3101 instances with ten variables, three of which are target variables. We utilized Microsoft's Optical Character Recognition (OCR) to extract text from the scanned text that was in image format. After that, we applied several data preparation techniques such as cleaning and consolidating before performing feature extraction. Models have been created for both the Judgment on Conviction (JOC) and the Penalty, which are the two main components of the final judicial decision in Ethiopia. JOC involves deciding whether a defendant is guilty or not, and a penalty is a sentence imposed on the defendant who is found guilty. Deep learning models including Bidirectional Long Short-Term Memory (BiLSTM), Hybrid Convolutional Neural Network and Bidirectional Long Short-Term Memory (CNN-BiLSTM), Bidirectional Gated Recurrent Unit (Bi-GRU), and Convolutional Neural Network (CNN) with Word2vec and FastText feature extraction are used for both JOC and penalty prediction. With performance parameters like accuracy, recall, f1-score, and precision as well as a confusion matrix, the performance of the models is assessed under stratified 10-fold cross-validation. The performance evaluation findings showed that the CNN model with FastText outperforms other models for judgment on conviction prediction. It achieved an f1-score of 99% and an accuracy of 98.74%. In terms of penalty prediction, hybrid CNN and BiLSTM with FastText outperform other models with 73.29% accuracy and 74% f1-score. While this study uses deep learning to predict judicial decisions in multi-defendant cases, it would be preferable to use similar techniques on a larger dataset that might include all possible punishments.

Keywords: Judicial Decision Prediction, Multiple Defendant, Criminal Case, Criminal Law, Deep Learning, and Natural Language Processing.

CHAPTER ONE

1. INTRODUCTION

1.1 Background of the Study

The "Nations, Nationalities, and Peoples of Ethiopia" are granted sovereign authority under the federal system of government established by the FDRE Constitution. Any law that conflicts with the Constitution is considered to be invalid. Governmental power is first constitutionally divided horizontally between the national and ten autonomous regional states. It provides that the 10 Regional States shall have legislative, executive, and judicial powers over matters falling under state jurisdiction. The federal courts and the state courts, each with their own separate structures and administrations, make up Ethiopia's dual judicial system. On both the federal and state levels, the courts are given judicial power. The State Supreme Court (which also includes a cassation bench to evaluate basic mistakes of state law), the High Court, and the First Instance Court are the three tiers of state courts that are established by the FDRE Constitution. The Supreme Court shall be responsible for ensuring the uniformity and predictability of the judicial services in the region; the high court shall in principle be an appellate court, the responsibility of which shall be to rectify, by way of appeal, the error of law or fact, or any fault in the judgments of the lower courts. And the first-instance court is responsible for deciding cases in its first-instance jurisdiction (Alemayehu Bekele, 2020).

The Oromia Supreme Court is one of the State Supremes in Ethiopia. The OSC has one cassation division, and it has full adjudication to review and overturn decisions in the courts of the first instance, higher courts, and the regular division of itself. In Ethiopia, the decision of the cassation division is a source of law as long as the requirements are fulfilled. The Supreme Court must, in principle, act as an appellate court, and its responsibility must be to appeal any appeal, misrepresentation, the fact of law, or lower court decisions (Oromia Supreme Court Mission, Objectives, and Values, 2021).

A judicial decision is the official result of a lawsuit or trial in a court. A court's decision on the rights and obligations of parties in a case or procedure also includes an explanation of the reasons why the ruling was made. A trial might be held against a single individual or a group of people. Two or more defendants may be charged in the same indictment, if they are accused of

participating in the same conduct or transaction that constitutes a crime or offense. The presence of several defendants in a lawsuit might pose complicated legal concerns and greatly lengthen the settlement of a dispute (CMPA, 2017). Judicial decision-making has its own procedure based on the legal subjects of a case. The Ethiopian legal system categorizes and identifies its legal subjects as civil law, labor law, commercial law, criminal law, business law, family law, tax law, sales law, contract law, tort law, and so forth.

Legal, personal, and ideological issues are just a few of the internal and external influences on judgments (Callaghan, 2013). The era of artificial intelligence is coming; it quietly changes people's daily lives and also brings new opportunities to the current predicament of numerous fields (Law Audience, 2020). Judges are most likely to allow extralegal biases to influence their decision-making to influence their decision-making . Deep learning provides a mechanism for finding such scenarios by predicting judicial decisions with varied degrees of accuracy based on judicial traits or case features (Meyer & Jesilow, 1996). This method also helps to resolve the complexity of cases involving multiple defendants. Judicial decision prediction aims to determine judicial decisions per the facts of a case. In the legal field, NLP and ML can develop prediction models to help better understand how a certain judge or court could decide. For instance, a 2016 study discovered that by combining NLP with machine learning, researchers could predict the outcome of a certain case at the European Court of Human Rights with 79 percent accuracy (Wilkinson, 2020). All most all laws and cases (lawsuits and charges) are written in natural language; therefore, NLP is a key component of understanding and predicting law at scale.

Our study aims to predict the outcome of a criminal multi-defendant case using Oromia Supreme Court cases. Here, the key idea that was explored in this research was that after identifying their role in the crime, they predicted judicial decisions: judgment on conviction and penalty for multi-defendant cases. To achieve this, a different neural network was compared, and the one with the highest accuracy was chosen. The Oromia Supreme Court's working language is Afaan Oromo. So, NLP feature extraction techniques that extract effective features from judicial documents were used.

1.2 Motivation of the Study

The legal system is complex. It explores how courts are acting properly when reviewing, checking, and balancing the power of the competing branches (Astrada, 2017). A few years back, I used to write judicial documents to help my father, who is a judge. That's when I got to know the procedures followed and how things work in the complex environment of the judicial system. It made me aware of the difficulties both litigants and legal professionals face. Litigants can lose what they are worth because of any small or big mistake made by lawyers, which weakens the capacity of judicial systems to guarantee the protection of human rights. Judges and lawyers also deal with complicated cases, and it becomes worse when the case involves multiple defendants since most of the crimes are committed by a group. Unsurprisingly, previous AI studies in this area, especially judicial decision prediction, didn't include multi-defendant cases. So, looking at these issues in the legal field and applications of AI, we are motivated to look forward to predict court outcomes of multi-defendant cases. Additionally, industry results demonstrating that intelligent systems can outperform legal experts serve as a source of motivation for us (Rory Cellan-Jones, 2017).

1.3 Statement of the Problem

Multi-defendant is when several defendants are sued in the same litigation or charged with the same crime. Multiple-defendant cases are more challenging and more complicated than those involving a single defendant (Marcus, 2003). It is difficult to distinguish between the role of the individual defendant, and the link between these defendants (Sun et al., 2019). Because, there is only one textual fact description in a case, and it comprises all of the fact descriptions for all of the defendants in this case. There are also other researchers (Luo et al., 2017), (Bao et al., 2019), (Zhong et al., 2018), (Hu et al., 2018), and (Yang et al., 2019) who mentioned it in their future work. They explained that, this is due to the intricacy of cases involving several defendants and the difficulty in recognizing descriptions of various defendants. Most group-committed crimes are not clear at all (Marcus, 2003). Because of the ambiguity, investigation and prosecution are undeniably powerful tools for law enforcement. Another issue is the large number of defendants charged at the same time. It has a significant impact on the trial; as the number of defendants increases, so does the difficulty in assigning responsibility to these defendants (Marcus, 2003). So far as we know, only (Sun et al., 2019) proposed a model for multi-defendant cases, however,

it’s only for charge prediction. Traditionally, a judge issues judgments based on his or her knowledge, experience, personality, and emotions. Because of this, the ideal judicial judgment of similar cases ended up having different judgments that should be the same.

The other challenging problem in the justice system is corruption. Corruption in the judicial field can be due to political inference or bribery. From simple to complex, man-made mistakes can turn justice into injustice. In Oromia's courts, the only automated system they are using is the case’s time span prediction. It predicts how long a case will take to reach a judicial decision. To our knowledge, only (Assefa, 2021) proposed an Afaan-Oromo-based model that predicts judicial decisions. Even so, the model can’t handle a multi-defendant case.

1.4 Research Questions

The following questions will be answered in this study to address the problems:

1. Which deep learning algorithm is more accurate at predicting judgment on conviction in cases with many defendants?
2. Which deep learning model accurately predicts the penalty for a case involving a multi-defendant?
3. Which feature extraction technique works well with the models?

1.5 Objectives of the Study

1.5.1 General Objective

The general objective of the study is to build a model of judicial decision prediction for Afaan Oromo multi-defendant cases.

1.5.2 Specific Objectives

The following specific objectives must be met to meet the study's overall goal:

- To collect data from OSC.
- To prepare and preprocess dataset for training.
- To build a prediction model for multi-defendant judicial decisions.
- To choose an evaluation technique and metrics.
- To evaluate the performance of the developed model.
- To select the model with high performance.

1.6 Significance of the Study

Once this research is implemented, it will provide numerous benefits to the legal system. First and foremost, it benefits Oromia's courts. By modernizing the system, it is possible to eradicate all human-made errors and build society's trust in justice. The time it takes to decide on a lawsuit will be reduced as well. Every day, society becomes more aware of the benefits of the legal system. As a result, the number of court cases is increasing, as is the load on judges. It gets worse when the case has multiple defendants. Unfortunately, in our world, the most prevalent crimes are those committed by groups, such as robbery, theft, murder, and so on. This study has the potential to minimize the workload of judges and allow for identical decisions on the same case by different judges. The last one is that it might be used as a starting point to study another issue in the legal area, and the dataset can be a wonderful resource for anyone interested in the problem, as this is the first study to analyze multi-defendant cases.

1.7 Scope and Limitation

1.7.1 Scope of the Study

This study aims to use a deep learning technique to provide a model that predicts a judicial decision for multi-defendant cases. Among different types of cases, it focuses on criminal cases. Since criminal cases are radially available, there are numerous similar cases. There are different types of cases examined and judged according to criminal law, like theft, robbery, murder, etc. From those, the most common crimes and most multi-defendant-involved crimes are selected. These cases are written in Afaan Oromo and were collected from OSC. Different neural network models are built to predict judicial decisions from two different aspects: JOC and penalty. The study's goal is to use stratified k-fold cross-validation to assess the model's performance using multiple metrics while training the dataset using recurrent and convolutional neural networks.

1.7.2 Limitation of the Study

However, there are a few limitations to this study. The first is that, due to time constraints, we are unable to prepare any additional data: It could play a great role in improving the accuracy of penalty prediction. The second is that, because of the small number of cases with fine penalties, we did not consider cases with penalties of more than 500 birr. The fine penalty, like imprisonment, has its own penalty level. A fine penalty of up to 500 birr is included since it falls under imprisonment penalty level 1.

1.8 Organization of the Study

The subjects we addressed in our thesis report are briefly covered in this section. A total of seven chapters make up the report.

The introduction chapter, to which this section belongs, focuses on the background of the study, the motivation behind the study, the statement of problems with research questions to be raised and addressed, the scope and limitation of the study, the objectives to be attained, and the application of the study.

The second chapter contains a review of the literature and related works on judicial decision prediction, as well as an overview of the OSC, Ethiopian judicial systems, and the Afaan Oromo language. It also discusses the methodologies used in the previous research: - feature extraction and deep learning classifier.

The third chapter addresses research methodologies. In this, we discussed or included several techniques, tools, methods, deep learning classification algorithms, and model selection methods that we used in our study.

The fourth chapter discusses the design and architecture of the proposed solution.

The experimentation and implementation of the suggested method are covered in chapter five, along with some sample code and an explanation of each line.

The results from experimenting with several deep learning models on our dataset are presented in chapter six.

The final part, chapter seven, concludes this thesis report and makes apparent how this work will continue to be researched in the future.

CHAPTER TWO

2 LITERATURE REVIEW AND RELATED WORKS

2.1 Introduction

This chapter gives a detailed review of existing techniques used in making a judgment for criminal cases. The review provides an introduction to the judicial system and multi-defendant cases, the application of AI in the legal field, deep learning techniques and algorithms that have been used in judicial decision prediction, the Afaan Oromo language, and related work previously studied on the problem.

2.2 Judicial System

The force behind maintaining appropriateness and decency in our society is the law. There isn't a single aspect of a company that hasn't been impacted by the legislation. Every transaction, including sales, purchases, mergers, acquisitions, partnerships, and more, involves a legally enforceable agreement (Rangaiah, 2020).

The judicial system is the system of courts that interprets, defends, and applies the law in legal matters. The term is widely used to refer to the courts, magistrates, judges, adjudicators, and other support staff who run the system. Courts are a good approach to settling conflicts fairly. Justice is the foundation of a stable social order. The judiciary normally does not enact or enforce the rule of law, but rather interprets, advocates, and applies facts to every case. Land and other asset conflicts have driven growing violence in numerous nations. A more formal method of exercising public power over disputants was utilized in the ancient Near East, the Carolingian empire, and medieval France (Commentary & Annals, 2002). A court of law provides an alternative, one in which facts are thoroughly examined and self-defense and other elements that may justify or explain the action are considered. As a result, the legal and judicial systems must provide a means for judging the truth and fairness of private agents' and the state's activities. Its major role is to maintain societal harmony.

In Ethiopia, the judicial system (as well as the codification of laws) was consolidated and streamlined during the regime of Emperor Haile Selassie I, who ascended the throne in 1930 and ruled the country until 1974. During his reign, two constitutions, the Constitution of 1931

and the Revised Constitution of the Empire of Ethiopia of 1955, and six basic codes were enacted. The 1931 Constitution provided for two separate systems of courts, i.e., the regular courts and special courts. The former was entrusted with the adjudication of civil and criminal cases, while the latter was authorized to deal with cases relating to administrative affairs The Federal Democratic Republic of Ethiopia (FDRE constitution), which was adopted in 1995, replaced "the Empire of Ethiopia" with the name "the Federal Democratic Republic of Ethiopia" (Vibhute, 1970).

Following the legal system, the FDRE Constitution provides for the formation of a three-tier 'regular' court system at the regional (state) and federal levels. The federal and state courts have separate systems and administrations. Judicial powers, both at federal and state levels, are vested in the courts. Each High Court has a civil, criminal, and labor division with a presiding judge and two other judges in each division. The Federal Supreme Court includes the Cassation Division, which has the power to examine and overturn decisions made by the lower federal courts, the Supreme Court itself, the regular division, and the state Supreme Court. In addition, the decisions of the Federal Supreme Court's Cassation Court on the interpretation of laws apply to both federal and state courts[1].

In a criminal case, the defendant is put on trial for behavior that the state legislature, or the government, deems to be against the law. Criminal cases frequently begin when a person is arrested and informed of their accusations, generally at an indictment hearing [2]. According to (Imperial Ethiopian Government, 1969), criminal procedure controls the entire process of detection, investigating, prosecuting, and punishing offenders. It has two functions. It enforces criminal laws. At the same time, it helps protect the rights of the suspect or defendant. In addition, in contrast to civil cases, criminal proceedings can result in the loss of life and liberty. However, the court proceedings code includes the process of how judges give judgments against offenders, how the prosecutor prepares charges, and how the defendant defends his or her rights.

[1] https://www.nyulawglobal.org/globalex/Ethiopia1.html
[2] https://www.legalmatch.com/law-library/article/what-is-a-criminal-case.html

2.3 Multi-Defendant Cases

A person who is being tried in court on charges of breaking the law is known as a defendant[3]. If two or more people are charged or accused of the same crime, they're considered as multi-defendants. When going through the legal process, they can either have separate trials or have them together, depending on the case (Karissa Key, 2020). In contrast to torts, the burden of proof in a criminal trial is on the government. Defendants are not required to prove their innocence. The burden of evidence shifts on the government to persuade the jury of the defendant's guilt. In a criminal trial, the burden of proof is substantially higher on the prosecutor than it is on the plaintiff. There must be sufficient evidence that proves the defendant committed the offense "beyond a reasonable doubt" to convict someone of a crime[4]. For a case to include multi-defendants, there must have been several people charged with the commission of the same crime. It can be a murder charge, drug charge, robbery charge, etc.

The penalty is the more important of the two components of a judicial decision, the judgment on conviction and the penalty. These decisions all reflect the belief that defendants if proven guilty, deserve harsh treatment. The decisions, however, do not imply that the mechanism by which we establish accountability has been altered in any way. Fairness must be maintained at all levels of the investigation, pre-trial, and trial. The burden remains on the government to establish each element of each charge beyond a reasonable doubt. Furthermore, criminality must be decided on an individual basis, even when multiple defendants are involved (Marcus, 2003)**.**

Judgments of multi-defendant cases vary from country to country. In Texas, the judge is asked to assign responsibility to each party. The penalty that the plaintiff receives can be significantly impacted by the percentage of the responsibility[5]. According to Ethiopian Criminal Code articles 32/3 and 41, defendants are judged and sentenced based on their roles in the crime as individuals and the whole action of the crime, which is common to all defendants (GoE, 2005). A judgment is about deciding if the defendant pleads not guilty or guilty. If the defendant is guilty, the judge determines the defendant’s sentence. A sentence may include time in prison, a fine to be paid to the government, or restitution to be paid to crime victims.

[3] https://www.collinsdictionary.com/dictionary/english/multiple-defendants
[4] https://www.uscourts.gov/about-federal-courts/types-cases/criminal-cases
[5] https://baumgartnerlawyers.com/blog/happens-multiple-defendants/

In Ethiopia, when a crime is committed by a group of people, a prosecutor charges them together based on the same evidence and witness. So they can't be charged apart unless the defendant has a special case like age (GoE, 2005). In other countries, even though defendants commit the crime together, they are charged separately, but a judge can join their trials, which are known as joint trials[6]. Joint trials take place where it is possible to maintain the right to a fair trial and when the charges brought against codefendants are closely related. A judge typically joins trials when there is a specific reason and also if the prosecution has charged the defendants together, which is rare.

2.4 Artificial Intelligence in the Legal Domain

Nowadays, from law firms and corporate practices to courtroom procedures and document management, legal technology has impacted every aspect of the legal profession. Modern software is now capable of analyzing legal papers, streamlining communications, and identifying cases that would be appropriate for legal practitioners to work on. This is made feasible by these developments in artificial intelligence and big data (Rangaiah, 2020). In a variety of situations, AI is adopting algorithms and machine learning to perform tasks that were formerly handled by entry-level lawyers.

AI has been applied to the legal or law field to improve the various problems that the legal profession has faced over the years, and it is now being implemented. It can boost productivity and help lawyers avoid costly mistakes. In some circumstances, it can also speed up the process of justice by facilitating research and decision-making (Stepka, 2022). Within the law field, a couple of aspects have been highlighted that prove to be promising for the application of AI. A few of them are:

Litigation Prediction: Various AI teams have started developing machine learning models for predicting the results of imminent cases by taking into account the factual pattern of the case as well as the bulk of relevant paradigms(Ward, 2014).

Contract review: Our economic system depends on contracts; business transactions would not be possible without them. Software companies are creating AI systems that can automatically

[6] https://www.nolo.com/legal-encyclopedia/when-defendants-joint-trials.html

swallow proposed contracts, fully analyze them using NLP technology, and identify the contracts as acceptable and problematic (Toews, 2020).

Case-text: Case-text's CARA (Case Analysis Research Assistant) is one such software business that enables lawyers to predict opposing counsel's arguments by locating previously utilized opinions by lawyers. It can also recognize untrustworthy scenarios and indicate them adversely (Pizzinini, 2021).

Case Outcome Prediction: Most AI in legal research aim to develop a computer model of legal reasoning that can present arguments and predict the outcome of legal disputes. They do this by frequently employing case-based reasoning (CBR) models or machine learning (ML) methods, or by combining the two. Case-specific legal characteristics, such as the nature of evidence, and extra-legal aspects, such as the court's ideological viewpoint, can be used to anticipate outcomes (Athar, 2020b).

2.5 Judicial Decision Prediction

In 1957, set out to demonstrate that it is feasible to take some decided instances, determine the factual factors that impacted the judgments, generate numerical values for these items using a formula, and then properly anticipate the outcomes of the remaining cases in the designated region (Kort, 1957). Based on this, In 1960 (Nagel, 1960) presents a predictive method that uses five hypothetical past criminal confession cases (A, B, C, D, and E) involving four hypothetical variables (p, q, r, and s) that will be used to predict one hypothetical future case, case X, by calculating a correlation coefficient and point value (100 times the correlation coefficient of the variable). In actual applications, more past cases and more variables are likely to be involved. Both of the above studies depend solely on mathematical calculation. Later (Lawlor, 1963), introduced the types of assistance a computer can offer to legal professionals. They can help find the law, and analyze the law, and they can help lawyers and lower court judges predict or anticipate decisions by employing modern logic.

Daniel Martin Katz and his team at Michigan State University's law school developed an algorithm in 2014 that predicts the outcome of cases considered by the U.S. Supreme Court (Ward, 2014). Unlike previous studies, data is the most important part, and it was the time when computers were used for other purposes rather than making a calculation and storing a few bits

of data. A bunch of researchers have been working on judicial decision prediction and trying to come up with a new model that can solve the problem in the legal system. Most of the studies have used machine learning. An application of AI called machine learning allows systems to learn from their past performance without having to be explicitly programmed. Machine learning seeks to develop computer algorithms that can access data and use it to learn on their own[7]. Both Support Vector Machines (SVM) and Logistic Regression (LR), two well-known machine learning techniques, are employed in predictive coding. Nowadays, with the successful usage of neural network methods on artificial intelligence tasks, researchers have begun to employ neural network models for judicial decision prediction tasks, and they are gaining good results. Current studies tend to prefer deep learning algorithms since their performance improves as the amount of data rises, whereas the performance of traditional machine learning algorithms improves to a point but then remains constant.

2.6 Deep Learning

Deep learning is a subtype of machine learning that consists of a neural network with three or more layers based on a specific task. These neural networks try to mimic the activity of the human brain by allowing it to 'learn' from enormous amounts of information[8]. Simply it's concerned with algorithms inspired by the building and function of the brain. When given raw data, the algorithm determines which attributes are meaningful on their own. DL networks are frequently enhanced as the quantity of data utilized to train them increases.

Deep learning uses ANN to conduct complex computations on huge amounts of data. There are several kinds of DL algorithms. CNN, RNN, LSTM, Multilayer Perceptron, and Generative Adversarial Networks are the most common deep learning algorithms. However, to select the optimal algorithm for the given issue, we must first understand how to apply these full algorithms. DL architectures provide a lot of advantages for text classification since they can perform extremely well with low-level engineering and computation. The two fundamental deep learning architectures for text classification are Recurrent Neural Networks (RNN)

[7] https://www.expert.ai/blog/machine-learning-definition/
[8] Ibm.com/cloud/learn/deep-learning

and Convolutional Neural Networks (CNN)[9]. Various models of CNN and RNN have been developed for retrieving and classifying relevant legal text.

The integration of deep learning and other AI techniques in legal services will speed up the judicial system's entire process. Deep learning models will be used to minimize time, effort, and overall cost in tasks such as legal data search, predictive systems, information retrieval, extraction of relevant text, intelligent interfaces, and legal conversational agents (Lucarelli & Borrotti, 2019).

2.6.1 Convolutional Neural Network (CNN)

CNN is a feed-forward artificial neural network (ANN) that employs a form of multilayer perceptron designed to need minimal preprocessing and to make use of local connection and weight sharing, which dramatically decreases the number of parameters required to learn in the network. Though CNN is more commonly connected with computer vision issues, they have lately been applied in NLP with promising results. CNN is just a sequence of convolutional layers with non-linear activation functions added to the output such as ReLU, Tanh, or SoftMax[10]. In a CNN for text classification, convolution layers involve one-dimensional convolution with a small size kernel to extract features, and max pooling to condense or summarize the features extracted from the convolution. Finally, the fully connected layer takes the features through activations and fits into the training data, and makes predictions (Wei et al., 2019).

Let's assume that the input text sequence has n words, each of which is represented by a d-dimension word vector. The input example, therefore, has n-width channels, 1 height channel, and d input channels. The stages that make up the majority of the CNN calculation in the text are as follows [11]:

- Convolution operations on the inputs are carried out using multiple one-dimensional convolution kernels that have been defined. The correlation of various numbers of neighboring words may be captured using convolution kernels of various widths.

[9] https://monkeylearn.com/text-classification/
[10] www.analyticsvidhya.com/intent-classification
[11] https://classic.d2l.ai/chapter_natural-language-processing/sentiment-analysis-cnn.html

- After all output channels have been subjected to max-over-time pooling, combine the pooling output values of these channels into a vector.
- Through the fully connected layer, the concatenated vector is changed into the output for each category. To address overfitting in this stage, a dropout layer might be applied.

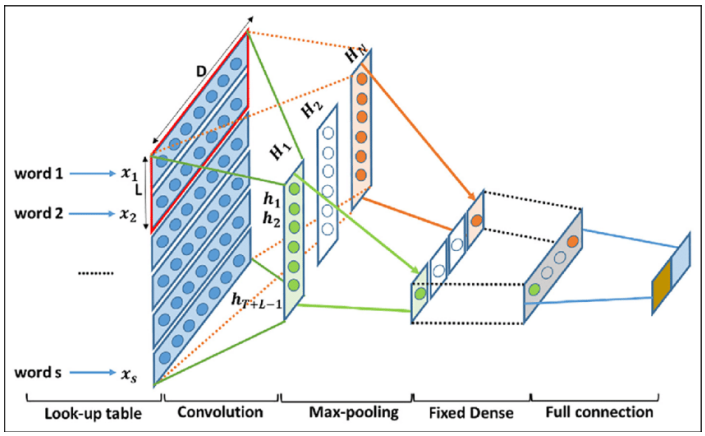

Figure 2.1 1D CNN model architecture[12]

(B. Chen et al., 2019), use the TextCNN model to analyze the fact description of the case and predict the sentence results from three aspects: penalty, accusation, and legal provisions.

2.6.2 Recurrent Neural Network (RNN)

RNN is made to function with time series or sequence data common feed-forward neural networks are designed just to handle disparate chunks of data. If the data in a series contain data points where one data point depends on the data point before it, the neural network must be changed to account for these dependencies. RNNs can "remember" the states or data of earlier inputs to create the next output in the sequence (Saeed, 2021). This makes them useful for time-based applications (audio, time-series data) as well as natural language processing RNNs perform well in situations where sequential information is critical since, without it, the meaning or grammar may be misinterpreted. Applications include image captioning, language modeling, and machine translation[13].

[12] https://www.researchgate.net/figure/Standard-CNN-on-text-classification_fig1_333752473
[13] https://towardsdatascience.com/text-classification-rnns-or-cnn-s-98c86a0dd361

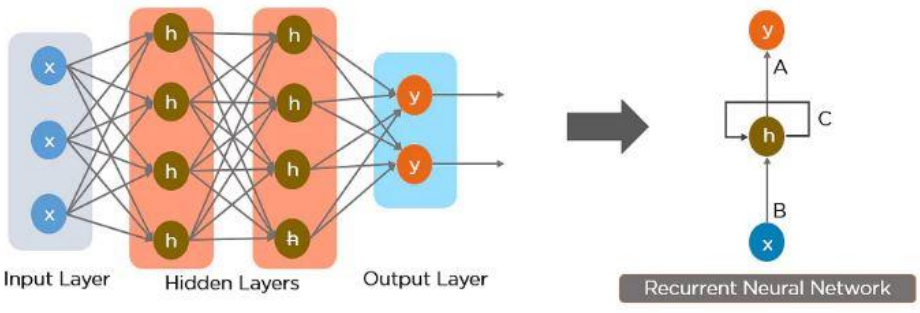

Figure 2.2 Fully connected recurrent neural network (RNN) architecture [14]

The input layer is "y," the hidden layer is "h," and the output layer is "x." The network parameters A, B, and C are used to improve the output of the model. The current input is the sum of input at y(t) and input at y(t) at any given time t(t-1). At each given time, the output is collected from the network in order to enhance it. The RNN was developed in response to many concerns with the feed-forward neural network: cannot handle sequential data, just evaluates the current input, and does not remember earlier inputs[14]. A CNN is trained to recognize patterns over space, whereas an RNN is trained to detect patterns across time.

2.6.3 Long Short-Term Memory networks

When working with longer data sequences, standard Recurrent Neural Networks (RNNs) suffer from short-term memory due to a vanishing gradient problem. Fortunately, we now have more powerful RNNs that can maintain and carry over critical information from earlier sections of the sequence. Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) are the two most well-known versions (Dobilas, 2022). Recurrent neural networks with order dependency, such as LSTM networks, can figure out sequence prediction issues. The LSTM recurrent unit is substantially more complicated than the RNN recurrent unit, which improves learning but requires more processing power. LSTM has 3 main gates: forget gate, input gate, and output gate (Shraddha Shekhar, 2021).

In LSTM, we may use a multiple-word string to identify which class it belongs to. When working with natural language processing, this is quite useful. The LSTM model will be able to

[14] https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

discern the true meaning of the input text and deliver the most correct output class if the right layers of embedding and encoding are employed.

2.6.4 Bidirectional gated recurrent unit (Bi-GRU)

A gated recurrent unit (GRU), which functions as a gating mechanism in recurrent neural networks, is comparable to an LSTM with a forget gate but has fewer parameters since it lacks an output gate. It was discovered that GRU's performance on tasks including polyphonic music modeling, speech signal modeling, and natural language processing was comparable to LSTM's[15]. That is why GRU can be considered a variation on the LSTM**.** The update gate and reset gate are used by GRU. In essence, two vectors determine what data should be sent to the output. They are unique in that they can be trained to retain knowledge from the past without having to wash it away over time or delete information that is irrelevant to the prediction (Simeon, 2017). There is no specified memory unit in GRU. The network and the memory unit are merged.

A Bidirectional GRU, often known as a BiGRU, is a sequence processing model made up of two GRUs. One takes information in a forward direction, whereas the other takes it backward. Only the input and forget gates are present in this bidirectional recurrent neural network.

2.6.5 Attention mechanism

A neural network is a computer program that attempts to emulate the operations of the human brain in a simplified fashion. The attention mechanism in deep neural networks attempts to mimic the comparable behavior of selectively focusing on a few important items while disregarding others (Prodip Hore, 2019). Since the decoder would only have limited access to the data provided by the input when utilizing a fixed-length encoding vector, the attention technique was developed by (Bahdanau et al., 2015) to address this bottleneck issue. It is a component of a neural architecture that allows users to dynamically highlight significant parts of incoming data, which in NLP is often a series of textual components.

Instead of using the entire sequence to predict an output word, the model just uses portions of the input where the most important information is concentrated. In other words, it only pays attention to a small number of input words. Utilizing this structure, the model may focus only

[15] https://en.wikipedia.org/wiki/Gated_recurrent_unit

on the most important portions of the input sequence and therefore understand the relationships between them. As a result, the model can handle longer input phrases more easily (Nagesh Singh Chauhan, 2011).

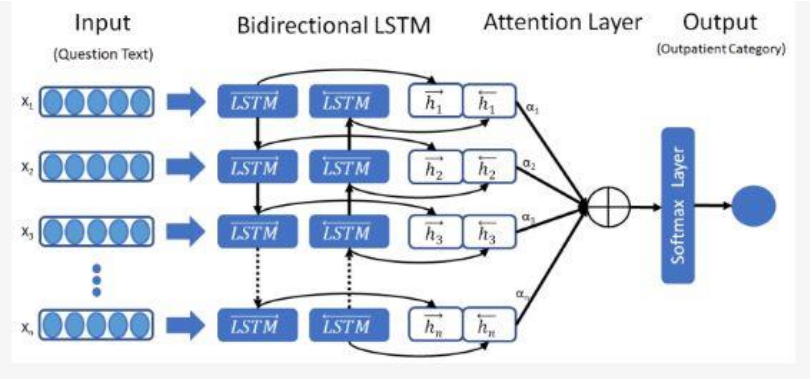

Figure 2.3 Architecture of attention-based Bidirectional LSTM model **(C. W. Chen et al., 2020)**

2.7 Feature Extraction used in Legal Texts

Text feature extraction, which extracts text information to represent a text message, is the foundation of a large number of text processing algorithms (V. Singh et al., 2013). Feature extraction is a method that involves selecting a group of features from a variety of effective approaches to lower the dimension of feature space. After the initial text has been cleaned and normalized, we must convert it into characteristics that may be utilized in modeling. Before modeling, we employ a specific way of giving weights to certain phrases in our content. Individual words are represented numerically since numbers are easier for the computer to process; in these circumstances, word embeddings are used (Siddharth, 2021).

2.7.1 Bag of Words

Text modeling with a bag of words is a Natural Language Processing approach. In technical terms, we can call it a method for extracting features from text data (Great Learning Team, 2020). In this method, we treat each document as a collection or bag, having all the words in it. Analyzing the documents is the idea. In this document, a unit is referred to. Each tweet on this page is a document in case we wish to search for all the bad tweets made during the pandemic. We always take all the necessary precautions, including cleaning, stemming, lemmatizing, etc.,

to get the bag of words. Then we generate a set of all the words that are available before sending them for modeling (Siddharth, 2021).

2.7.2 Term Frequency — Inverse Document Frequency

The acronym TF-IDF stands for "Term Frequency—Inverse Document Frequency". This method may be used to determine the number of words in a collection of documents. Each word is typically given a score to show how important it is in the document and corpus. This technique is widely utilized in the areas of text mining and information retrieval (Scott, 2019).

Term frequency is the frequency with which a word appears in a document (TF). The easiest way to determine this frequency is to just count the instances of each word in the text. The word's average inverse document frequency (IDF) over a collection of documents. This indicates how common or uncommon a word is over the entire document collection. The closer the number gets to zero, the more popular the word becomes. The entire number of documents may be divided by the number of documents that include a word, and then multiplied by the total number of documents to find the logarithm.

2.7.3 Word embeddings

Word embeddings are a kind of word representation that enable words with comparable meanings to have comparable representations. They are a distributed representation of text that may be one of the main breakthroughs in deep learning approaches' excellent performance on difficult natural language processing issues[16]. An embedding is a dense vector of floating-point data. They are trainable parameters rather than straight embedding values (weights learned by the model during training). When dealing with huge datasets, it is usual to observe word embeddings that are 8-dimensional (for small datasets) and up to 1024-dimensional. Although it requires more data to learn, a higher-dimensional embedding can capture the best associations between words (Siddharth, 2021).

Following (Mikolov et al., 2013) proposal of word2vec, a neural network model for word representation, it has been demonstrated that neural network topologies are more convenient and efficient than several hidden layers due to their simple projection layer. The word2vec

[16] https://machinelearningmastery.com/what-are-word-embeddings/

architecture is designed to be trained across a huge amount of data to obtain a high-quality word representation.

CBOW (Continuous Bag of Words) and Skip-Gram are the two most used models. In essence, both neural network architectures aid the network in learning how to represent a word. This is unsupervised machine learning, which necessitates the use of labels to train the model. Either of these two models can generate labels for the input and prepare the neural network for training and performing the required task (Fangyug, 2020). Both models move a window of a set length along the corpus, and the network is trained using the words inside the window at each step. In contrast to the CBOW model, which is trained to predict the center word depending on the surrounding words, the Skip-gram model is taught to anticipate contexts based on the core word. Once the neural network has been trained, the learned linear transformation in the hidden layer is employed as the word representation (Archak, 2018).

FastText is another way of converting words to vectors. It is an open-source library established by the Facebook AI Research Lab. Its major purpose is to analyze massive datasets quickly and accurately while also developing scalable solutions to text categorization and representation difficulties. (Anala, 2020). We can develop supervised or unsupervised learning algorithms using the model to generate vector representations of words. For 294 languages, Facebook offers pre-trained models. FastText embeds words using a neural network. It performs superior to Gensim in terms of syntactic performance and fairs equally well in the case of semantic performance and works better on small datasets (NSS, 2020).

2.8 The Afaan Oromo Language

Afaan Oromo is the language of the Oromo people, who comprise the biggest ethnic group in Ethiopia. The Oromo are Cushitic people who mainly live in the Oromia region of Ethiopia, constituting the largest ethnic group. The Afaan Oromo language is not spoken only by the Oromo people in Ethiopia. Afaan Oromo is probably the third-most widely spoken Afro-Asiatic language in the world, after Arabic and Hausa. It is one of the primary indigenous African languages, extensively spoken and utilized across Ethiopia and some surrounding nations (Bulcha, 1997). The language is now used as a medium of teaching from elementary school to graduate school in the regional offices of Oromia. Afaan Oromo has a very rich morphology

like other African and Ethiopian languages. The history of written Afaan Oromo started in the first part 19th century (Tegegne, 2016). Religion especially Christian and Muslim Oromo scholars, played a pivotal role in the development of the Afaan Oromo literature and writing system. According to (Bulcha, 1997)Oromo religious leaders and academicians have worked to make Afaan Oromo a literate language.

2.8.1 Afaan Oromo Alphabets and Writing System

The current orthography of the language is based on the work of a committee formed in the neighboring country of Somalia at the beginning of the 1980s. The members of this committee, which was the first to accept the Latin alphabet in its current form, had personal experience writing Afaan Oromo in different scripts. They were a group of passionate amateurs who were primarily influenced by Dr. Muhammed-Rashad Abdulle's earlier writings. In 1976, in Somalia, where the Latin-based alphabet had already been formally approved for the Somali language, Abdulle released an Oromic Reader (Youssouf, 2018). Although it employs the Roman alphabet, the Afaan Oromo script, also known as "QUBEE Afaan Oromo," has its own consonants and vowels (Roba, 2001).

Afaan Oromo is made up of thirty-three fundamental letters, including five vowels and twenty-four consonants. Seven of them are paired letters that fall together (for example, 'DH'). Vowels in the Afaan Oromo language are sound creators as well as sound by themselves, much like in the English language. Vowels in Afaan Oromo are characterized as short and long vowels. The basic alphabet in Afaan Oromo does not contain ‘p’, ’ ts’, ’zh’, ’v’, and ‘z’. They are loaned letters because there are no native words in Afaan Oromo that are formed from these characters. However, they are employed to refer to foreign terms like "polisii" (police) while writing in Afaan Oromo (Tesema & Tamirat, 2017).

Table 2.1 Afaan Oromo alphabets

A a	B b	C c	CH ch	D d	DH dh	E e	F f	G g	H h	I i
J j	K k	L l	M m	N n	N Y	O o	P p	PH ph	Q q	R r
S	SH	T	TS	U	V	W	X	Y	Z	ZH

s	sh	t	ts	u	v	w	x	y	z	zh

Oromo includes a set of ejective consonants, that's, voiceless stops or affricates that are followed by glottalization and an explosive burst of discussion. The primary one is famous as a punctuation mark and is called 'hudhaa' representing punctuation. It plays a vital part in the Afaan Oromo reading and writing framework. For example, it is utilized to type in a word in which most of the time, two vowels show up together(Youssouf, 2018). For example, mi'ii means new drain, whereas "mii" (without punctuation) implies isn't. Another uncommon glottalized phone in Oromo is called an implosive retroflex stop, or "dh" in Oromo spelling. It is similar to the English "d" and is produced by slightly twisting the tongue back and drawing the mouth in such that a glottal stop has been heard before the next vowel begins [17].

2.9 Related Work

The emergence of newer technology is continuing to change how the world functions and works. The prospect that such technology may substitute for or enhance human operations is not limited to the judicial system. Recent developments in artificial intelligence (AI), particularly in the fields of natural language processing (NLP) and machine learning (ML), provide us the means to automatically evaluate legal texts and create prediction models for judicial outcomes based on the semantics of legal and case texts. Researchers in the legal area have been interested in using automated analytic techniques for judicial judgment for years.

The work of (Virtucio et al., 2019) is "Predicting Decisions of the Philippine Supreme Court Using Natural Language Processing and Machine Learning. The goal of their study was to predict Philippine Supreme Court case outcomes as affirmed or reversed. They worked on an appeal case, those previously decided by lower courts, and escalated to the Supreme Court for reconsideration. To achieve their goal, they have used the SVM algorithm and bag-of-words feature extraction without any comparison with other algorithms and feature extraction. Eighty percent of their dataset was utilized for training, while twenty percent was used for testing. A subset test set accuracy can reach as high as 68% and 59% on average.

[17] https://en.wikipedia.org/wiki/Oromo_language

In (B. Chen et al., 2019) study, legal judgment prediction has multiple tasks. After analyzing the basic description of the case, their deep learning-based model predicts the judgment results from three aspects: penalty, accusation, and legal provisions. It is divided into three models. The first one is the prediction model of the penalty based on FastText and TextCNN. The average accuracy of TextCNN is 88.76%. The second one is the legal provisions prediction model; they have used the TF-IDF feature extraction technique along with the TextCNN algorithm and achieved 86.27% accuracy. The last one is the accusation prediction model. For this one, they made a comparison among FastText, TextCNN, and MLKNN algorithms, and TextCNN had the highest accuracy, which is 60.33%. Their model can't handle complex law cases, to illustrate: the accusation of multiple defendants with a single crime.

(Sun et al., 2019) proposed a charge prediction for multiple defendants with multiple scale attention, global and local. The important thing is figuring out the right charges for a particular criminal based on their identity and their record. They used both local attention, which is closely tied to where a single defendant's name appears in the fact description for that defendant, and global attention, which was used to enrich the model with overall case-relevant data. They employed attention-based BI-GRU in their multi-scale attention to complete those tasks. Even though their model can solve the intricacies of cases that involve multiple defendants; it still cannot distinguish confusing charges. They didn't conclude the average accuracy they got.

(Luo et al., 2017) present an attention-based neural network architecture for concurrently modeling the charge prediction problem and the significant article extraction task in Chinese criminal cases, where the weighted relevant articles can serve as a legal basis to support the charge prediction. First-word embedding was trained using word2vec on judgment documents, then they used sentence-level and document-level BiGRU, to capture the whole story as well as important details of the case. From their experiments on news data, they showed that their model also has reasonable generalization ability on fact descriptions written by non-legal professionals and achieved 79.12% accuracy. But it still cannot explicitly handle multi-defendant cases. Their models are intended for a certain set of subtasks and are difficult to scale to different subtasks.

In real-world scenarios, the judicial decision often comprises several smaller responsibilities, such as deciding on charges, the length of the sentence, and the application of the relevant legal

provisions. Furthermore, these subtasks are interdependent. TOPJUDGE is a topological multi-task learning framework provided by (Zhong et al., 2018) that formalizes subtask dependencies as a Directed Acyclic Graph (DAG) and incorporates a large number of subtasks and DAG dependencies into judgment prediction. They employed a fact encoder to generate the fact description's vector representation to the input of TOPJUDGE based on a Convolutional neural network and employ a specific LSTM cell for each task to get the output of each task in the topological order. The model achieved 86.3% for the article, 85.7% for the charge, and 33.1% for the penalty prediction. They have shown the dependencies exist between law article, charge, and penalty, which plays a crucial role in the accuracy of the judicial last decision (penalty). However, their model predicts penalties in intervals rather than predicting the exact period of imprisonment.

In Ethiopia, not much research has been conducted in this field. (Alem, 2014) presented a case-based reasoning system that manages case-based reasoning tasks of retrieving, reusing, revising, and retaining cases in the domain of Ethiopian labor law using a knowledge-based system. (Mesfin, 2009) aimed to predict a court case's time span that benefits a plaintiff and defendants to know their case time length in prior as well as the federal courts to perform courtroom monitoring, ensuring transparency and work efficiency. A model to address these needs was constructed using a feed-forward multilayer neural network perceptron. (Assefa, 2021) constructed a model that predicts judicial decisions from two aspects: the accusation (identifying whether the suspect committed the alleged crime or not) and the penalty (if the suspect is found guilty of the alleged crime, impose a penalty) using a criminal case dataset collected from OSC. To carry off these tasks, he has used RF and SVM models with TF-IDF feature extraction. He has achieved 98.5% accuracy for accusation prediction and 79.68% accuracy for penalty prediction.

Summary of related work

Employing ML and DL techniques for judicial decisions has drawn the attention of researchers in the legal domain for decades. Even though legal judgment has multiple tasks, early works usually focus on a single task like article prediction, charge prediction, affirmed or reversed case prediction, etc. Other researchers (B. Chen et al., 2019) and (Zhong et al., 2018) added the ultimate step (penalty) to their model, which fills the gap left by former researchers. However,

it didn’t predict the exact imprisonment period. In the real world, once a judge gives a judicial decision to someone who committed a crime, she/he imposes a sentence that's sufficient, like 14 years of imprisonment, instead of a decision involving intervals like 12–14 imprisonment. Therefore, giving a precise amount of imprisonment is important. In (Sun et al., 2019), the objective was to predict charges for multi-defendant cases, but their model cannot distinguish complicated charges. In order to achieve a good value for precision, in contrast to (Virtucio et al., 2019), comparing different algorithms is important. The other thing, that is basic for this study is multi-defendant cases. Even though there are various multi-defendant cases in criminal cases, most of the researchers didn’t include those cases in their studies except (Sun et al., 2019), which is restricted to only charge prediction.

Table 2.2 Summary of related work

S/N	Title	Authors	Objective	Gap	Algorithm
1.	Predicting Decisions of the Philippine Supreme Court Using Natural Language Processing and Machine Learning	(Virtucio et al., 2019)	Predicting affirmed or reversed case	• They have used only one algorithm without any comparison. • They didn't elaborate on how they have dealt with the case, for law cases contain multiple tasks i.e. article, charges, or the law	SVM
2.	Charge Prediction for multi-defendant Cases with Multi-scale Attention.	(Sun et al., 2019)	Predicting charges for multi-defendant cases	•Can't handle complex cases (distinguish confusing charges). •Didn't conclude the average accuracy they got.	Bi-GRU
3.	Predicting Outcomes of Legal Cases based on Legal Factors using Classifiers	(Athar, 2020)	To predict a murder case as Acquittal or Conviction	• Can't handle multi-defendant. •They extracted features manually.	CART
4.	A Deep Learning Method for Judicial Decision Support	(B. Chen et al., 2019)	Predicting accusation, penalty, and legal provision	•Their model can't handle the accusation of one person with multiple crimes or the accusation of multiple defendants with a single crime.	TextCNN
5.	Predicting Brazilian Court Decisions	(Lage-freitas et al., 2022)	Predicting Brazilian court decisions of appellate Court	• Poor dataset preparation	XGBoost
6.	Predicting Criminal Cases of Oromia Supreme Court Using Machine Learning	(Assefa, 2021)	Predicting judgment and penalty.	•They used the penalty level as an independent feature. •The model can't handle multi-defendant cases	SVM

CHAPTER THREE

3 RESEARCH METHODOLOGY

3.1 Introduction

This chapter oversees procedures or techniques that are used in accomplishing proposed study objectives. It consists of four parts. The methods for gathering, preparing, and preprocessing data utilized by different deep learning models are covered in the first part. Deep learning classification models and feature extraction methods are covered in the second part. The model performance evaluation approach used in this study is described in the last part.

3.2 Building Dataset

Among the social sciences, the law may come the closest to a system of formal logic. To put it too simply, a legal decision involves establishing axioms derived from precedents, applying those axioms to the specific circumstances at hand, and coming to a conclusion following those axioms. This logic-based approach is exactly the kind of task that may benefit from the use of artificial intelligence. Even though the law is logic-oriented, which makes it useful for AI study, there aren't many computational resources available, such as pre-trained models and datasets. Numerous researchers cited the complexity of the judicial system as the cause. That is relevant in terms of Ethiopian law. There is no dataset created for a criminal case with multiple defendants. So, it's needed to build a new dataset.

The process of building the dataset for JDP for the multi-defendant case has two main steps:

1. Collecting criminal cases and selecting multiple defendant cases.
2. Filtering and creating a text-based file from an image-based PDF as a single dataset.

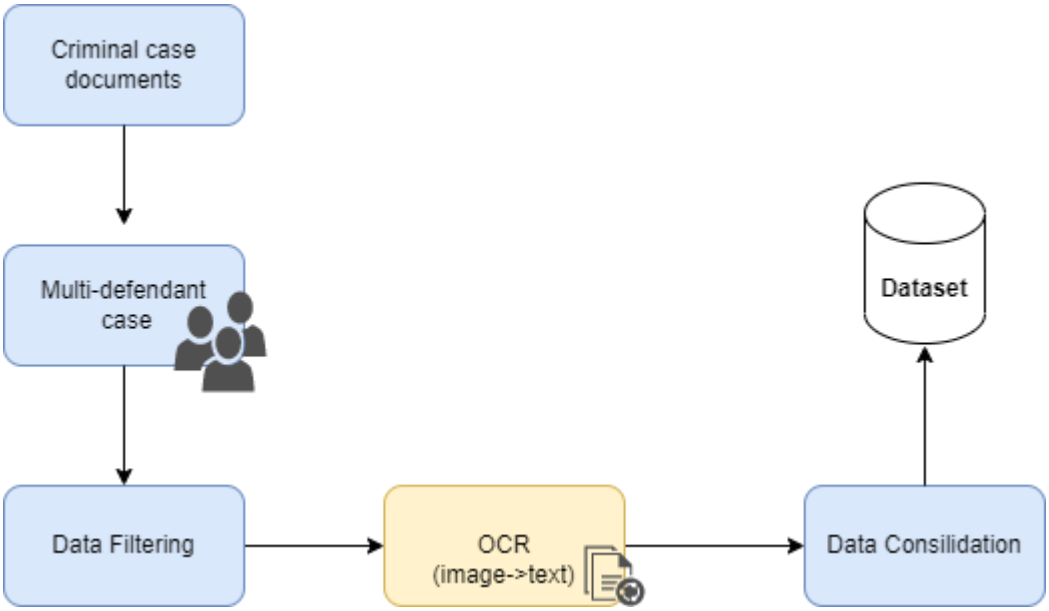

Figure 3.1 A flow of dataset building

The data collected is in the form of scanned documents that cannot be processed by other software. The problem is solved by optical character recognization (OCR) technology, which converts text images into text data that can be analyzed or used for further data processing.

3.2.1 Data Source

Court documents are any documents filed concerning a legal case before the courts. These cases can be either criminal or civil. For this study, criminal cases are selected. A criminal case is a sort of court hearing in which a prosecutor from the federal, state or local government charges someone with a crime[18]. Data for this study was gathered from the Oromia Supreme Court. One of Ethiopia's regional courts is the Oromia Supreme Court. For the community to have access to justice, OSC has its own hierarchy. It has been divided into the Supreme Court, Zone High Courts, and First Instance Courts (woreda). In the Supreme Court, there are two divisions of judges. These are the cassation division and regular division. Regular divisions have the authority to see and examine cases that were appealed to the Supreme Court from other high courts as well as newly filed cases. The regular division is appointed by three to five judges. However, the Cassation Division Inquiry investigates fundamental legal errors in the region.

[18] https://www.law.cornell.edu/wex/criminal_case

There are a group of seven judges, including the president and vice president of the Supreme Court. Their verdict is used as a law for similar cases in the Oromia region (Assefa, 2021).

3.2.2 Data Collection

Data collection is the process of gathering the required information from the applicable sources to provide the desired solution to the existing problem. This step is very important because the quality and quantity of data that we gather will directly determine how good our predictive model can be. For this study, various closed cases will be collected from the Oromia Supreme Court. Every case document that has been gathered is in image format.

For this study, we used cases decided between 2011 and February 2014 E.C. Because OSC began converting hardcopy to softcopy using a scanning device in 2011 E.C. We have collected more than 7,206 criminal cases. These cases include single-defendant cases, multi-defendant cases, and mixed cases. Mixed cases are cases with multiple charges, both for single and multiple defendants. Inconvenient multi-defendant cases are also involved; requests for bail or warranties. They are covered by articles 63, 64, and 28 of the Criminal Procedure Code. Although they are technically criminal cases with multiple defendants, there is no verdict of guilty or not guilty. Therefore, we have only chosen multi-defendant cases that are feasible.

Table 3.1 Distribution of cases first collected from OSC

	Criminal case type	**No of cases**	**Description**
1	Single defendant case	3001	Not-selected
2	Multi-defendant case	2002	Selected
3	Mixed criminal case	1005	Not-selected
4	Inconvenient multi-defendant cases	1200	Not-selected

3.2.3 Data Preparation

After selecting the raw data, data preparation is the next important step in building a dataset. The collected data should pass through some data filtering, cleaning, and consolidation processes to alleviate the data quality issue, which may have an undesirable effect on building classification models. It starts with classifying case documents into single and multi-defendant cases, by looking at the cover page of the document. The cover page of the case document contains information like identification number, name of defendant/s, name of the plaintiff, case

type, etc. So, if there is more than one defendant's name, we will select the case and put it in our folder. The second step is converting image-merged data to a text file with the use of optical character recognition (OCR) since the collected data is in image-combined format. Then, cases that miss some fields (not full case documents), unreadable cases, and duplicated cases will be eliminated.

So, after performing data preparation, 997 cases are eliminated and we have got 1005 cases.

Table 3.2 The amount of filtered and prepared data

	Crime type	**#No of case**
1	Homicide	220
2	Crime against health and person	240
3	Crime against property	221
4	Disturbances of Meetings or Assemblies	168
5	Insulting	100
6	Abduction of a Woman.	56
7	**Total**	**1005 cases**

3.3 Data Preprocessing Techniques

As a part of data preparation, data preprocessing refers to any type of processing done on raw data to get it ready for another data processing technique. Text data is widely accessible and used for problem analysis and solutions. Processing the data is crucial before using it for analysis or prediction. In most cases, we observe that the text data is not totally clean. Text preprocessing is done to get the text data ready for model creation. The prepared data passed through some processes during preparation. It begins by deleting special characters and punctuation. Next, the text is lowered and normalized. Finally, the stop word is eliminated, and the data is tokenized.

3.4 Feature Extraction Method

The field of computer science and machine learning known as "natural language processing" (NLP) focuses on teaching computers to process huge amounts of human (natural) language input[19]. NLP, in its simplest form, is the capacity of computers to understand human language. To create results for the test data, machine learning algorithms learn from a predefined set of features from the training data. However, the primary issue with NLP is that machine learning techniques cannot be used to directly handle raw text. So, to turn text into a matrix (or vector) of features, we need certain feature extraction techniques. The term "feature extraction" refers to the process of extracting and creating feature representations suitable for DL or ML models.

3.4.1 Word2vec

Word2vec, a two-layer neural network, uses the word "vectorization" to parse text. Its input is a text corpus, and its output is a collection of feature vectors that stand in for the words in the input corpus. Word2vec is a program that turns text into numerical data that deep neural networks can interpret. Word2vec's objective and use are to group vectors of related words in vector space[20]. Word characteristics, such as the context of individual words, are distributed numerically represented as vectors via Word2vec. Without the assistance of a human.

Continuous Bag of Words and Skip-Gram are the two primary algorithms for obtaining a Word2vec implementation. To obtain the word vectors, these techniques make use of neural network models. Context is how these models function. This means that the embedding is discovered by looking at words that are close to the target word; if a set of terms is consistently located close to a target word, they will eventually have similar embeddings. The words in a target word's neighborhood are used by the CBOW model to learn to predict that word. The sum of the context vectors is utilized to forecast the target word. The Skip Gram technique, on the other hand, is trained to predict a word based on a neighboring word.

[19] https://www.geeksforgeeks.org/feature-extraction-techniques-nlp/
[20] https://wiki.pathmind.com/word2vec#crazy

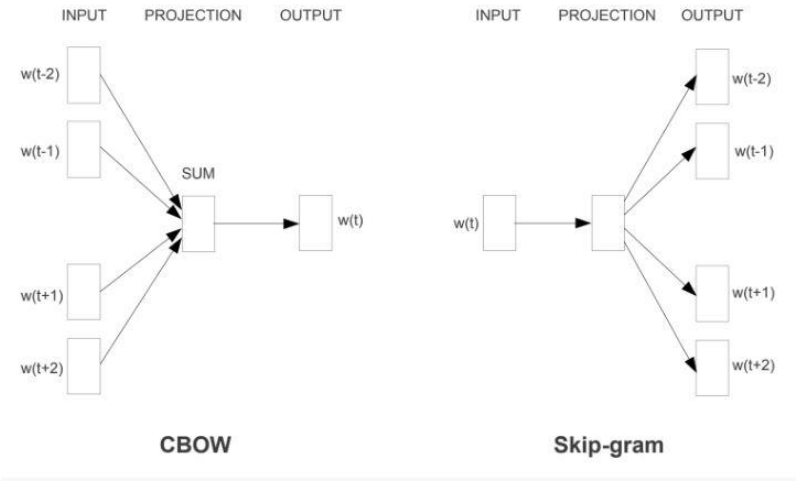

Figure 3.2 CBOW and Skip-gram architecture (Mikolov, Chen, et al., 2013)

3.4.2 FastText

FastText is an open-source library for effective word and text classification. As a result, we can view it as an expansion of traditional text classification techniques. To help a machine learning algorithm understand a text, we typically turn words or sentences into vectors that contain numeric values. FastText does this by dividing words into several sub-words. FastText embeddings create word embeddings by using sub-word information. Learned representations include word sums represented as n-gram vectors and character-gram representations. This enhances the word2vec type models' sub-word information[21].

3.5 Deep Learning Models

Deep learning techniques are proven to be particularly effective at classifying text, producing cutting-edge results on a variety of common academic benchmark tasks. Deep learning is a sort of machine learning that uses numerous algorithms in a continuing series of actions. Similar to how the human brain makes judgments, it processes massive volumes of data simultaneously in a variety of ways. After the dataset is preprocessed and represented in numeric form using word embedding, the data is given to different deep learning classification models.

3.5.1 CNN Based Model

Convolution Neural Networks (ConvNets) are neural networks that convolve (roll over) the original phrase matrix to reduce it into additional low-dimension matrices. They do this by using

[21] https://paperswithcode.com/method/fasttext

a series of filters of various sizes and shapes. ConvNets are used in text categorization to apply distributed and discrete word embedding (Wei et al., 2019). Two primary layers make up a convolutional neural network: a convolution layer for obtaining features from the data, and a pooling layer for reducing the size of the feature map. In the NLP task, the text is represented by a 1D array.

3.5.2 RNN Based Model

A recurrent model-based approach to natural language processing (NLP) in deep learning is the most engaging. The ability of RNN-based models to interpret the text as a collection of words makes them ideal for identifying word dependencies and text categorization. The most common RNN variant, Long Short-Term Memory (LSTM), is designed to better capture long-term situations than other RNN variations. Like any other neural network, LSTM comprises a few layers that aid in pattern recognition and learning for improved performance. The fundamental function of LSTM can be thought of as holding the necessary information while discarding the information that is not necessary or helpful for subsequent prediction [22].

Gated recurrent units (GRU) are the alternative effective option. In this study, bidirectional LSTM and GRU models are used in addition to LSTMs and GRUs.

3.5.3 Hybrid CNN-BiLSTM Model

It is a hybrid classification model that is based on bidirectional long short-term memory (BiLSTM) and convolutional neural network (CNN) models. The proposed CNN-BiLSTM model included a BiLSTM model for classification and a CNN model for feature extraction.

3.6 Model Evaluation

3.6.1 Prediction Model Evaluation

The model development process includes a step called model evaluation. Finding the model that best captures our data is helpful. It also emphasizes how effective the selected model will be in the future. In data science, evaluating model performance using training data is unacceptable. It is simple to produce overly optimistic and overfit models (D. Singh, 2019). Hold-Out and Cross-Validation are two techniques used in data science to assess models. The holdout method is the first and most basic approach, and it is based on testing a model with data that it was not exposed

[22] https://analyticsindiamag.com/a-complete-guide-to-lstm-architecture-and-its-use-in-text-classification/

to during training. The approach has a drawback in that it results in differences in estimating accuracy because of the high variability among the training, validation, and test sets, despite being advantageous in terms of simplicity, speed, and flexibility. The second performance evaluation technique for comparing and assessing models is cross-validation, which divides data into parts, one of which is used to train the model and the rest is used to test or validate it.

K-fold Cross-validation

The only parameter of the technique, k, indicates how many groups a particular data sample should be split into. Consequently, the procedure is usually referred to as "k-fold cross-validation." When a particular number for k is selected, it may be substituted for k in the reference to the model; for example, k=5 would become a 5-fold cross-validation.

The general procedure is as follows:

1. Randomly shuffle the dataset.
2. Dividing the dataset into groups of k
3. For each distinct group:
    1. Consider the group as a holdout or test data set.
    2. Use the rest of the groups' data as a training set.
    3. Use the training set to fit a model, then use the test set to evaluate it.
    4. Discard the model and keep the evaluation result
4. Summarize the model's capability using a sample of the evaluation outcomes[23].

This process has shown an optimistic result for balanced classification tasks, but it fails for imbalance classes. This results from cross-validation, which also randomly divides the data without addressing the class imbalance.

Stratified k-fold cross-validation

The stratified k-fold cross-validation technique is an enhancement of the cross-validation approach used for classification issues. Across all K folds, it maintains the same class ratio as in the dataset. It is typically useful when we have imbalanced data and where the data size is

on the small side[23]. In this study, stratified k-fold cross-validation was performed due to the imbalanced class distribution in the data that was gathered.

3.6.2 Classification Metrics

We can evaluate the effectiveness of machine learning models using classification metrics. Classification is about predicting the class labels depending on input data. There are just two possible output classifications in binary classification. In multiclass classification, there may be more than two possible classes.

To assess the effectiveness of our model, we employed a range of classification measures, including the confusion matrix, accuracy, precision, recall/sensitivity, and f1-score.

The followings are the fundamental terms in performance measure:

- True positive (TP): is the condition when both actual value and predicted value are true.
- True negative (TN): is the condition when both the actual value of the data point and the predicted are False.
- False-positive (FP): These are the cases when the actual value of the data point was False and the predicted is true.
- False-negative (FN): are the cases when the actual value of the data point was true and the predicted is False.

A. Confusion matrix

A table called a confusion matrix is used to describe how well a classification system performs. The output of a classification algorithm is visualized and summarized in a confusion matrix. We may have a better understanding of what our classification model is doing correctly and the kinds of mistakes it is making by calculating a confusion matrix.

For the first binary class the terms are:

- True positives (T.P.): These are cases in which we predicted not guilty (yakkamaa miti) and they are.
- True negatives (T.N.): We predicted guilty (yakkamaa), and the defendant is guilty.

[23] https://www.analyseup.com/python-machine-learning/stratified-kfold.html

- False positives (F.P.): We predicted not guilty (yakkamaa miti), but the defendant is guilty (Also known as a "Type I error.")
- False negatives (F.N.): We predicted guilty (yakkamaa), but the defendant is not guilty. (Also known as a "Type II error.")

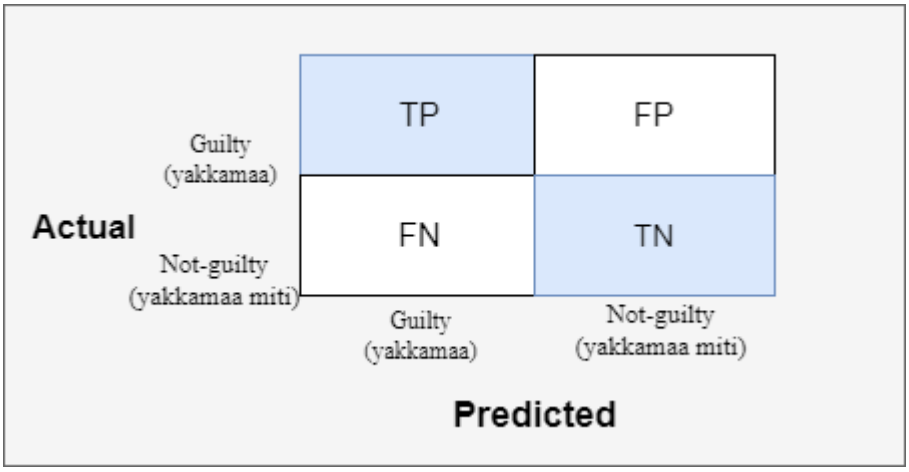

Figure 3.3 Confusion matrix for binary classification

B. Accuracy

It is the most common and popular performance evaluation metric for classification tasks and is defined as the percentage of correct predictions for the test data.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad \textbf{Equation 3-1}$$

C. Precision

Precision is a measure of the likelihood of getting the correct positive class classification, and it is how close a measured value is to each other.

$$Precision = \frac{TP}{TP+FP} \qquad \textbf{Equation 3-2}$$

D. Recall

A recall is the total number of true positives divided by the number of all relevant samples, or it is the ratio of a true positive to a true positive plus a false negative.

$$Recall = \frac{TP}{TP+FN} \qquad \textbf{Equation 3-3}$$

E. F1-score

The harmonic means of the precision and recall values for a classification task is known as the F1-Score. When False Positives and False Negatives are more important than True Positives and True Negatives, the F1 score is a better choice for measuring Accuracy. The following is the F1-Score formula:

$$F1\text{-}score = 2 * (\frac{precision * recall}{precision+recall}) \qquad \textbf{Equation 3-4}$$

3.7 Development Tools

The following tools were utilized to create the proposed model.

3.7.1 Data preparation Tools

MS-office OneNote: Along with applications like Word, Excel, Publisher, and PowerPoint, OneNote is a component of Microsoft Office. OneNote is essentially a limitless digital notebook. It collects user remarks, sketches, screenshots, and audio commentary[24]. OneNote has Optical Character Recognition (OCR), a feature that enables you to copy text from a printout of a file or a photo and paste it into your notes so that you can edit the contents. Since the data gathered from OSC is an image-merged document, we use OCR to transform it into editable text data.

MS-excel: Microsoft Excel is a piece of software produced by Microsoft, that gives users the ability to do simple math, work with graphing tools, make tables, etc[25]. It was used to save the filtered and cleaned data as a single file in a tabular format.

[24] https://www.techwalla.com/articles/onenote-vs-word
[25] https://www.techopedia.com/definition/5430/microsoft-excel

3.7.2 Design Tools

Diagrams.net is a free online UML tool. It is a cross-platform, free and open-source graph drawing program created in HTML5 and JavaScript[26]. All of the diagrams in this study were created using diagrams.net.

3.7.3 Hardware Tools

We use a personal computer with an Intel® Core™ i7-7500U CPU @2.90GHz, 8 Gigabytes of physical memory, 1 Terabyte of hard disk storage capabilities, and 64 bits of Windows 10 Pro operating system to view the results of the experiments.

3.7.4 Implementation Tools

The following resources were used to implement the proposed models.

a. Data analytics and visualization tools

Pandas: Python-based toolkit pandas offers a wide range of utilities, from parsing various file formats to transforming an entire data table into a NumPy matrix array[27].

Matplotlib: Building static, animated, and interactive visualizations are made simple with the help of Python's Matplotlib package. Matplotlib makes simple things simple and complex things possible[28].

NumPy: It's a Python module that offers support for big, multidimensional arrays and matrices, as well as the ability to work on them[29].

Scikit-learn: It is the most useful and robust library for machine learning in Python. It provides several powerful tools for statistical modeling and machine learning via a Python consistency interface, including classification, clustering, regression, and dimensionality[30]. we used scikit-learn to train our model with stratified k-fold cross-validation and the generation of classification reports based on each fold's metrics results.

[26] https://www.diagrams.net/blog/move-diagrams-net
[27] https://pandas.pydata.org
[28] https://matplotlib.org
[29] https://numpy.org
[30] https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm

Jupyter notebook: Using the free, open-source, interactive jupyter application, researchers may combine software code, computational results, explanatory text, and multimedia resources in a single document (Perkel, 2018). While developing the proposed model, we used this application to write, run, and process data using a Python program.

Modeling Tools

Python: Python is a popular computer programming language for creating software and websites, automating processes, and performing data analysis. Python offers several libraries, including TensorFlow and Keras, those help programmers create data analysis and machine learning applications more rapidly and effectively (Coursera, 2022). For this thesis, we used a stable version of python (which is 3.9.3).

TensorFlow: TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. Data is accepted by TensorFlow in the form of tensors, which are multidimensional arrays with greater dimensions (Simplilearn, 2022). TensorFlow version 2.6.1 was used for this study.

Keras: It is a lightweight Python deep learning package that runs on top of TensorFlow. It was designed to facilitate the research and creation of deep learning models as rapidly and straightforwardly as possible (Brownlee, 2019). Keras is very straightforward to understand and use since it provides a high degree of abstraction in Python front and the option of numerous back-ends for computation. For this study, Keras version 2.6.0 was used.

CHAPTER FOUR

4 PROPOSED JUDICIAL DECISION PREDICTION MODEL

4.1 Introduction

The proposed judicial decision prediction for cases involving multiple defendants using a deep learning approach is extensively covered in this chapter. To address the research questions in chapter one and cover the gaps identified in the literature study, we developed the proposed model architecture, which demonstrates the methods and approaches we used. It includes our dataset construction procedure, feature extraction methods, and deep learning algorithms that we used. The methods we used to evaluate the models are also discussed.

4.2 Proposed Model Architecture

The goal of this study is to predict judicial decisions in criminal cases involving multiple defendants. Cases with several defendants differ from those with a single defendant in that a single offense is typically committed by a group of people. Depending on their involvement in the crime and the sort of crime, each defendant in the case will receive their judgment. In Ethiopia, judgment has two parts: judgment on conviction and penalty. Our study includes both parts.

The model architecture is designed to show how the proposed judicial decision prediction model is built. We constructed two models since in criminal case judgment there are two components as discussed above. The first model is the judgment on conviction prediction model. It pleads the defendant as guilty or not guilty. The second model is the penalty model, which predicts the penalty for defendants found guilty depending on the committed crime. The data is gathered from OSC. To prepare the dataset for data preprocessing, as stated in chapter 3, section 3.1, it first goes through a stage of preparation that includes data cleaning, filtering, and consolidation. The model was built on top of the prepared data, which is the dataset.

First, the data preprocessing technique will be applied to the consolidated data. Preprocessing is about removing unwanted characters and stop words, applying normalization, and tokenizing, so that the data can easily be utilized by the next step, which is feature extraction. Since most feature extractions get complicated when uncleaned data is used as an input, the data was

cleaned. To give focus to terms worthy of focus, it was normalized and stop words were eliminated.

Feature extraction is the subject of the second phase. Feature extraction is the process of converting text-based raw data into numerical features while keeping the original data set's content unchanged. Word2vec and FastText are the two feature extraction methods we proposed in this study. They both take into account the word's context while converting the data into vectors, unlike traditional feature extraction techniques. The third step, deep learning algorithms, enters at this point. To recognize patterns and create predictions, extracted vectors were submitted to CNN, CNN-BiLSTM (hybrid), BiGRU, and BiLSTM. To determine which model performed the best, these models were tested using the stratified k-fold cross-validation approach and other classification metrics.

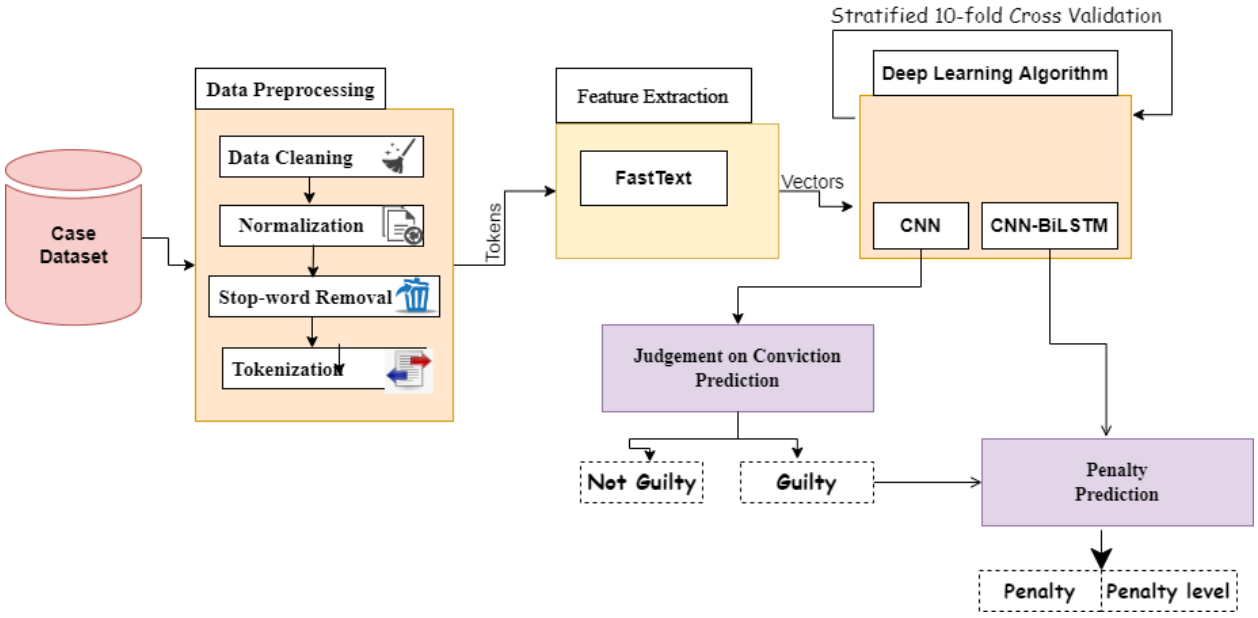

Figure 4.1 Proposed judicial decision prediction model architecture

4.3 Proposed Data Preprocessing

Data preprocessing is a crucial stage in machine learning since the quality of the data and the information that can be gained from it have a direct impact on our model's capacity to learn. The preprocessing techniques were applied to the already prepared dataset file and converted into a common separated value (CSV) format with.csv.

4.3.1 Cleaning the Data

In principle, our preprocessing should be identical to the preprocessing carried out before training the word embedding. The first thing we did was remove punctuation and other special characters from our dataset because the majority of embeddings don't supply vector values for them. Because if the text is in the same case, it is easy for a machine to interpret the words because the lowercase and uppercase are treated differently by the machine.

4.3.2 Normalization

Normalization, technically, refers to the process of converting a word to its original form. As it was previously noted, data is gathered from the Supreme Court. The supreme court reviews cases from several zones. Cases are written in Afaan Oromo, and the terms used in the case can vary depending on the zone. For example, ‘hoggu’ and ‘yommu’ are two terms with the same meaning. By removing variations from a text, normalization can help to lower the number of unique tokens that are present in the text. as well as edit the text by omitting unnecessary details. Therefore, normalization is required to facilitate machine learning.

4.3.3 Stop word removal

Stop words are frequently used words that are eliminated from the text since they add nothing to the analysis. These words have little or no meaning. They can be safely removed without sacrificing the meaning of the sentence. For instance, words like "itti," "kan," ‘"kara," etc. A list of words that are considered stop words for various languages is included in the NLTK library, although Afaan Oromo is not on the list. Most known Afaan Oromo stop-words prepared by former Afaan Oromo researchers are used fully in legal terms. Consequently, we created the Afaan Oromo stop word and included it in the NLTK library.

4.3.4 Tokenization

After some preprocessing techniques, the text was tokenized. Tokenization is the process of breaking down a sentence into words and a paragraph into sentences. In this study, we used the former one, which works by separating the words by spaces or punctuation. For example, the sentence ‘dhagaan adda miidhamaa rukute burrukse’ is tokenized as [‘dhagaa’, ‘adda’, ‘miidhamaa’, ‘rukute’]. The broken pieces are called tokens. Therefore, the final result of data preprocessing was tokens.

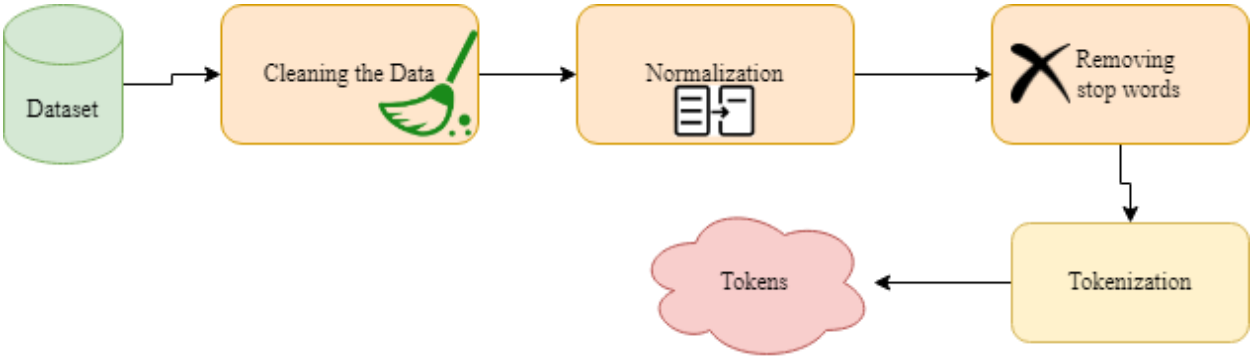

Figure 4.2 Proposed data preprocessing technique

4.4 Feature Extraction methods

Both dependent and independent variables are accepted by deep learning algorithms in numerical form. Therefore, we need to use certain feature extraction techniques to turn the tokens into vectors and prepare the data for our DL model. Two feature extraction techniques are proposed in this study.

4.4.1 Word2vec

A vocabulary containing vectors associated with each word is created by the word2vec neural network, which may then be queried to discover relationships between words or fed into a deep learning network. Word2vec works like an autoencoder in that it encodes each word in a vector, even though it trains words against other words that are near to them in the input corpus. It does so in one of two ways: either by using continuous bag-of-words (CBOW) or continuous skip-gram.

The CBOW model uses the words' context as input and makes an effort to understand it. It then tries to predict contextually accurate words. Consider the sentence, "cuubee qabatee jiruun laphee isaa waraane’ (stabbed him in the chest with the knife he is holding). This sentence is converted by the model into word pairs of the form (context-word, target-word). The user will have to set the window size. If the window for the context word is 2 then the word pairs would look like this: ([cuubee, jiruun], qabatee), ([qabatee, laphee], jiruun), ([jiruun, isaa], laphee), ([laphee, waraane], isaa). With these word pairs, the model tries to predict the target word considering the context words.

The Skip-gram model learns by predicting the surrounding words given a current word. The Continuous Skip-Gram Model, in other words, forecasts words within a specific range before and after the present word in the same phrase (Vijaysinh Lendave, 2021). This is completely the opposite task of CBOW. The input is a center word, and the model predicts the context words. Let's construct a training example with a 'cuubee qabatee jiruun laphee isaa waraane' sentence.

Table 4.1 How the skip-gram function

Source text	**Training samples generated from the source text**
cuubee ***qabatee*** jiruun laphee isaa waraane	(qabatee, cuubee), (qabatee, jiruun), (qabatee, laphee)
cuubee qabatee ***jiruun*** laphee isaa waraane	(jiruun, cuubee), (jiruun, qabatee), (jiruun, laphee), (jiruun, isaa)
cuubee qabatee jiruun ***laphee*** isaa waraane	(laphee, qabatee), (laphee, jiruun), (laphee, isaa), (laphee, waraane)
cuubee qabatee jiruun laphee ***isaa*** waraane	(isaa, jiruun), (isaa, laphee), (isaa, waraane)

Given the sentence: 'cuubee qabatee jiruun laphee isaa waraane' and window size of 2, if the target word is 'laphee', its neighboring words will be (qabatee, jiruun, isaa, waraane). Our input and target word pairs would be (laphee, qabatee) (laphee, jiruun), (laphee, isaa) and (laphee, waraane). In skip-gram, within the sample window, the proximity of the words to the source word plays no role. So 'qabatee', 'jiruun', 'isaa' and 'waraane' will be treated the same while training.

Both models are concerned with discovering relevant information about words within the context of their neighbors, which is determined by a window of nearby words (Brownlee, 2017). This window is a customized parameter of the model.

4.4.2 FastText

Another word embedding technique that relies on the word2vec model is FastText. FastText presents each word as an n-gram of characters rather than directly learning word vectors. Let's consider a word from our dataset, 'yakkamaa'. FastText will take a look at the word and will break it into its n-gram components as [ya, yak, yakk, yakka, yakkam, yakkama, yakkamaa, aa…]. As in the above example, we can see that one of the components of the word "yakkamaa" is the word "yakka". This information aids the model in determining the semantic similarity of the two terms . It may also capture the meaning of suffixes and prefixes for words in the corpus and produce word embeddings for out-of-vocabulary (OOV) terms. Words that are not in the model dictionary have no vector representation in word2vec. This is a significant benefit of this approach.

4.5 Deep Learning algorithm

The goal of our study is to develop a deep learning-based model that predicts judicial decisions in multi-defendant cases. Each defendant in the multi-defendant case is assessed based on their participation in the offense and the offense itself. Each of them committed the same kind of crime, yet their roles indicated that they participated in the crime when it was committed. Legal judgment involves multitasking in the real world. Judgment on conviction (JOC) and penalty are the two duties we consider. The JOC prediction, which determines whether the accused defendant is guilty or not, is a binary classification. Article, role, charge, prosecutor witness, and defendant witness are the features used to predict the JOC. The penalty is the second one. This is a sentence imposed on offenders who were found guilty. It is multi-label, which includes two target classes.

The dataset we utilized for both predictions was the same. To accomplish our goal, we built CNN, LSTM, and GRU-based models, which are the most used algorithms in the legal domain.

4.5.1 Hybrid CNN-BiLSTM Model

Convolutional Neural Network (CNN) layers are used in the CNN-BiLSTM architecture to extract features from input data, and BiLSTMs are used to assist in sequence prediction. By adjusting the kernel sizes and concatenating their outputs, we may use CNN to classify text and find patterns of several closely related words. Therefore, regardless of where they are in the phrase, CNNs may recognize them. Although the Convolutional Neural Networks (CNN)-based

text analysis approach may extract important text features by pooling, it is difficult to get contextual information, but the BiLSTM algorithm can. Therefore, combining CNN with BiLSTM could result in some surprising results.

We used the proposed hybrid convolutional neural network-bidirectional long short-term memory model for both JOC prediction and penalty prediction, in extracting features from the case to make it faster and more efficient. The word embedding applied to features is given as an input to the model. After tokens are converted to vectors, we define the CNN model. We used pooling, convolutional layers with rectified linear activation units (ReLu), and dropout layers. We used the dropout layer and ReLu activation function to handle overfit since neurons can get overfit easily. Then we defined the BiLSTM model that uses the CNN model's output as input parameters. We used a single BiLSTM layer for both penalty prediction and the JOC. Here are also the activation functions and dropout layer. Unlike CNN, we used the Tanh activation function for our LSTM model. The last layer was the dense layer.

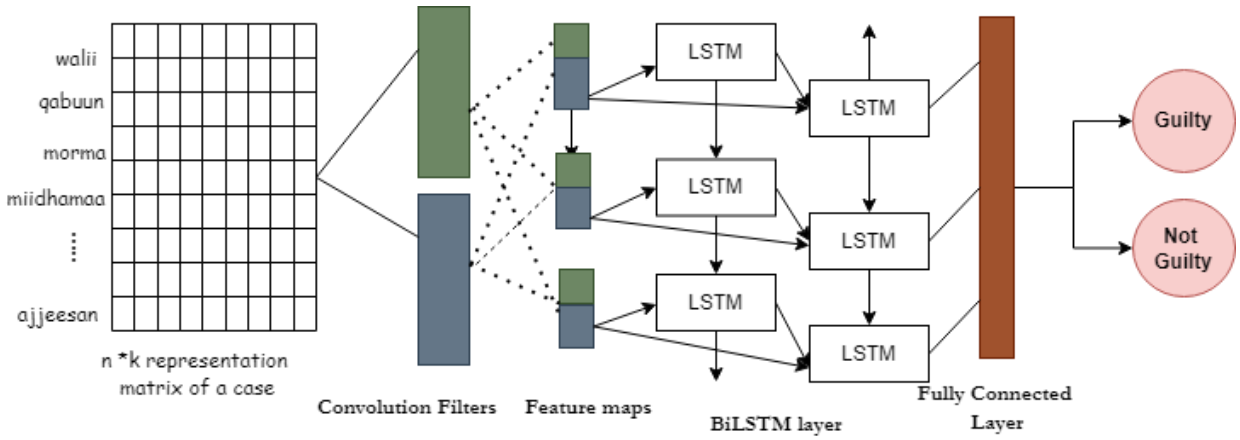

Figure 4.3 Architecture of proposed hybrid CNN-BiLSTM model

4.5.2 BiGRU Model

GRU is a kind of newer version of RNN, widely used for learning sequential data prediction model problems. It is capable of handling vanishing gradients that exist in the traditional RNN model. It contains only 2 gates (Reset and Update Gate), and GRU is faster than LSTM. As with every other thing, they both have some layers which help them to recognize and learn the pattern in the case for better performance. Since GRU works in one direction, we chose BiGRU. Because BiGRU tries to capture information from both sides, left to right and right to left.

This model has an embedding layer, a BiGRU layer, a dropout layer, and a dense layer. As recurrent networks are prone to errors, a dropout layer with a 50% error rate was added after the BiGRU Layer.

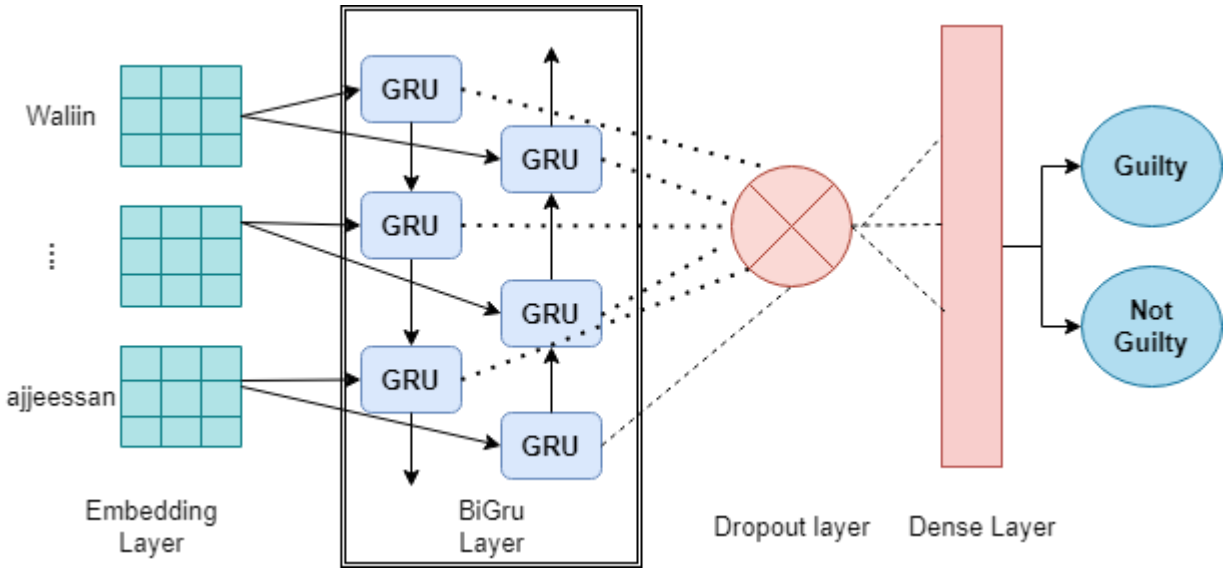

Figure 4.4 BiGru model architecture for JOC prediction

4.5.3 BiLSTM Model

Bidirectional LSTM (BiLSTM) uses information from both sides and allows input to flow in both directions. One extra LSTM layer is added by BiLSTM, which reverses the information flow's direction. Our BiLSTM model's architecture is similar to that of the BiGRU model.

4.5.4 CNN Model

The only way CNN is different from other neural networks is with its convolutional layer. The model that was used has four layers. The words are first converted to their appropriate embedding vectors by the embedding layer, and then a one-dimensional convolutional layer with 32 filters and a kernel size determined by the number of words to be read simultaneously follows. In this layer, predefined filters roll over the case matrix and reduce it to a low-dimensional matrix. The output from the convolutional layer is combined in the third layer, which is a max pooling layer. The final layer is a flattened layer, which converts the output from three dimensions to two dimensions for concatenation. We took advantage of layers that all models share, the dropout layer and the dense layer.

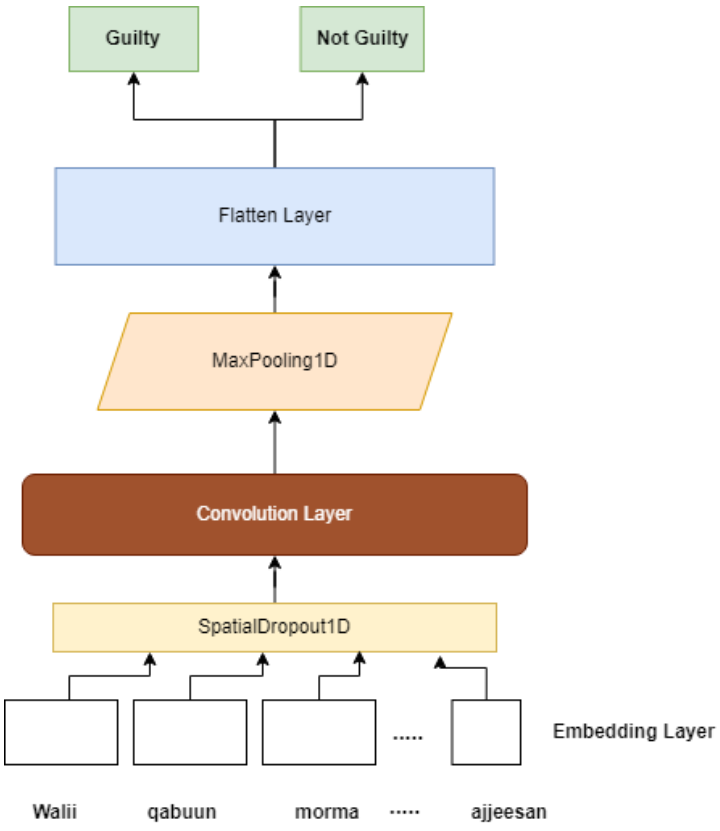

Figure 4.5 CNN model architecture for JOC prediction

4.6 Hyperparameter Tuning

Building the architecture is a prerequisite for neural networks before they can be integrated into a model. The parameters known as "hyperparameters" affect both the architecture of the network and how the network is trained. Finding hyperparameter values that optimize the model's performance is known as hyperparameter tuning. We used a grid search to optimize the number of neurons, learning rate, dropout rate, batch size, and epochs as hyperparameters.

4.7 Model Evaluation

A machine learning model's performance, as well as its strengths and weaknesses, are understood through the process of model evaluation, which employs many evaluation metrics. It’s important to assess the model's efficiency. We used stratified k-fold cv with k=10, and performance will be assessed after training using the confusion matrix, accuracy, precision, recall, and F1 metrics.

CHAPTER FIVE

5 EXPERIMENT AND IMPLEMENTATION

5.1 Introduction

This chapter presents how the methodology covered in chapter three and the proposed solution implemented in chapter four can be used to predict judicial decisions in multi-defendant cases. The criminal code of Ethiopia defines a judgment as having two elements. We created models for each component. To create our models, we combined deep learning techniques with feature extraction based on natural language processing. We evaluate the efficacy of each model after putting these models into practice.

5.2 Working Environment

This study used several development tools and packages, as was mentioned earlier in chapter 3 section 3.8, to put the suggested solution into action. To analyze and visualize data, we used the tools pandas, matplotlib, numpy, scikit-learn, and jupyter notebook. Several more tools and libraries were employed in the implementation of our predictions model, in addition to Tensorflow and Keras as modeling tools.

5.3 Dataset Description

The dataset used for this study is collected from Oromia Supreme Court (OSC) located in Addis Ababa, Ethiopia. Data were gathered by concentrating on criminal court decisions. We created a dataset of judicial decisions from cases with several defendants that were written in Afan Oromo. Since the JOC and punishment are the two components of a judicial judgment, the dataset comprises two target variables and eight independent variables. There is another feature called confession. It’s the defendant's confession that identifies if the defendant admitted the crime or not. We excluded this feature because out of 3101 defendants only 5 defendants in three cases confessed to the crime. Our dataset, which includes 3101 rows and 10 columns with two target classes, was created using 1005 multi-defendant criminal cases. The columns are article, role, charge, prosecution witness, defense witness, JOC, mitigating circumstances, aggravating circumstances, penalty level, and penalty.

Table 5.1 Features description

No	Features	Distinctive feature names	Description
1	Article	Keewwata	A rule outlined in the criminal code.
2	Role	Ga'ee	A part the defendant plays in the crime
3	Charge	Himata	A formal accusation of criminal activity.
4	Prosecution witness	Ragaa Abbaa Alangaa	Every piece of information that could shed light on the crime
5	Defendant witness	Raga himatamaa	The testimony of the defendant disproves the charges brought against him or her.
6	Judgment on conviction (JOC)	Murti balleessumaa	A determination of the defendant's guilt or innocence in the offense.
7	Mitigating circumstances	Yaada adabbii hir'isu	aspects that make a criminal act less serious or culpable.
8	Aggravating circumstances	Yaada adabbii dabalu	factors that make a criminal act more serious or culpable.
9	Penalty level	Gulantaa	a level of punishment imposed on a violator per the article under which they were accused.
10	Penalty	Adabbii	the punishment meted out to one who has broken the law.

Since the penalty level identifies the penalty, the court should modify the level before issuing a sentence. The Ethiopian sentencing rules distinguish between two different sorts of penalty levels. The first one specifies the levels of penalty for sanctions that deny freedom, while the second one lists the levels of penalty for fines. The former contains 39 different penalty levels, and this study only looks at 28 of them. One punishment level may contain several penalties.

We treat them as dependent variables because there is a dependency between them. Using an example from Ethiopian sentencing standards, the table below illustrates the relationship between penalty and penalty level for penalty levels 28-31.

Table 5.2 Sample of penalty level

Penalty level	Penalty
28	9 years – 10 years and 10 months
29	10 years – 12 years
30	11 years – 13 years and 3 months
31	12 years – 14 years and 5 months

5.4 Preprocessing and Label Encoding

5.4.1 Importing Libraries

Python libraries are a collection of pre-built Python modules used for a variety of functions. Some libraries were employed to create a friendly environment and implement the proposed solution, as can be seen in figure 5.1.

```
import re
import sys
import nltk
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from sklearn import preprocessing
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from keras.initializers import Constant
from keras.layers import Input, Dense, Embedding, Flatten
from keras.layers import SpatialDropout1D
from keras.layers.convolutional import Conv1D, MaxPooling1D
from keras.models import Sequential
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Dense, LSTM, GRU, Dropout, Bidirectional, SpatialDropout1D
from tensorflow.keras.utils import to_categorical
from gensim.models import Word2Vec
from tqdm import tqdm
```

Figure 5.1 Sample code for importing libraries

5.4.1 Loading Dataset

The data should be made available to the Python environment we're using before any pre-processing is done on it. We use pandas to import our dataset. The panda's method read the CSV file and loads it as DataFrames, where instances of the data are available via column name.

```
df=pd.read_csv('datasetb.csv', encoding='cp1252')
```

Figure 5.2 Implementation of data loading using pandas

5.4.2 Cleaning Dataset

Following dataset loading, we cleaned the charge ("himata"), and role("ga'ee") columns. Because they include categorical data, other columns do not require cleaning.

```
def clean_non_alphanumeric(text):
    return re.sub('[^a-zA-Z]',' ',text)
df['himata']=df['himata'].apply(clean_non_alphanumeric)
def clean_non_alphanumeric(text):
    return re.sub('[^a-zA-Z]',' ',text)
df['gae']=df['gae'].apply(clean_non_alphanumeric)
```

Figure 5.3 Implementation of data cleaning

5.4.3 Normalization

```
contractions_dict={'rukutu':'rukutuun', 'rukute':'rukutuun','rukuttee': 'rukutuun', 'rukutte':'rukutuun',
                   'rukuutee':'rukutuun','rukutee':'rukutuun','qaamaa':'qaama', 'qamaa': 'qaama',
                   'saman':'saamaan','bakkaa':'bakka','ajjese':'ajjeese', 'ajjeesan': 'ajjese',
                   'ajjeesaniif':'ajjese','yemmu':'yeroo', 'yommuu':'yeroo','yammuu':'yeroo','sia': 'yeroo',
                   'warraane':'waraanee','cuuphee':'cuubee','miidhamtuu':'miidhamtu','miidhamtuun':
                   'miidhamtu','tilmaama':'tilmaamaan','tilmaaman':'tilmaamaan', 'cinaachaa':'cinaacha',
                   'cuube':'cuubee','cubee':'cuubee','cuuphee':'cuubee', 'warane': 'waraanee',
                   'warannee':'waraanee','warane':'waraanee','kufisu':'kuffisee', 'kufisee': 'kuffisee',
                   'kanaa':'kana','bollaa':'boollaa','itii':'itti', 'mukaa':'muka','inii':'inni',
                   'guraa':'gurra','bollaa':'boollaa','dugida':'dugda','altokko':'1', 'tokko':'1' ,
                   'fudhatee':'fudhate','fudhattee':'fudhate',
                   'rukuutuun':'rukutuun' ,'lama':'2','miila':'miilla','miillaa':'miilla'}
contractions_re=re.compile('(%s)' % '|'.join(contractions_dict.keys()))
def expand_contractions(text,contractions_dict=contractions_dict):
    def replace(match):
        return contractions_dict[match.group(0)]
    return contractions_re.sub(replace, text)
df['charge']=df['charge'].apply(lambda x:expand_contractions(x))
```

Figure 5.4 Implementation of word normalization

5.4.4 Removing Stop-words

Filtering away pointless data is one of the main types of pre-processing. Stop words are worthless words (data) that are used in natural language processing. Afaan Oromo is not included in the list of stop words maintained in the NLTK (Natural Language Toolkit) in Python. Afaan Oromo stop words have been gathered and added to the NLTK library for this research.

```
stopwords.words('oromic')
stop=set(stopwords.words('oromic'))
stop_words=[string.replace(" ","") for string in stop]
stop_words = stopwords.words('oromic')
df['himata'] = df['himata'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]
df['gae'] = df['gae'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop_words)]))
```

Figure 5.5 Implementation of removing stop words

5.4.5 Tokenization

To automate the tokenization of our training data, we used the TensorFlow (Keras) tokenizer, class. It enables the vectorization of text corpora by converting each word into a sequence of

integers. First, we create the Tokenizer object and fit it on the concatenated feature with the fit_on_texts method. Next, the method texts_to_sequence creates a sequence of representative numbers based on the word index generated from the fit_on_texts method. Finally, since we need all of our encoded sequences to be the same length, we determined the length of the longest sequence and used that to add extra 0s to the end of all other sequences.

```
X=df['all_features']
#initialize Tokenizer to encode strings into integers
tokenizer = Tokenizer()
# create vocabulary from all words in our dataset for encoding
tokenizer.fit_on_texts(all_features)

maxlen = max([len(x.split()) for x in X.values])
# count number of unique words
vocabulary_size = len(tokenizer.word_index) + 1
word_index = tokenizer.word_index



tokenizer.fit_on_texts(list(X))
X = tokenizer.texts_to_sequences(X)
X = pad_sequences(X, maxlen=maxlen,padding='post')
```

Figure 5.6 Implementation of tokenization

5.4.6 Label Encoding

Our model's target variables are categorical. We must encode categorical features into a representation that is compatible with the models because deep learning algorithms only operate with numeric input. We used the label encoder class to accomplish this.

```
#encoding target features by 0 and 1
le = preprocessing.LabelEncoder()
df['murte'] = le.fit_transform(df['murte'])
df['murte'].unique()
```

Figure 5.7 Implementation of the encoding target variable

5.5 Feature Extraction Implementation

5.5.1 Training Word2vec Embedding

The process of extracting a word embedding from text includes loading the text, structuring it into sentences, and passing those sentences to the constructor of a new *Word2Vec ()* instance. Each sentence must be tokenized specifically. There are a lot of parameters on this constructor, but a couple of important ones we configured are *sentences, sg, min_count, vector_size, and workers.* Sentences are tokenized features, and vector size is the number of dimensions of the embedding, which is 50. We used skip-gram in our research by setting the sg parameter value to 1 since we discovered it to be superior to CBOW. Finally, the word vector model's *save_word2vec_format ()* function was used to save a trained model to a file.

```
train_sentences = list(df['all_features'].progress_apply(str.split).values)

100%|| 2502/2502 [00:00<00:00, 45189.18it/s]
100%|| 2502/2502 [00:00<00:00, 250894.13it/s]

start_time = time.time()

model = Word2Vec(sentences=train_sentences,
                 sg=1,
                 min_count=5,
                 vector_size=50,
                 workers=4)

print(f'Time taken : {(time.time() - start_time) / 60:.2f} mins')
filename='w2v.txt'
model.wv.save_word2vec_format(filename,binary=False)
```

Figure 5.8 Implementation of training word2vec

5.5.2 Training FastText

FastText uses the same training procedure as Word2Vec, and the corpus must be a list of tokens. We created the object for FastText () with the required parameters, and then built the model by using tokenized features. These parameters are vector_size, min_count, workers, alpha, sg, and epoch. The initial learning rate is defined by alpha, while the number of training epochs is specified by epoch. The save_word2vec_format () is also available for FastText models to save the trained model.

```
from gensim.models.fasttext import FastText

model = FastText(vector_size=50, min_count=1, workers=5, sg=0, alpha=0.25, epochs=3)
# build the vocabulary
model.build_vocab(df['all_features'])
model.corpus_count
model.train(df['all_features'], epochs=model.epochs,total_examples=model.corpus_count,
            total_words=model.corpus_total_words)
model.wv.save_word2vec_format('FT', binary=False)
print("The trained FTC Model is saved ")
```

Figure 5.9 Implementation of training FastText

5.5.3 Embedding Matrix Preparation

We must load the word embedding file into memory as a dictionary of words to the embedding array before the trained vectors are assigned to the accessible word indexes in our dataset.

```
embeddings_index={}
f=open(os.path.join('','w2v.txt'), encoding='utf-8')
for line in f:
    values=line.split()
    word=values[0]
    coefs=np.asarray(values[1:])
    embeddings_index[word]=coefs
f.close()
```

Figure 5.10 Implementation for loading word embedding

For each word in the training dataset, an embedding matrix must be created. To do so, we enumerated all unique words in the Tokenizer.word_index and located the embedding weight vector from the loaded word2vec or FastText embedding.

```
EMBEDDING_DIM=50
num_words=len(word_index)+1
embedding_matrix=np.zeros((num_words, EMBEDDING_DIM))

for word, i in word_index.items():
    if i>num_words:
        continue
    embedding_vector=embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

Figure 5.11 Implementation for embedding matrix preparation

5.6 Model Implementation

After we got a pre-processed data and embedding matrix, which the deep learning model requires to be trained, now it is possible to implement all the architectures considered in this study. Before training or fitting the model, we categorized the vectors dataset into independent and dependent variables as X and y. X is a concatenation of all features which is tokenized and padded, while y is the labeled target class.

```
X=df['all_features']
Y = (df['murte'].values)
```

Figure 5.12 Splitting dataset into dependent and independent variables for classification

Ethiopian sentencing guideline offers generic guidance for determining categories for crime with several prescribed offense categories in the guideline. The court must first determine the seriousness of the crime being sentenced. If there are aggravating and mitigating circumstances are found, the court is required to adjust the penalty level before determining a provisional starting penalty. As a result, both the penalty level and the sentence are determined by the court. An offense with the same penalty level can have a different penalty depending on the seriousness of the crime. To deal with the interdependence between penalty and penalty level, we used both as target labels. We picked *MultiLaberBinarizer* to handle the multi-label problem. MultiLabelBinarizer converts labels with multiple classes into a binary vector with all other values set to zero except for the indexes associated with each class, which are denoted with a 1.

As with JOC models, independent characteristics were combined in one location and then held on the X variable.

```
df['label']=np.hstack(df['Adabbii']+' ,'+df['Gulantaa'])
df['label']=df['label'].apply(lambda x:ast.literal_eval(x))

from sklearn.preprocessing import MultiLabelBinarizer
multilab= MultiLabelBinarizer()
X=df['all_features']
y=multilab.fit_transform(df['label'])
```

Figure 5.13 Splitting dataset for penalty classification

5.6.1 Create the Model

The Keras model is a representation of the real neural network model. Simple and user-friendly Sequential API and sophisticated Functional API are the two methods offered by Keras for creating models. The Functional API is used when a model has to share layers or have many inputs or outputs. For modeling more basic neural networks, where each layer only accepts one input and transmits one output, there is the Sequential API, a little less detailed API. For both JOC and penalty predictions, we created a sequential model by using the *Sequential ()* API.

```
model=Sequential()
```

Figure 5.14 Implementation of creating a model

After that, we called the *add ()* method to add layers that only depend on the type of algorithm.

Implementation of CNN: The method used to produce features is convolution. The CNN is then fed these features. The most crucial features are found via the convolution process. We used CNN in one dimension for text classification. The weighted total is filled in in the output after calling each row in the embedding dimension and multiplying each block element by the weight value. This is true across all channels. Each channel's weights are learned by it. In this method, the words are combined into smaller values. When the max-pooling layer is eventually invoked, the network is forced to keep only the highest value of each feature vector, which is typically

the most helpful local feature. Finally, flatten layer was called. The pooled feature map is flattened into a single column and then sent to the fully linked layer.

```
model=Sequential()
embedding_layer = Embedding(num_words,
                            EMBEDDING_DIM,
                            embeddings_initializer=Constant(embedding_matrix),
                            input_length=max_length,
                            trainable=False)
model.add(embedding_layer)
model.add(SpatialDropout1D(0.2))

model.add(Conv1D(64, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(2, activation='sigmoid'))
```

Figure 5.15 Implementation of the CNN model

Implementation of CNN-LSTM: Combining these two models aims to produce a model that benefits from the advantages of CNN and BiLSTM, capturing the CNN-extracted features and using them as an LSTM input. As a result, we create a model that achieves this goal and uses the vectors created in the word embedding section as the input for convolutional neural networks. After each filter, a layer of max pooling is applied to update and reduce the size of the data. Then, the results of all max-pooling layers are concatenated to build the BiLSTM input, which applies a BiLSTM layer to filter the information. Finally, to assign classes to cases and produce the appropriate output, we use the softmax function as an activation function for the penalty and sigmoid for JOC.

```
model=Sequential()
embedding_layer = Embedding(num_words,
                            EMBEDDING_DIM,
                            embeddings_initializer=Constant(embedding_matrix),
                            input_length=max_length,
                            trainable=False)
model.add(embedding_layer)
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Dropout(0.8))
model.add(Bidirectional(LSTM(100,return_sequences=True)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(2, activation='sigmoid'))
```

Figure 5.16 Implementation of the CNN-BiLSTM model

Implementation of BiGru and BiLSTM: Both BiGru and Bidirectional Long Short-Term Memory (BiLSTM) is a type of recurrent neural network Since they use two hidden layers, they process data in two ways. For the JOC model, we used three layers: embedding, BiGRU/BiLSTM, and dense layer. Four hidden layers are utilized in the penalty model by adding one additional hidden layer.

```
model=Sequential()
embedding_layer = Embedding(num_words,
                            EMBEDDING_DIM,
                            embeddings_initializer=Constant(embedding_matrix),
                            input_length=max_length,
                            trainable=False)
model.add(embedding_layer)

model.add(Bidirectional(LSTM(50,return_sequences=True)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Figure 5.17 Implementation of the BiLSTM model

5.6.2 Compile the Model

To build the model, the Keras model provides a method called *compile ()*. We included three arguments in the compile method: loss, optimizer, and metrics. The loss function is used to detect errors or deviations in the learning process. A binary_crossentropy loss function is used for both multi-label and binary classification. As previously stated, the two target classes we employed in the multi-label are multi-classes and multi-class use categorical_crossentropy. However, in multi-label, we structure the objective function as a binary classifier, with each neuron(y.shape[1]) in the output layer responsible for one vs all class categorization.

Optimization is an important technique that compares the prediction and the loss function to optimize the input weights. Keras has several optimizers, as modules, one of which was Adam. The Adam method is easier to build, has a shorter run time, consumes less memory, and requires less tuning than any other optimization technique. Model performance is assessed using metrics.

```
history=model.fit(X[train], y[train], epochs=30, batch_size=50, verbose=0)
```

Figure 5.18 Implementation of model compiling

5.6.3 Model Training

Fit is used in the training of models. This fit function's objective is to assess the model's performance during training. The next step after training the model is model evaluation. When building a model, evaluation is a procedure that determines whether the model is the best fit for the given problem and associated data.

```
    #train the model
history=model.fit(X[train], y[train], epochs=50, batch_size=30, verbose=0)
    # evaluate the model
scores = model.evaluate(X[test], y[test], verbose=0)
```

Figure 5.19 Implementation of model training

5.7 Model Evaluation

We evaluated our models by using stratified 10-fold cross-validation. From the 10 folds, 9 folds are used for training and 1-fold for testing iteratively. An improvement on the cross-validation method used for classification issues is the stratified k-fold cross-validation. It aims to address the issue of imbalanced target classes.

```
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)
cvscores = []
```

Figure 5.20 Implementation of Stratified cross-validation

Additionally, we evaluated our model's performance using evaluation metrics such as the confusion matrix, precision, recall, and F1-score.

CHAPTER SIX

6 EVALUATION, RESULTS, AND DISCUSSIONS

6.1 Introduction

The experimental findings of the proposed methodology for applying deep learning to predict judicial decisions in multi-defendant situations are described in this chapter. The usage dataset's summary and class distribution are covered first. The confusion matrix and classification report are then given depending on the validation result. A comparison of the four models is offered in the third section. Finally, a thorough comparison of this study with the prior research is offered to ensure that it addresses the research topic and covers the gaps found in the literature review.

6.2 Dataset Class Distributions Result

After preprocessing, the dataset utilized for this study has 3002 rows and 10 columns, comprising three target classes. Judgment on Conviction, which is a binary classification, is the first class. All of the data, 3101 rows, was used for the JOC class. Using the method outlined in section 4.5, it was labeled as 2248 guilty and 853, not guilty. Consequently, there is a class imbalance in the dataset, since instances are not distributed equally among the target classes. When it comes to the machine learning technique, training the model with dataset distributions that have a class imbalance causes the model to be biased toward the majority class, which results in prediction issues for the minority class. We used stratified cross-validation to handle class imbalance. All classification metrics were used to assess the model's performance.

For both JOC and penalty prediction, we utilized the same dataset. However, as only defendants who enter guilty pleas are sentenced, we deleted instances of not guilty in the second one. As a result, the penalty dataset contains 2232 rows. There are 2248 cases, but only 2232 of these are exactly predicted by the best model, CNN. The second class is the multi-label penalty class. In light of this, the two left target classes, penalty level, and penalty, have 38 and 42 classes, respectively.

There is a class imbalance in both target classes. Our model ended up having low accuracy as a result. Therefore, we have chosen to employ a method that neither violates the court process nor forces our model to overfit. It involves making certain adjustments to classes with low penalty

counts. Classes with few occurrences are by default changed from the penalty since it relies on the penalty level. The first step in doing this is to completely rewrite the classes' mitigating circumstances and aggravating circumstance attributes. We allow the class to fall into either the previous or following class by doing this. The classes that go too far when we rearrange those characteristics are then deleted. For example, a category with many or few mitigating factors. As the result, the degree of class inequality has reduced, and there are 28 classes of penalty levels and 29 classes of penalties.

6.3 Model Evaluation Result

In this experiment, we evaluate the performance of both the two models in terms of the JOC, and penalty. Cross-validation has been used to prevent overfitting and assess and evaluate the performance of models on unseen data. Our decision to use stratified cross-validation rather than other cross-validation types was based on the fact that both models deal with imbalanced classification. Three K's (2, 5, and 10) were compared to one another. All of our evaluations are carried out on top of 10-fold stratified cross-validation because it outperforms the most.

6.3.1 Judgment on Conviction Models Evaluation Results

The judgment on conviction model is a binary class, that identifies each defendant in a multi-defendant case as guilty and not guilty. We used six features including the target class of the built dataset which has 3101 instances. To evaluate the performance of this model variety of classification metrics are used in a coming section.

Confusion Matrix for CNN

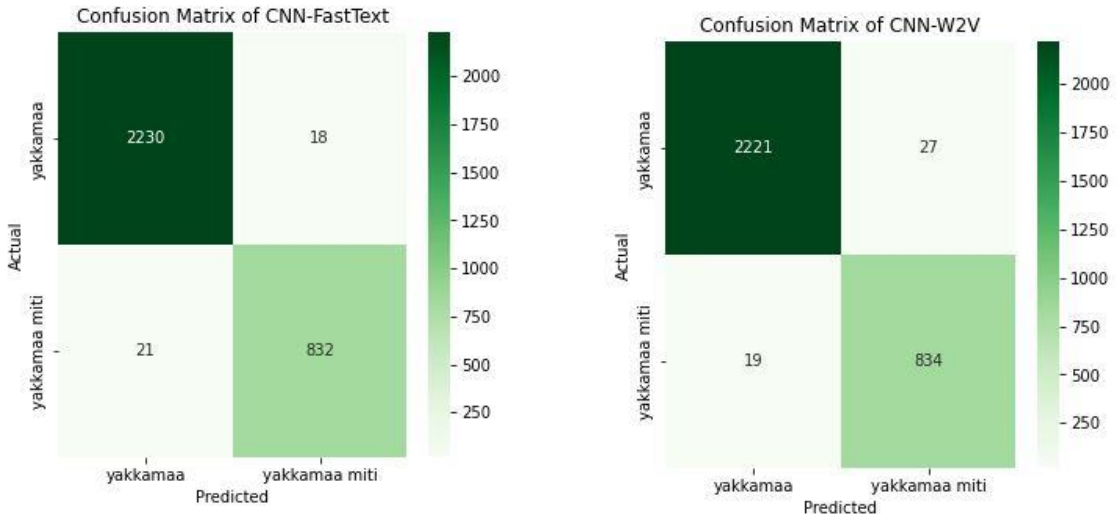

Figure 6.1 Confusion matrix for CNN with two feature extraction

We employed a CNN model with two feature extractions, as indicated in the above figure. With 98.74% accuracy, the initial model created with FastText identified 2230 instances as "yakkamaa" out of 2248 instances, and 832 as "yakkamaa miti" out of 853 instances. Out of a total of 3101 instances, the second model correctly identified 2221 as "yakkamaa" and 834 as "yakkamaa miti" with an accuracy of 96.19 %.

Confusion Matrix for BiLSTM

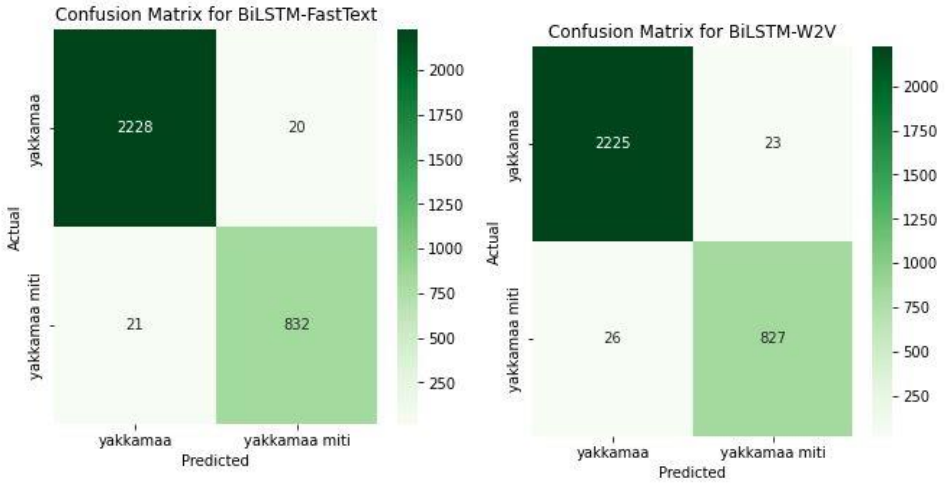

Figure 6.2 Confusion matrix of BiLSTM with fastText and word2vec

Next is the BiLSTM model. Out of 2248 "yakkamaa" cases, our BiLSTM across with Word2vec feature extraction model accurately identified 2225 cases, and 827 of the 853 cases were classified as "yakkamaa miti." The accuracy is 95.5%. The BiLSTM model with FastText identified 832 cases as "yakkamaa miti" and 2228 cases as "yakkamaa with 98.4% accuracy.

Confusion Matrix for Hybrid CNN and LSTM Model

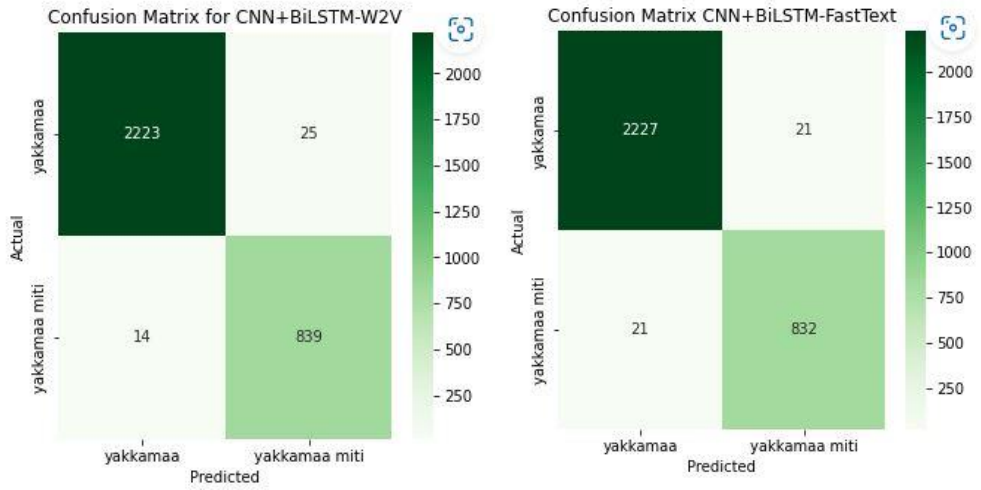

Figure 6.3 Confusion matrix for hybrid CNN and BiLSTM model

Both BiLSTM and CNN performed admirably on our method and dataset, as shown in the aforementioned figures. The third model is a combination of these two. The hybrid model and FastText correctly identified 2223 of the cases as True Positive, 25 as False Positive, 14 as False Negative, and the remaining 839 as True Negative. Among 3101 instances with an accuracy of 96.5%. Additionally, trained with word2vec, it achieved an accuracy of 94.4 %.

Confusion Matrix for BiGRU Model

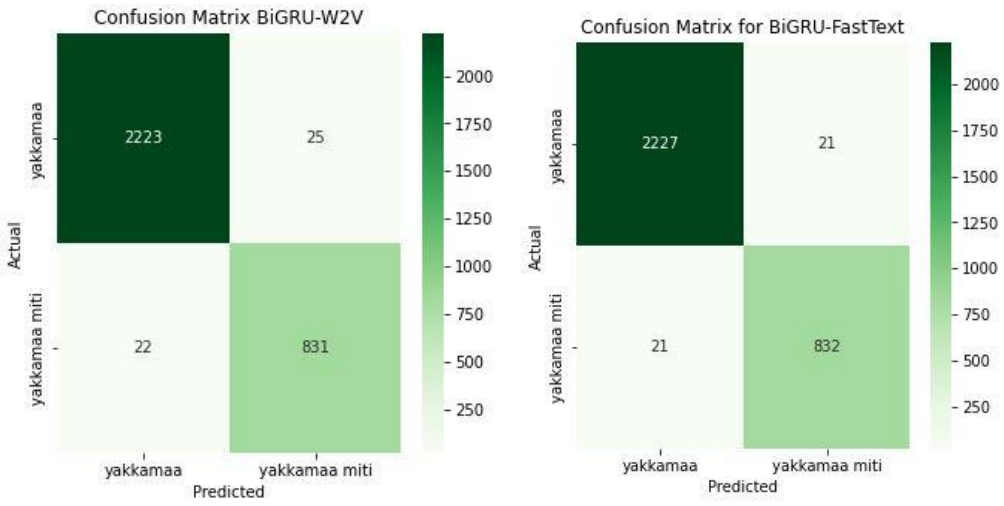

Figure 6.4 Confusion matrix for BiGRU model with word2vec and FastText

The BiGRU model is also used with word2vec and FastText. The first achieves an accuracy of 94.4 % and 47 cases are misclassified. Only 42 cases out of 3101 are incorrectly classified in the second, which receives an accuracy of 98.3%.

Table 6.1 Summary for classification performance of the JOC model

Feature Extraction	**Models**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 score (%)**
Word2vec	CNN	96.19	98.4	98.4	98.4
	BiGRU	94.48	95	95	**95**
	BiLSTM	95.52	97.5	97.5	97.5
	CNN-BiLSTM	94.42	97.7	97.7	97.7
FastText	CNN	**98.74**	99	99	**99**
	BiGRU	98.32	98.6	98.6	98.6
	BiLSTM	98.41	98.1	98.1	98.1
	CNN-BiLSTM	96.52	94.8	94.8	94.8

Table 6.1 shown above illustrates the classification performance of models with FastText and woed2vec on our dataset. CNN model with FastText outperforms the other models in all used metrics.

6.3.1.1 Tuning Hyperparameters

The reason behind hyperparameter optimization is that neural networks have a lot of parameters and are a little bit difficult to configure in such a way that they can perform well on a given problem. We used the grid search optimization technique. Even though it's a bit time-consuming due to the need to run a lot of trials, it is the best method for hyperparameter optimization. We tuned parameters for every model starting from several neurons. The number of neurons in a layer is an important parameter to tune. Generally, the number of neurons in a layer controls the representational capacity of the network, at least at that point in the topology.

```
Best: 0.978000 using {'model__neurons': 100}
0.976800 (0.014945) with: {'model__neurons': 10}
0.976000 (0.017251) with: {'model__neurons': 15}
0.975600 (0.010500) with: {'model__neurons': 20}
0.973200 (0.013151) with: {'model__neurons': 25}
0.967600 (0.027565) with: {'model__neurons': 30}
0.975200 (0.011143) with: {'model__neurons': 32}
0.975200 (0.010553) with: {'model__neurons': 50}
0.974400 (0.013764) with: {'model__neurons': 64}
0.978000 (0.016420) with: {'model__neurons': 100}
```

Figure 6.5 Hyperparameter tuning number of neurons for the CNN model

We also tuned batch size and epochs. They define how many patterns to read at a time and the number of times that the training dataset is shown to the network. To optimize dropout and learning rate, the same approach was used.

```
Best: 0.985600 using {'batch_size': 10, 'epochs': 10}
0.985600 (0.003921) with: {'batch_size': 10, 'epochs': 10}
0.980398 (0.002838) with: {'batch_size': 10, 'epochs': 50}
0.981199 (0.002834) with: {'batch_size': 10, 'epochs': 100}
0.983600 (0.002043) with: {'batch_size': 20, 'epochs': 10}
0.979998 (0.004427) with: {'batch_size': 20, 'epochs': 50}
0.980798 (0.005192) with: {'batch_size': 20, 'epochs': 100}
0.985200 (0.003443) with: {'batch_size': 30, 'epochs': 10}
0.983199 (0.003400) with: {'batch_size': 30, 'epochs': 50}
0.975998 (0.006128) with: {'batch_size': 30, 'epochs': 100}
0.767587 (0.043645) with: {'batch_size': 60, 'epochs': 10}
0.983999 (0.002268) with: {'batch_size': 60, 'epochs': 50}
0.981199 (0.003001) with: {'batch_size': 60, 'epochs': 100}
```

Figure 6.6 Grid search tuning number of epochs and batches for the CNN model

6.3.2 Penalty Models Evaluation Results

The penalty model is multi-label. It has seven features, including two target classes. Each target class has multiple classes; the penalty has 28 classes and the penalty level has 29 classes. This model is fed the output of the first JOC model. As demonstrated in table 6.1, the CNN model outperforms by correctly identifying 2232 of 2248 occurrences as guilty. Only these 2232 are used to predict penalties. Four deep learning models, as well as two feature extractions, are employed, as in JOC. The results of each model are shown in the table below.

Table 6.2 Summary for classification performance of the penalty models

Feature Extraction	**Models**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Word2vec	CNN	58.33	62.1	63.4	62.6
	BiGRU	50.12	51	51	51
	BiLSTM	66.03	67	67	**67**
	CNN-BiLSTM	**66.21**	67	66	**67**
FastText	CNN	70.21	71	72	71
	BiGRU	67.23	64.3	64.5	64.4
	BiLSTM	71.95	72	72	72
	CNN-BiLSTM	**73.29**	74	73	**74**

We use the fastest model CNN and run both before and after eliminating stop words to observe the effects on our data. When stop words were removed, the CNN model's accuracy fell by 1.34%. Therefore, we found that removing stop words was beneficial in terms of performance enhancement.

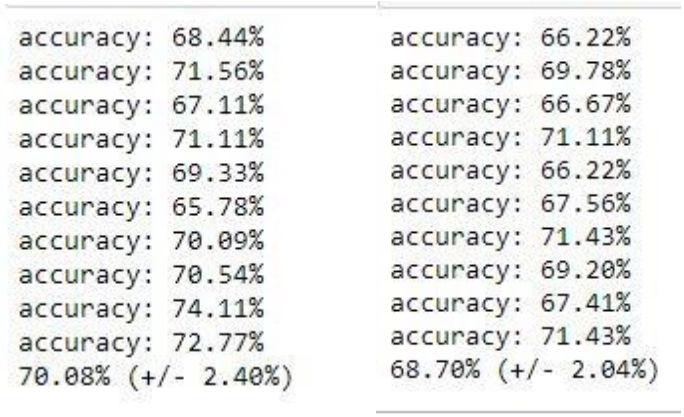

Figure 6.7 10-fold validation result for CNN model after and before removing stop words

6.4 Discussions

The objective of this study is to predict judicial decisions in multi-defendant cases by using deep learning approaches. This judicial decision has two parts, judgment on conviction and penalty. For the experimentation, we prepared a dataset with 1005 multi-defendant cases collected from OSC. Our models mimic the actual Ethiopian judicial decision-making process when predicting these outcomes. A penalty is imposed on defendants who are found guilty after the defendant is classified as either guilty or not guilty in the judgment of conviction. In contrast to JOC, penalty prediction uses multiple labels since the amount of the penalty is completely determined by the degree of the penalty, which in turn is determined by the defendant's charge as well as any mitigating or aggravating factors.

Although the (Assefa, 2021) study didn't take into account cases with many defendants, it, however, addressed both parts of judicial decisions. However, the predicted penalty did not accurately reflect Ethiopian court procedures. The offender is subject to punishment, along with the degree of the penalty. As was stated earlier in this section, these features are very dependent on one another. Consequently, it is against court procedure to make the penalty level an independent variable. Instead of forecasting, the precise amount of the penalty, the (Zhong et al., 2018) model predicted the penalty in intervals, which corresponds to the level of the penalty

in the Ethiopian criminal code. Numerous studies in the legal field, such as those by (Virtucio et al., 2019), (Athar, 2020b), and (Lage-freitas et al., 2022) are restricted to binary classification. Other researchers have studied multiple tasks in court decisions, such as (Zhong et al., 2018) and (B. Chen et al., 2019; but like (Assefa, 2021), they did not consider many defendants.

The study looked at a variety of deep learning neural networks and natural language processing techniques to predict the outcome of multi-defendant cases. The study attempted to deal with multi-defendant cases by distinguishing the roles of each defendant in the offense. We implemented four different types of deep learning algorithms for the judgment on conviction (JOC), which is the first portion of the judicial decision. These algorithms include BiLSTM, BiGRU, a hybrid of BiLSTM, and CNN. The CNN with FastText achieved the highest accuracy of 98.74% and 99% f1-score. The penalty model is the second prediction model. As we discussed in various sections of this study, the penalty is divided into two parts. The fundamental thing we've done before developing a deep learning model for penalty prediction is to make these parts of the penalty target variables. As a result, the penalty is a multi-label classification. After implementing the four DL algorithms used for the JOC, along with FastText and word2vec, the hybrid of CNN and BiLSTM outperforms other models with an accuracy of 73.29% and a 74% f1-score. When compared to the JOC model, the performance of the penalty prediction is inferior. The amount of training data may indeed have a significant impact in determining some of the differences, in addition to the number of classes. From the overall experiment results, as shown in Tables 6.1 and 6.2, we can conclude that FastText is the best feature extraction technique that works well with different DL algorithms for this study.

A grid search was carried out to identify the most suitable parameters for these models. Along with classification metrics including accuracy, precision, recall, f1-score, and confusion matrix, stratified 10-fold cross-validation was employed to evaluate their results. This study is the first one we are aware of that took into account many defendant cases when predicting a court's decision.

6.5 Contribution

We used different NLP and DL techniques to answer research questions in section 1.4 and address problems in section 1.3 and contributed. The main contributions are:

- Preparing a multi-defendant dataset with data from the Oromia Supreme Court. These cases are in image format because they were scanned documents. We used OCR to convert the flattened image file to a text file after filtering multi-defendant cases from collected criminal cases to make it suitable for dataset preparation.
- Predicting the penalty according to Ethiopian Court procedures. In Ethiopia, the court must first determine the gravity of the crime. The court is then required to adjust the penalty level based on aggravating and mitigating circumstances before determining the penalty. As a result, in addition to the penalty, the penalty level is considered part of the sentence. We made them target variables to handle the dependency between penalty and penalty level. So, the penalty has multiple labels.
- Take into account criminal cases with several defendants. Numerous researchers mentioned it in their future work due to the complexity that exists in multi-defendant cases. By considering the defendant's involvement and the nature of the offense, we have attempted to resolve the difficulty. Articles 32/3 and 41 of the Ethiopian Criminal Code provide that the defendant's role plays a role in the decision-making process.
- Predicting judicial decisions using deep learning algorithms along with FastText and word2vec feature extractions.

CHAPTER SEVEN

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

Judicial decision-making is one of several responsibilities that fall within the umbrella of the legal field. The judgment of conviction and the penalty are the two aspects of the judicial decision in criminal proceedings. As a result, this study provides two prediction models implemented on top of a newly collected dataset with 1005 multi-defendant cases. The dataset comprises 3101 instances with ten variables, three of which are target variables. The labeled dataset was reviewed by three legal experts. We utilized FastText and word2vec feature extraction to make the cases we gathered intelligible to the deep learning algorithm. For both JOC and penalty prediction, we compared four deep learning models for classification and tested each one using a 10-fold stratified cross-validation technique.

To maximize the performance of the CNN, BiLSTM, BiGRU, and CNN-BiLSTM models, the optimal hyperparameters were chosen using the grid search CV hyperparameter tuning technique. In this study, the best performing model for judgment on conviction prediction was determined to be a CNN using FastText feature extraction. Under stratified 10-fold cross-validation, it received scores of 98.74 % accuracy and 99 % f1-score. The Hybrid CNN and BiLSTM performed well for the second model. Its f1-score is 74%, and its accuracy is 73.29 %.

7.2 Future Work

Numerous other possibilities are available for future research. The first is improving the accuracy of penalty prediction after gathering more data so that it can truly encompass all levels of imprisonment punishment. Fines may be imposed either in addition to other sentences or as the sole punishment in Ethiopia. The level of fine penalties ranges from 1 to 23. We included a variety of offenses in our experiment, but we didn't include offenses for which the fine was more than 500 birr. We included up to 500 birrs' sole fine penalties because they can fall under the imprisonment penalty level. The inclusion of fine penalties will therefore be the second option. The study could also be extended to predict court outcomes in multiple-charge cases.

* * *

Special Acknowledgement: This research was funded by Adama Science and Technology University with a grant number **ASTU/SM-R/556/22**

REFERENCES

Alem, A. (2014). *Application of Case-Based Reasoning in Legal Case Management: An Experiment with Ethiopian College of Natural Sciences*. Addis Ababa University.

Anala, S. (2020). *fastText for Text Classification. I explore a fastText classifier for… | by Shraddha Anala | Towards Data Science*. Towardsdatascience. https://towardsdatascience.com/fasttext-for-text-classification-a4b38cbff27c

Archak, S. (2018). *Word Embeddings : Word2Vec and Latent Semantic Analysis*. https://www.linkedin.com/pulse/word-embeddings-word2vec-latent-semantic-analysis-shrikar-archak

Assefa, G. (2021). *Predicting Criminal Cases of Oromia Supreme Court Using Machine Learning* (Issue October). Adama Science and Technology University.

Athar, R. (2020a). Predicting Outcomes of Legal Cases based on Legal Factors using Classifiers. *Procedia Computer Science*, *167*(2019), 2393–2402. https://doi.org/10.1016/j.procs.2020.03.292

Athar, R. (2020b). ScienceDirect Predicting Outcomes of Legal Cases based on Legal Factors using Predicting Outcomes of Legal Cases based on Legal Factors using Classifiers Classifiers. *Procedia Computer Science*, *167*(2019), 2393–2402. https://doi.org/10.1016/j.procs.2020.03.292

Brownlee, J. (2017). *What Are Word Embeddings for Text?* https://machinelearningmastery.com/what-are-word-embeddings/

Brownlee, J. (2019). *Introduction to Python Deep Learning with Keras*. https://machinelearningmastery.com/introduction-python-deep-learning-library-keras/

Bulcha, M. (1997). The politics of linguistic homogenization in Ethiopia and the conflict over the status of afaan Oromoo. *African Affairs*, *96*(384), 325–352. https://doi.org/10.1093/oxfordjournals.afraf.a007852

Chen, B., Li, Y., Zhang, S., Lian, H., & He, T. (2019). A Deep Learning Method for Judicial Decision Support. *Proceedings - Companion of the 19th IEEE International Conference*

on Software Quality, Reliability and Security, QRS-C 2019, 145–149. https://doi.org/10.1109/QRS-C.2019.00040

Chen, C. W., Tseng, S. P., Kuan, T. W., & Wang, J. F. (2020). Outpatient text classification using attention-based bidirectional LSTM for robot-assisted servicing in hospital. *Information (Switzerland)*, *11*(2). https://doi.org/10.3390/info11020106

Commentary, G., & Annals, A. (2002). The Judicial System. In *World Development Report*.

Coursera. (2022). *What Is Python Used For? A Beginner's Guide | Coursera*. https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python

Dobilas, S. (2022). *LSTM Recurrent Neural Networks — How to Teach a Network to Remember the Past | by Saul Dobilas | Towards Data Science*. Towardsdatascience. https://towardsdatascience.com/lstm-recurrent-neural-networks-how-to-teach-a-network-to-remember-the-past-55e54c2ff22e

Fangyug. (2020). *NLP: Word Embedding Techniques for Text Analysis | by Fangyug | SFU Professional Computer Science | Medium*. https://medium.com/sfu-cspmp/nlp-word-embedding-techniques-for-text-analysis-ec4e91bb886f

GoE. (2005). The Revised Criminal Code of the Federal Democratic Republic of Ethiopia. *Negarita Gazeta*, *414*, 182–183.

Imperial Ethiopian Government. (1969). *Criminal Procedure Code Of Ethiopia PROCLAMATION No.185 OF 1961*. *185*, 1–97.

Karissa Key. (2020). *Co-Defendant Cases - Pumphrey Law*. https://www.pumphreylawfirm.com/criminal-defense/co-defendant-cases/

Kort, F. (1957). Predicting Supreme Court Decisions Mathematically: A Quantitative Analysis of the "Right to Counsel" Cases. *American Political Science Review*, *51*(1), 1–12. https://doi.org/10.2307/1951767

Lage-freitas, A., Allende-cid, H., & Santana, O. (2022). *Predicting Brazilian Court Decisions*. *2015*, 1–23. https://doi.org/10.7717/peerj-cs.904

Lucarelli, G., & Borrotti, M. (2019). *Approach for Automated Cryptocurrency Trading* (Vol. 2, Issue 691154). Springer International Publishing. https://doi.org/10.1007/978-3-030-19823-7

Luo, B., Feng, Y., Xu, J., Zhang, X., & Zhao, D. (2017). *Learning to Predict Charges for Criminal Cases with Legal Basis*.

Marcus, P. (2003). *Re-evaluating Large Multiple-Defendant Criminal Prosecutions*. *11*(1).

Mesfin, E. (2009). *Application of Multilayer Feed Forward Artificial Neural Network Perceptron in Prediction of Court Case's Time Span: The Case of Federal Supreme Courts*. Addis Ababa University.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations ofwords and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 1–9.

Nagesh Singh Chauhan. (2011). *Attention mechanism in Deep Learning, Explained - KDnuggets*. https://www.kdnuggets.com/2021/01/attention-mechanism-deep-learning-explained.html

NSS. (2020). *FastText | FastText Text Classification & Word Representation*. https://www.analyticsvidhya.com/blog/2017/07/word-representations-text-classification-using-fasttext-nlp-facebook/

Perkel, J. M. (2018). Why Jupyter is data scientists' computational notebook of choice. *Nature*, *563*(7729), 145–146. https://doi.org/10.1038/D41586-018-07196-1

Pizzinini, H. P. (2021). *AI Based Applications For The Legal Field - Analyzed*. https://speedlegal.io/post/ai-based-applications-for-the-legal-field-analyzed

Prodip Hore, S. C. (2019). *Attention Mechanism In Deep Learning | Attention Model Keras*. AnalyticsVidhya. https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/

Saeed, M. (2021). *An Introduction To Recurrent Neural Networks And The Math That Powers Them*. https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-

and-the-math-that-powers-them/

Scott, W. (2019). *TF-IDF from scratch in python on a real-world dataset. | by William Scott | Towards Data Science*. Towardsdatascience. https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

Shraddha Shekhar. (2021). *LSTM for Text Classification | Beginners Guide to Text Classification*. https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/

Siddharth, M. (2021). *Feature Extraction and Embeddings in Natural Language Processing*. AnalyticsVidhya. https://www.analyticsvidhya.com/blog/2021/07/feature-extraction-and-embeddings-in-nlp-a-beginners-guide-to-understand-natural-language-processing/

Simeon, K. (2017). *Understanding GRU Networks. In this article, I will try to give a… | by Simeon Kostadinov | Towards Data Science*. Towardsdatascience. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

simplilearn. (2022). *What is Tensorflow: Deep Learning Libraries and Program Elements Explained*. https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-tensorflow

Sun, Y., Lu, T., Yu, Z., Fan, H., & Gao, L. (2019). *Computer Supported Cooperative Work and Social Computing*.

Tegegne, W. (2016). The Development of Written Afan Oromo and the Appropriateness of Qubee , Latin Script , for Afan Oromo Writing. *Issn 2224-3178*, *28*(1976), 8–14. https://www.iiste.org

Tesema, W., & Tamirat, D. (2017). *Investigating Afan Oromo Language Structure and Developing Effective File Editing Tool as Plug-in into Ms Word to Support Text Entry and Input Methods. June 2019*. www.pubicon.in

Toews, R. (2020). *AI Will Transform The Field Of Law*. https://www.forbes.com/sites/robtoews/2019/12/19/ai-will-transform-the-field-of-law/?sh=207423e7f01e

Vijaysinh Lendave. (2021). *Guide To Word2vec Using Skip Gram Model* -. https://analyticsindiamag.com/guide-to-word2vec-using-skip-gram-model/

Virtucio, M. B. L., Aborot, J. A., Abonita, J. K. C., Avi, R. S., Copino, R. J. B., Neverida, M. P., Osiana, V. O., Peramo, E. C., Syjuco, J. G., & Tan, G. B. A. (2019). *Predicting Decisions of the Philippine Supreme Court Using Natural Language Processing and Machine Learning*. *June*. https://doi.org/10.1109/COMPSAC.2018.10348

Ward, K. (2014). *Using data to predict Supreme Court's decisions | MSUToday | Michigan State University*. https://msutoday.msu.edu/news/2014/using-data-to-predict-supreme-courts-decisions

Wei, F., Qin, H., Ye, S., & Zhao, H. (2019). Empirical Study of Deep Learning for Text Classification in Legal Document Review. *Proceedings - 2018 IEEE International Conference on Big Data, Big Data 2018*, 3317–3320. https://doi.org/10.1109/BigData.2018.8622157

Youssouf, T. A. (2018). *Hamza / ʔ /; The Difficult Element in the Oromo Orthography*. *January*.

Zhong, H., Guo, Z., Tu, C., Xiao, C., & Liu, Z. (2018). *Legal Judgment Prediction via Topological Learning*. 3540–3549.

APPENDICES

Appendix A: Afaan Oromo Stop-words

itti	jechuun	keessatti	utuu	ta'ullee
egaa	waan	yookaan	henna	bira
kanaafuu	eega	ammo	amma	yoom
bitaa	teenya	booddee	duuba	sana
mirga	gama	dura	booda	nurraa
altakka	karaa	sitti	silaa	inni
al	ffaa	tanaaf	immoo	kana
keessa	akkasumas	eegasii	isii	bitaarra
mirgaarra	malee	fi	hanga	akkuma
gubbaa	warra	keenna	naaf	ykn
yoo	ishiif	waan	jala	ammas
ffaan	utuu	alatti	akka	irra
jara	na	mirgaan	dhaan	illee
ittuu	eegasii	ta’ullee	isaanirraa	bitaan

Appendix B: Normalized Words

Yemmuu=yeroo	Yommuu=yeroo
Yammuu=yeroo	Guraa=gurra
Si’a=yeroo	Ajeesse=ajjeesse
Rukute=rukutuun	Rukutte=rukutuun
Daddabalee=deddeebisee	Mukaa=muka
Warraane=waraane	Bakkaa=bakka
Cuuphee=cuubee	Cubee=cuubee
Miidhamitu=miidhamtu	Dullaa=ulee
Tilmaamaan-tilmaama	Kufisu=kuffise
Dugida=dugda	Miila=miilla
Bolla=boolla	Saman=saamaan

Appendix C: Supporting Result

10-Fold JOC Models Result

A. CNN Model

```
Result of SCV for CNN+FastTEXT
accuracy: 99.36%
accuracy: 98.39%
accuracy: 97.74%
accuracy: 98.39%
accuracy: 99.03%
accuracy: 98.39%
accuracy: 99.35%
accuracy: 98.06%
accuracy: 99.03%
accuracy: 99.68%
98.74% (+/- 0.60%)
```

```
Result of SCV for CNN+W2vec
accuracy: 97.75%
accuracy: 95.16%
accuracy: 88.71%
accuracy: 95.48%
accuracy: 97.42%
accuracy: 94.84%
accuracy: 98.39%
accuracy: 97.42%
accuracy: 98.06%
accuracy: 98.71%
96.19% (+/- 2.82%)
```

B. BiGRU Model

```
Result of SCV for BiGRU+FastTEXT
accuracy: 99.36%
accuracy: 97.74%
accuracy: 98.71%
accuracy: 98.39%
accuracy: 87.74%
accuracy: 98.71%
accuracy: 97.74%
accuracy: 98.39%
accuracy: 98.71%
accuracy: 98.39%
97.39% (+/- 3.25%)
```

```
Result of SCV for BiGRU+W2vec
accuracy: 98.07%
accuracy: 82.58%
accuracy: 97.42%
accuracy: 84.19%
accuracy: 96.45%
accuracy: 98.06%
accuracy: 97.42%
accuracy: 96.77%
accuracy: 98.39%
accuracy: 95.48%
94.48% (+/- 5.62%)
```

C. BiLSTM Model (for penalty prediction)

```
accuracy: 68.44%
accuracy: 70.22%
accuracy: 73.33%
accuracy: 69.78%
accuracy: 69.33%
accuracy: 76.00%
accuracy: 71.43%
accuracy: 75.00%
accuracy: 74.55%
accuracy: 71.43%
71.95% (+/- 2.49%)
```

```
accuracy: 64.89%
accuracy: 64.00%
accuracy: 67.11%
accuracy: 68.89%
accuracy: 62.67%
accuracy: 61.78%
accuracy: 66.07%
accuracy: 68.30%
accuracy: 67.41%
accuracy: 69.20%
66.03% (+/- 2.48%)
```

Appendix D: Sample Code of Model Implementation

Hybrid CNN-BiLSTM model for penalty prediction

```
import numpy
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=7)
cvscores = []
for train, test in kfold.split(X, y.argmax(1)):
    max_length=54
    model=Sequential()
    model.add(embedding_layer)
    model.add(Conv1D(filters=100, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.5))
    model.add(Bidirectional(LSTM(200,return_sequences=True)))
    model.add(Dropout(0.5))
    model.add(Flatten())
    model.add(Dense(29, activation='sigmoid',kernel_regularizer=regularizers.L2(0.001)))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    history=model.fit(X[train], y[train], epochs=30, batch_size=100, verbose=0)
    scores = model.evaluate(X[test], y[test], verbose=0)
    print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
    cvscores.append(scores[1] * 100)
print("%.2f%% (+/- %.2f%%)" % (numpy.mean(cvscores), numpy.std(cvscores)))
```

```
accuracy: 71.56%
accuracy: 75.56%
accuracy: 68.89%
accuracy: 72.44%
accuracy: 72.44%
accuracy: 71.56%
accuracy: 75.89%
accuracy: 76.79%
accuracy: 73.21%
accuracy: 74.55%
73.29% (+/- 2.29%)
```

Sample code for parameter tuning

```
def create_model(neurons):
    model=Sequential()
    model.add(embedding_layer)
    model.add(SpatialDropout1D(0.2))
    model.add(Conv1D(100, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Flatten())
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
seed = 7
tf.random.set_seed(seed)

model = KerasClassifier(model=create_model, epochs=100, batch_size=10, verbose=0)
dropout_rate = [0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
param_grid = dict(model__dropout_rate=dropout_rate)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=3)
grid_result = grid.fit(X, Y)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))
```