

A Project Report on

**CLINICAL DATA ANALYSIS USING  
NAMED ENTITY RECOGNITION**

submitted in partial fulfillment of the requirements  
for the award of the degree

**BACHELOR OF TECHNOLOGY**

in

**COMPUTER SCIENCE AND ENGINEERING**

Submitted by

**GUTTI KAVYA (20021A0544)**

**JINAGAM DALIA BHANU (20021A0504)**

**CHANDU LALITHA HARINI (20021A0503)**

**GANDUBOYINA SHYAM SAI VENKAT ATCHYUT (20021A0521)**

Under the supervision of

**Dr. S.S.S.N. USHA DEVI N.**

Assistant Professor (Ph.D), CSE Department



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA(A)  
JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY  
KAKINADA, KAKINADA-533003, ANDHRA PRADESH, INDIA**

**2020-2024**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)  
JAWHARLAL NEHRU TECHNOLOGICAL UNIVERSITY  
KAKINADA, KAKINADA-533003, ANDHRA PRADESH, INDIA**



**CERTIFICATE**

This is to certify that the project titled **“CLINICAL DATA ANALYSIS USING NAMED ENTITY RECOGNITION”** submitted by **GUTTI KAVYA (20021A0544) , JINAGAM DALIA BHANU (20021A0504), CHANDU LALITHA HARINI (20021A0503), GANDUBOYINA SHYAM SAI VENKAT ATCHYUT (20021A0521)**, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** to the **UCEK(A) JNTUK**, Kakinada. This is a record of bonafide work carried out by him/her under my guidance and supervision during the academic year 2020-24. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission.

**Signature of Supervisor**

Dr. S.S.S.N. Usha Devi N.

Assistant Professor

Department of CSE

UCEK(A)

JNTUK, Kakinada

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)  
JAWHARLAL NEHRU TECHNOLOGICAL UNIVERSITY  
KAKINADA, KAKINADA-533003, ANDHRA PRADESH, INDIA**



**CERTIFICATE**

This is to certify that the project titled **“CLINICAL DATA ANALYSIS USING NAMED ENTITY RECOGNITION”** submitted by **GUTTI KAVYA (20021A0544) , JINAGAM DALIA BHANU (20021A0504), CHANDU LALITHA HARINI (20021A0503), GANDUBOYINA SHYAM SAI VENKAT ATCHYUT (20021A0521)**, in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** to the **UCEK(A) JNTUK**, Kakinada. This is a record of bonafide work carried out by him/her under my guidance and supervision during the academic year 2020-24. The project fulfills the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission.

**Signature of Head of the Department**

Dr. O. Srinivasa Rao

Professor & HOD

Department of CSE

UCEK(A)

JNTUK, Kakinada

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
UNIVERSITY COLLEGE OF ENGINEERING KAKINADA (A)  
JAWHARLAL NEHRU TECHNOLOGICAL UNIVERSITY  
KAKINADA, KAKINADA-533003, ANDHRA PRADESH, INDIA**



**DECLARATION**

We hereby declare that the project report entitled “**CLINICAL DATA ANALYSIS USING NAMED ENTITY RECOGNITION**” submitted by us to University College of Engineering, Jawaharlal Nehru Technological University Kakinada in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering is a record of bonafide project work carried out by us under the guidance of **Dr. S.S.S.N. Usha Devi N.**, Assistant Professor in CSE. We also declare that this project is a result of our effort and that has not been copied from anyone and we have taken only citations from the sources which are mentioned in the references. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

**GUTTI KAVYA (20021A0544)**

**JINAGAM DALIA BHANU (20021A0504)**

**CHANDU LALITHA HARINI (20021A0503)**

**GANDUBOYINA SHYAM SAI VENKAT ATCHYUT (20021A0521)**

## ACKNOWLEDGEMENTS

We want to express our deepest gratitude to the following people for guiding us through this course and without whom this project and its results would not have been completed.

We wish to thank first our supervisor **Dr. S.S.S.N. Usha Devi N.**, Assistant Professor, Department of CSE for accepting and supporting us as a project team. We sincerely convey our gratefulness and heartfelt to our honorable and esteemed project guide for her supervision, guidance, encouragement, and counsel throughout our project. Without her invaluable advice and assistance, it would not have been possible for us to complete this project.

We would like to thank **Dr. O. Srinivas Rao**, Professor, and Head of the Computer Science and Engineering department, UCEK, JNTUK for his fabulous support and useful remarks throughout the project.

We are thankful to **Dr. M.H.M. Krishna Prasad**, Principal, University college of engineering Kakinada (Autonomous), JNTUK for his support and enlightenment during the project.

We are thankful to all the PRC members for their continuous suggestions.

We thank all the teaching and non-teaching staff of the Department of Computer Science and Engineering for their support throughout our project work.

**GUTTI KAVYA (20021A0544)**

**JINAGAM DALIA BHANU (20021A0504)**

**CHANDU LALITHA HARINI (20021A0503)**

**GANDUBOYINA SHYAM SAI VENKAT ATCHYUT (20021A0521)**

# **ABSTRACT**

Clinical Data Analysis using Named Entity Recognition addresses the intricate task of extracting and comprehending vital information from handwritten doctor prescriptions. This entails utilizing image processing concepts, employing Optical Character Recognition (OCR) to convert handwritten text into standard text, and integrating Natural Language Processing (NLP) techniques. The existing models encounter challenges in real-time deployment and performance due to their complexity, manual query generation, and lack of OCR functionality.

The primary objective is to facilitate the seamless extraction and categorization of essential information from handwritten prescription images. The extracted text from OCR undergoes categorization into entities such as drugs, and diseases using Named Entity Recognition (NER).

The core of the project lies in integrating Machine Reading Comprehension (MRC), utilizing Bidirectional Long Short-Term Memory (Bi-LSTM) and BERT (Bidirectional Encoder Representations from Transformers) to enhance contextual understanding of medical entity relationships for precise outputs. Query generation refines the model's ability to formulate meaningful queries, contributing to nuanced comprehension.

In summary, Clinical Data Analysis using Named Entity Recognition employs advanced techniques for precise entity recognition, enhancing insights into clinical narratives. Positioned at the intersection of visual processing, NLP, and clinical data management, the project aims to improve the efficiency of data interpretation.

# TABLE OF CONTENTS

Title	Page No.
<b>ACKNOWLEDGEMENTS</b> . . . . .	ii
<b>ABSTRACT</b> . . . . .	iii
<b>TABLE OF CONTENTS</b> . . . . .	iv
<b>LIST OF FIGURES</b> . . . . .	viii
<b>ACRONYMS AND ABBREVIATIONS</b> . . . . .	ix
<b>CHAPTER 1 INTRODUCTION</b> . . . . .	1
1.1 Introduction to Natural Language Processing (NLP) . . . . .	1
1.1.1 Named Entity Recognition (NER) . . . . .	1
1.1.2 Machine Reading Comprehension (MRC) . . . . .	2
1.2 Deep Learning . . . . .	3
1.2.1 Why Deep Learning? . . . . .	4
1.3 Workflow . . . . .	5
1.3.1 Data Collection . . . . .	5
1.3.2 Data Preprocessing . . . . .	5
1.3.3 Model Selection . . . . .	5
1.3.4 Model Training . . . . .	6
1.3.5 Model Evaluation . . . . .	6
1.3.6 Hyperparameter Tuning . . . . .	7
1.3.7 Model Deployment . . . . .	7
1.4 Problem Statement . . . . .	8

<b>CHAPTER 2</b>	<b>LITERATURE SURVEY</b>	9
<b>CHAPTER 3</b>	<b>SYSTEM ANALYSIS</b>	16
3.1	Existing System	16
3.1.1	Architecture of the Existing System	16
3.1.2	Limitations of the Existing System	17
3.2	Proposed System	18
3.2.1	Models in Proposed System	18
3.2.2	Advantages	18
3.3	Hardware Requirements	19
3.4	Software Requirements	19
3.4.1	Python	19
3.4.2	PyTorch	20
3.4.3	Pandas	20
3.4.4	Keras	20
3.4.5	Transformers	20
3.4.6	Scikit-learn	21
3.4.7	Matplotlib	21
3.4.8	PyPDF2	21
3.4.9	Streamlit	22
<b>CHAPTER 4</b>	<b>Architecture of the Proposed System</b>	23
4.1	Architecture of the Proposed System	23
4.2	BERT Model	24
4.2.1	BERT	24
4.2.2	Architecture	24
4.3	Word Embeddings	25
4.4	Bi-LSTM Model	27
4.4.1	Bi-LSTM	27
4.4.2	Architecture	28



<b>CHAPTER 5 Workflow of the Proposed System . . . . .</b>	<b>31</b>
5.1 Text Extraction from Prescriptions . . . . .	32
5.1.1 Text Extraction from Prescription Image . . . . .	32
5.2 Text Extraction from Prescription PDF . . . . .	32
5.3 Data . . . . .	33
5.3.1 Pubmed Abstracts . . . . .	33
5.3.2 BC5CDR Dataset . . . . .	34
5.4 Data Preprocessing . . . . .	35
5.5 Building the Model . . . . .	35
5.5.1 Cross Entropy Loss . . . . .	36
5.5.2 Adam Optimizer . . . . .	36
5.6 Testing the Model . . . . .	38
5.7 Machine Reading Comprehension Module . . . . .	38
5.8 Automatic Query Generation Module . . . . .	39
5.9 Using the Model . . . . .	40
<b>CHAPTER 6 SOURCE CODE . . . . .</b>	<b>42</b>
6.1 SOURCE CODE . . . . .	42
6.1.1 Importing the necessary Libraries . . . . .	42
6.1.2 Data Preprocessing . . . . .	42
6.1.3 Initializing the Bi-LSTM Module . . . . .	43
6.1.4 Integrating BERT and Bi-LSTM Module . . . . .	44
6.1.5 Preparing BERT inputs and Processing BERT outputs . . . . .	44
6.1.6 Training of the model . . . . .	45
6.1.7 Evaluating the model . . . . .	47
6.1.8 Saving the model . . . . .	49
6.1.9 Testing the model . . . . .	49
6.1.10 Question Answering Module in MRC . . . . .	52
6.1.11 Generating Automatic queries along with Answers . . . . .	52
6.1.12 Extracting text from PDF file . . . . .	53

6.1.13	Extracting handwritten text from Image . . . . .	54
6.1.14	Integrating the code into Streamlit . . . . .	54
<b>CHAPTER 7</b>	<b>OUTPUTS . . . . .</b>	<b>59</b>
7.1	Screenshots . . . . .	59
7.1.1	User Interface . . . . .	59
7.1.2	NER from Manual Text . . . . .	59
7.1.3	Extracting Handwritten text from an Image . . . . .	60
7.1.4	NER from Extracted Text of Image . . . . .	60
7.1.5	Extracting Text from a PDF file . . . . .	60
7.1.6	NER from Extracted Text of PDF . . . . .	61
7.1.7	Automatic Query Generation along with Answers . . . . .	61
7.1.8	Machine Reading Comprehension . . . . .	62
<b>CHAPTER 8</b>	<b>RESULTS AND PERFORMANCE ANALYSIS . . . . .</b>	<b>63</b>
8.1	Results and Performance Analysis . . . . .	63
8.1.1	Existing System . . . . .	63
8.1.2	Proposed System . . . . .	64
8.1.3	Accuracy . . . . .	64
8.1.4	Precision . . . . .	65
8.1.5	Recall . . . . .	65
8.1.6	F1-Score . . . . .	65
<b>CHAPTER 9</b>	<b>CONCLUSION AND FUTURE SCOPE . . . . .</b>	<b>67</b>
9.1	CONCLUSION AND FUTURE SCOPE . . . . .	67
9.1.1	Conclusion . . . . .	67
9.1.2	Future Scope . . . . .	67
<b>REFERENCES</b>	<b>. . . . .</b>	<b>69</b>

## LIST OF FIGURES

1.1	Named Entity Recognition . . . . .	2
1.2	Machine Reading Comprehension . . . . .	3
3.1	Architecture of the Existing System . . . . .	17
4.1	Architecture of the Proposed Model . . . . .	23
4.2	Architecture of BERT . . . . .	25
4.3	Example for Representation of input IDs and attention masks . . . . .	26
4.4	Example for Word Embeddings produced by BERT . . . . .	27
4.5	Architecture of Bi-LSTM . . . . .	28
4.6	Logits- Output produced by Bi-LSTM . . . . .	30
5.1	Workflow of the Proposed Model . . . . .	31
5.2	Example of Pubmed Abstract . . . . .	33
5.3	Screenshot of BC5CDR dataset . . . . .	34
5.4	Loss in each Epoch: Tracking Model Training Progress . . . . .	37
8.1	Performance Comparison of Existing and Proposed Model . . . . .	66
8.2	Classification Report of the Proposed Model . . . . .	66

## ACRONYMS AND ABBREVIATIONS

NER	Named Entity Recognition
MRC	Machine Reading Comprehension
NLP	Natural Language Processing
DL	Deep Learning
OCR	Optical Character Recognition
BERT	Bidirectional Encoder Representations from Transformers
LSTM	Long Short Term Memory
Bi-LSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
BC5CDR	BioCreative V Chemical Disease Relation
Adam	Adaptive Moment Estimation

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Introduction to Natural Language Processing (NLP)**

Natural Language Processing (NLP) is an integral subset of artificial intelligence (AI) dedicated to equipping computers with the ability to comprehend, interpret, and generate human language in a manner that is both meaningful and contextually relevant. Covering a broad spectrum of tasks, ranging from basic functions such as language translation and sentiment analysis to more intricate processes like language generation and comprehension, NLP occupies a significant position at the intersection of computer science, linguistics, and artificial intelligence.

In the realm of healthcare, where crucial and important data reside in unstructured text formats such as medical records and doctors prescriptions, NLP plays a pivotal role in extracting meaningful insights and streamlining informed decision-making processes. By empowering machines to grasp and extract crucial information from these textual sources, NLP emerges as an essential resource, enhancing the efficiency and effectiveness in clinical data analysis.

#### **1.1.1 Named Entity Recognition (NER)**

Named Entity Recognition (NER) is a fundamental task in NLP that focuses on identifying and categorizing entities within text into predefined categories such as names of persons, organizations, locations, dates etc. In specialized fields like healthcare, recognizing entities like drugs and diseases is essential for tasks such as clinical document analysis.

Advanced deep learning techniques, including BERT (Bidirectional Encoder Representations from Transformers) and Bi-LSTM (Bidirectional Long Short-Term Memory), have proven highly effective for NER. These models leverage pre-trained language representations and bidirectional context understanding to accurately classify entities within text.

BERT learns contextual embeddings for each token, capturing bidirectional dependencies and contextual information. Fine-tuning BERT on annotated datasets allows it to adapt to specific NER tasks effectively.

Similarly, Bi-LSTM excels at capturing sequential patterns and long-term dependencies within text. Processing input sequences bidirectionally, Bi-LSTM learns representations that incorporate both past and future context, enhancing entity recognition accuracy.

During his last medical appointment, John, a 45-year-old male, was diagnosed with hypertension, diabetes, and asthma, and was prescribed lisinopril, metformin, and albuterol, respectively.

Figure 1.1: Named Entity Recognition

### 1.1.2 Machine Reading Comprehension (MRC)

Machine Reading Comprehension (MRC) is a domain within NLP focused on training machines to read, comprehend, reason, and respond to questions about unstructured natural language text. Unlike simpler tasks like entity recognition, MRC involves understanding and extracting information from entire passages of text. These systems are designed to answer questions by comprehending both the query and the passage, enabling accurate retrieval of relevant information. In the medical field, MRC technology can be applied to extract answers from medical literature on topics such as drugs, diseases, treatments, and clinical outcomes.

Combining NER with MRC capabilities can enable advanced applications in healthcare, such as automatically extracting drug-disease relationships, summarizing medical documents and assisting healthcare professionals in decision-making processes. These technologies hold significant promise for improving efficiency, accuracy and insights in healthcare settings, ultimately leading to better information extraction from medical documents and outcomes.

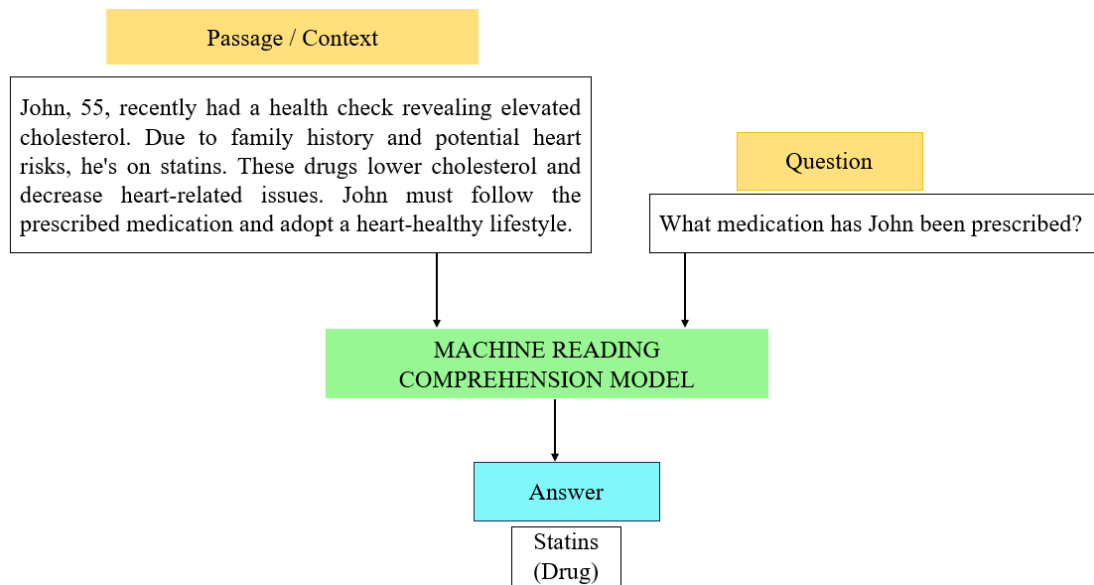


Figure 1.2: Machine Reading Comprehension

## 1.2 Deep Learning

Deep Learning is a subset of machine learning that involves training neural networks to recognize patterns in data. It is inspired by the structure and function of the human brain and is capable of learning from large and complex datasets. Deep Learning has been used to achieve state-of-the-art results in various applications such as computer vision, natural language processing, speech recognition, and robotics.

Deep Learning models typically consist of multiple layers of artificial neurons, each of which performs a simple computation on its inputs. By stacking these layers together, the model can learn complex representations of the input data. The training process involves adjusting the weights and biases of the neurons to minimize the error between the predicted output and the true output.

One of the key advantages of Deep Learning is its ability to automatically learn features from raw data, without the need for manual feature engineering. This has led to significant improvements in the accuracy and robustness of machine learning models. However, Deep Learning also requires large amounts of data and computing resources, making it computationally expensive and time-consuming to train.

### 1.2.1 Why Deep Learning?

Deep Learning (DL) models are highly suitable for NER tasks, particularly in fields such as medical and pharmaceutical domains, where identifying entities like drugs and diseases is crucial. DL models, including architectures like BERT and Bi-LSTM, offer several advantages in this context.

Firstly, DL models excel at learning intricate patterns and dependencies within text data. The entities in medical and pharmaceutical texts exhibit complex contextual relationships and variations. DL models, with their ability to process large volumes of textual data, can effectively capture these nuanced patterns, enhancing the accuracy of entity recognition.

Secondly, DL models eliminate the need for manual feature engineering. In NER tasks, identifying drugs, diseases, and other entities requires understanding both the local context surrounding the entity and the broader context of the entire text. DL models can autonomously learn relevant features directly from the raw text, alleviating the burden of handcrafting features. This feature learning capability enhances the adaptability and robustness of the NER system, enabling it to generalize well to diverse texts and entity types.

Moreover, DL models offer scalability, enabling efficient training on large datasets and facilitating real-time applications. In fields like healthcare, where the volume of textual data continues to expand rapidly, DL models can seamlessly accommodate the growing data influx. Additionally, DL architectures like BERT and BiLSTM are versatile and can be deployed across various platforms, including desktops, servers, and mobile devices, ensuring accessibility to NER applications across different environments.

In essence, the ability of DL models to learn complex patterns, automate feature extraction, scale effectively, and deploy flexibly makes them indispensable for Named Entity Recognition tasks, particularly in domains like medicine and pharmacology.



## **1.3 Workflow**

### **1.3.1 Data Collection**

Data collection is the process of gathering relevant data from various sources for analysis and decision-making. In the context of machine learning, it involves collecting a large and representative dataset that is used to train the model. The data collected should be diverse, well annotated, and balanced to ensure the model can generalize well to new data. Data collection may involve gathering data from public sources, scraping data from websites, or collecting data using sensors or other hardware devices. It is an essential step in the machine learning pipeline and has a significant impact on the performance of the trained model.

### **1.3.2 Data Preprocessing**

Data Preprocessing is a critical step in the Deep Learning workflow that involves preparing the raw data for analysis. It typically involves tasks such as cleaning, normalization, transformation, and feature extraction to ensure that the data is in a format that is suitable for training a model. Data Preprocessing is essential because raw data often contains noise, missing values, and inconsistencies that can lead to inaccurate models. By preprocessing the data, we can improve the quality and reliability of the results obtained from the model. Ultimately, good Data Preprocessing practices can help to ensure that a DL model is accurate, reliable, and scalable.

### **1.3.3 Model Selection**

Model Selection is the process of choosing an appropriate Machine Learning or Deep Learning algorithm or architecture for a particular problem. The goal of Model Selection is to find a model that can generalize well to new, unseen data and minimize the risk of overfitting or underfitting. There are various factors to consider when selecting a model, including the size and complexity of the dataset, the type of problem (e.g., classification or regression), and the available computing resources. Some common approaches to Model Selection include:

- 1. Train-Test Split:** This involves splitting the data into a training set and a vali-

validation set and evaluating the performance of different models on the validation set. This approach is simple but may lead to overfitting if the validation set is too small.

**2. Cross-Validation:** This involves dividing the data into multiple folds and evaluating the performance of different models on each fold. This approach provides a more reliable estimate of the model's performance but can be computationally expensive.

**3. Grid Search:** This involves trying out different combinations of hyperparameters for a given model and selecting the best combination based on a performance metric. Ultimately, the goal of Model Selection is to find a model that achieves high accuracy while avoiding overfitting or underfitting. This requires careful consideration of the problem and dataset, as well as an understanding of the strengths and weaknesses of different Deep Learning models.

### 1.3.4 Model Training

Model training is a critical process in Deep Learning, where a model is trained to recognize patterns and make predictions from input data by adjusting the model's weights and biases using a loss function and an optimization algorithm. The data is divided into training and validation sets, and the model is initialized with random weights and biases. The training data is fed into the model, and the output is compared to the actual output using the loss function. The weights and biases are updated to minimize the loss using an optimization algorithm. The training process is repeated multiple times, with each iteration called an epoch, and the validation set is used to evaluate the model's performance after each epoch. The training process is stopped when the model's performance on the validation set no longer improves. Proper model training is crucial to a model's ability to generalize and make accurate predictions on new and unseen data.

### 1.3.5 Model Evaluation

Model evaluation is the process of assessing the performance of a machine learning model. It involves measuring the model's accuracy, precision, recall, and other metrics on a validation set that was not used during training. The purpose of model evaluation is to determine whether the model is generalizing well to new data

and to identify any areas for improvement. Evaluation metrics can vary depending on the problem and the type of model used. Common evaluation techniques include cross-validation, hold-out validation, and A/B testing. Model evaluation is critical for ensuring the model is robust and reliable in real-world scenarios.

### **1.3.6 Hyperparameter Tuning**

Hyperparameter tuning is the process of selecting the optimal hyperparameters for a machine learning model. Hyperparameters are parameters that are set before training the model and cannot be learned during training. Examples of hyperparameters include learning rate, vocab size, number of epochs, and number of hidden layers. Hyperparameter tuning involves exploring different combinations of hyperparameters and evaluating the model's performance on a validation set. This can be done manually or using automated techniques such as grid search or random search. The goal of hyperparameter tuning is to find the combination of hyperparameters that results in the best performance on the validation set. Proper hyperparameter tuning can significantly improve the performance of a model.

### **1.3.7 Model Deployment**

Model deployment refers to the process of deploying a trained deep learning model into a production environment where it can be used to make predictions on new data. This involves integrating the model with other software systems, setting up infrastructure to handle the model's computation and storage requirements, and ensuring that the model's performance meets the requirements of the production environment. Model deployment also involves ongoing monitoring and maintenance of the deployed model to ensure that it continues to perform well over time. Proper model deployment is essential for ensuring that the model is effective in real-world scenarios and can provide meaningful insights or predictions to end-users.

## **1.4 Problem Statement**

Efficiently extracting information from handwritten doctor prescriptions remains a challenge in clinical data analysis. Integrating OCR and NLP techniques is essential, yet current models face hurdles in real-time deployment, performance, and manual query generation. This project aims to streamline this process by incorporating NER with BERT and Bi-LSTM, alongside MRC for improved comprehension, ultimately enhancing clinical data interpretation.

## **CHAPTER 2**

### **LITERATURE SURVEY**

In the realm of medical imaging and signal processing, the fusion of natural and artificial cognitive systems heralds a new era of understanding and innovation. Presenting a sweeping panorama of this integration, the paper [1] introduces a comprehensive overview of the integration of natural and artificial cognitive systems in medical imaging and signal processing, with a keen focus on the electronic medical records (EMRs) that serve as the bedrock of modern healthcare. It discusses the significance of EMRs as vital sources of medical data and outlines the challenges posed by the complexity and diversity of medical texts. The survey highlights the application of Natural Language Processing (NLP) techniques, such as Named Entity Recognition (NER) and Relation Extraction (RE), in extracting meaningful information from EMRs. It reviews existing research efforts in NER and RE tasks in electronic medical records, emphasizing the importance of advancements in these areas for medical informatics. Additionally, it discusses the limitations of traditional methods and introduces the BERT model as a promising solution due to its rich semantic representation capabilities. The survey underscores the need for an end-to-end framework integrating NER and RE tasks to overcome limitations and improve overall system performance. Furthermore, it introduces multi-task learning as a strategy to leverage correlations between different tasks and enhance the model's generalization ability. Finally, the survey outlines the methodology of the proposed approach, highlighting the utilization of the BERT model for NER and RE tasks, the adoption of an end-to-end framework, and the implementation of multi-task learning for simultaneous training and optimization.

Within the domain of biomedical named entity recognition (BioNER), the study [2] at hand tackles the associated challenges by introducing a novel approach. This method harnesses the power of the BERT model within a machine reading comprehension (MRC) framework. BioNER, crucial for extracting biomedical knowledge from unstructured texts, has conventionally been approached as a sequence labeling

problem, relying on feature engineering and domain-specific knowledge. In contrast, this study formulates BioNER as an MRC problem, utilizing BERT in the BioBERT-MRC framework. Notably, this approach eliminates the need for decoding processes such as conditional random fields (CRF) and introduces well-designed queries to enhance semantic information utilization. The experiments conducted on six BioNER datasets showcase the method’s effectiveness, achieving state-of-the-art performance across BC4CHEMD, BC5CDR-Chem, BC5CDR-Disease, NCBI-Disease, BC2GM, and JNLPBA datasets, with F1-scores ranging from 78.93% to 94.19%. The study emphasizes the advantages of MRC in BioNER, highlighting its capacity to leverage prior knowledge and enhance model performance. Additionally, the work explores the impact of different BERT models and MRC strategies on BioNER performance, shedding light on the effectiveness of the proposed framework compared to traditional sequence labeling methods.

In this paper [3], the authors proposed a novel tagging strategy to address the difficulties posed by overlapping triplets in biomedical datasets, particularly in the context of chemical-protein interactions (CPI) and drug-drug interactions (DDI) extraction. The original training and development sets are combined to form a new training set, with 20% and 10% randomly selected as validation sets for hyperparameter tuning. The tagging approach utilizes queries in the MRC framework to incorporate specific relation information. In comparison to traditional pipelined methods, which separate named entity recognition (NER) and relation classification (RC) tasks, the proposed one-stage MRC method streamlines the extraction process. The MRC4BioER model, which utilizes BERT as the backbone, outperforms several baseline methods, including NovelTagging and CasRel, achieving higher F1 scores on CPI and DDI extraction tasks.

Within the scope of the paper [4], the authors delve into the challenge of machine reading comprehension (MRC) within the healthcare domain. They shed light on the shortcomings of current models in delivering comprehensive, nonconsecutive responses in real-world healthcare scenarios. The proposed SMURF model introduces a novel approach using attentive pooling and a span matrix, demonstrating superior performance on long multispans datasets (MASHQA and MASHQA-S). The experi-

ments showcase optimal effectiveness in outperforming state-of-the-art MRC models, emphasizing its contribution to addressing the shortcomings of current systems. The study also discusses the prevalence of recent MRC systems utilizing large-scale pre-trained language models like BERT, Roberta, ELECTRA, and XLNet.

The GFMRC model introduced in the paper [5] presents novel features, incorporating a sentence-level N-gram training method to capture contextual information and a hybrid representation extraction technique that combines CNN and LSTM models. The model is validated on both English (CoNLL 2003, OntoNotes 5.0) and Chinese (OntoNotes 4.0, MSRA) datasets, demonstrating improved performance over the baseline BERT-MRC+DSC model across various evaluation metrics. The paper underscores the importance of NER in information extraction, discussing traditional and deep learning-based approaches. GFMRC’s framework consists of a preprocessing module, a feature extraction module utilizing CNN and LSTM, and an entity prediction module, emphasizing its effectiveness in capturing contextual information and enhancing NER performance.

In addressing the need for enhanced Chinese clinical named entity recognition, the authors in the paper [6] embarked on a comprehensive exploration, beginning with the construction of a corpus sourced from website data. Their proposed NER method integrates BERT, Convolutional Neural Networks (CNN), Bidirectional Long Short-Term Memory (Bi-LSTM), and Conditional Random Field (CRF), exploiting BERT’s contextual word vector generation. For experimentation, a dataset is collected from health websites, comprising 8750 sentences across seven entity types, divided into training, validation, and test sets. Results demonstrate the model’s superior performance, achieving an optimal F1-score of 87.84%, credited to BERT’s contextual information, CNN’s local feature extraction, Bi-LSTM’s long-distance characteristics, and CRF’s sequence selection.

The challenges for Named Entity Recognition (NER) in the health-preserving field were addressed by the authors in the paper [7] by proposing a model based on BERT, CNN, LSTM, and CRF. The absence of publicly available datasets in health preserving necessitated the creation of a corpus by extracting data from websites and defining seven types of entities. The proposed BERT-CNN-LSTM-CRF model

comprises four layers, incorporating BERT for semantic feature extraction, CNN for local feature extraction, Bi-LSTM for capturing global characteristics, and CRF for optimal sequence tagging. This model demonstrates superiority over other models, achieving a remarkable F1-score of 87.84.

The paper [8] addresses a proposed method called Dic-Att-BiLSTM-CRF (DABLC) for disease named entity recognition (NER) that combines an efficient string matching approach with a disease dictionary based on the Disease Ontology, along with a unique dictionary attention layer. Leveraging a bidirectional long short-term memory network and conditional random field (Bi-LSTM - CRF), DABLC achieves superior performance on NCBI disease and BioCreative V CDR corpora, achieving F1 scores of 88.6% on NCBI and 88.3% on BioCreative V CDR datasets, outperforming state-of-the-art methods. The model addresses challenges in identifying rare diseases, complex names, and tagging inconsistencies.

To improve the Named Entity Recognition (NER) tasks through context similarity-based data augmentation, the authors in the paper [9] introduced COSINER (Context Similarity-based data augmentation for NER). Unlike existing methods that may generate noisy samples, COSINER replaces entity mentions with more plausible ones based on context similarity. The approach outperforms baselines in various few-shot scenarios, addressing limitations in training data. Notably, COSINER achieves competitive computing times compared to simpler augmentation methods and surpasses models relying on pre-trained architectures. The study contributes by extending the baseline, analyzing execution times, providing precision and recall values, and demonstrating interpretability through relevant tokens in NER.

For extracting medication names and prescription information from unstructured medical discharge summaries, the authors of the paper [10] introduced a machine learning approach. By leveraging word embeddings and framing the task as sequence labeling, the proposed method achieves a horizontal phrase-level F1-measure of 0.864 on the 2009 i2b2 Challenge benchmark set, surpassing the current state-of-the-art. This underscores the efficacy of the machine learning approach in improving the extraction of prescription details from medical documents.

Various architectures are proposed in [11] to address the task of identifying



and categorizing entities into domains such as problems, treatments, tests, and more. These architectures include BiLSTM n-CRF, BiLSTM-CRF-Smax-TF, and BiLSTM n-CRF-TF, designed specifically for multi-label entity tagging. Evaluation on the 2010 i2b2/VA and i2b2 2012 datasets demonstrates the effectiveness of these approaches, achieving top NER scores of 0.903 and 0.808, respectively. Additionally, high span-based micro-averaged F1 polarity scores of 0.832 and 0.836, along with macro-averaged F1 polarity scores of 0.924 and 0.888, underscore the robustness of the proposed methodologies.

Using a domain-specific pre-trained BERT model, the study described in [12] introduces a medical specialty prediction system to address the imperative for precise outpatient treatment initiation during the COVID-19 pandemic. Fine-tuned specifically for 27 medical specialties, the model exhibits superior performance compared to four competitive deep learning NLP models in both cross-validation and testing. Demonstrated interpretability through case studies and contributions include comprehensive specialty prediction and the use of BERT pre-trained specifically for the medical domain. However, data quality concerns and variations in prediction difficulty across specialties are noted limitations, necessitating cautious interpretation.

The Medical Query Generator (MQG) introduced in [13] addresses user challenges in expressing specific medical information needs concisely. MQG analyzes information need descriptions, selecting terms correlated to medical categories to enhance precision. Comparative evaluations show MQG's superiority over methods using complete dictionaries of medical terms. Empirical evaluation on the OHSUMED (Oregon Health Sciences University MEDical) database further validates its effectiveness. Additionally, MQG reduces the computational burden on text rankers and search engines by retrieving fewer terms in generated queries, providing a practical solution to the limitations of traditional keyword-based searches for complex medical queries.

Leveraging a modified BiLSTM-CNN-Char DL architecture on Apache Spark, the study [14] introduces a cutting-edge clinical and biomedical Named Entity Recognition (NER) algorithm. It achieves state-of-the-art accuracy on biomedical

NER benchmarks and clinical concept extraction challenges, surpassing commercial solutions like AWS Medical Comprehend and Google Cloud Healthcare API without relying on memory-intensive language models. This agile implementation, available in the Spark NLP library, offers flexibility in training with any embeddings, minimal hyperparameter tuning, and support for multiple languages, making it invaluable for NLP tasks in healthcare.

NER and Relation Extraction (RE) tasks in unstructured clinical text are at the forefront of analysis, highlighting advancements in clinical information extraction in the paper [15]. With machine learning methods, especially Transformer-based models, showing promise, their translation into clinical practice is limited. Challenges such as dataset scarcity and tool availability underscore the need for further research translation and validation for effective implementation in clinical settings.

The paper [16] details the development of a specialized language model tailored for biomedical NLP tasks, achieved through pre-training on a large biomedical text corpus. It covers aspects such as data selection, model architecture adaptations for biomedical nuances, and evaluation metrics including accuracy in entity recognition and relation extraction. A comparison with general-purpose models like BERT or GPT were also included to highlight the efficacy of the specialized approach.

The resource, BioCreative V CDR task corpus, is used for extracting chemical-disease relationships from biomedical texts which is discussed in [17]. It details the creation process, data sources, annotation guidelines, and corpus structure. Additionally, it discusses applications for machine learning models and potential research challenges.

BioBERT, which is discussed in the paper [18], introduces a specialized pre-trained language model tailored for biomedical text mining tasks. It enhances the BERT architecture through fine-tuning on a large biomedical corpus, including scientific articles and clinical notes. Key aspects discussed include the pre-training process, tokenization scheme, and modifications to accommodate biomedical language. Evaluation results highlight BioBERT's superior performance in tasks like named entity recognition and relation extraction compared to other models. Its avail-

ability for integration into NLP frameworks enables broader usage in biomedical text mining research.

Handwritten text recognition, as discussed by the authors in [19], delves into the development and evaluation of models utilizing deep learning techniques. It details architectures like CNNs, RNNs, or LSTM networks, along with preprocessing steps such as normalization and augmentation. The paper explores datasets comprising handwritten text samples, presenting experimental results on performance metrics and comparisons with other methods. Additionally, it discusses practical applications and future directions for enhancing handwritten text recognition systems.

## **CHAPTER 3**

### **SYSTEM ANALYSIS**

#### **3.1 Existing System**

The existing system uses the approach of BERT and Long Short-Term Memory (LSTM) models for NER in clinical data analysis. BERT comprehensively understands medical records by capturing contextual information bidirectionally, while LSTM specializes in analyzing sequential dependencies in clinical data. This hybrid architecture empowers clinicians and researchers to extract valuable insights from complex clinical datasets efficiently.

##### **3.1.1 Architecture of the Existing System**

The existing model architecture combines BERT, a transformer model, with the LSTM layer, forming a robust system for extracting valuable insights from complex medical data. BERT excels at understanding medical text by analyzing the contextual nuances within patient records and reports. Its bidirectional nature allows it to capture relationships between words efficiently, aiding in the accurate interpretation of medical terminology and syntax.

On the other hand, LSTM networks specialize in analyzing sequential data, such as patient vitals or longitudinal records, by capturing long-range dependencies and temporal patterns.

By integrating BERT and LSTM models within the architecture, the system leverages the strengths of both approaches synergistically. BERT's contextual understanding forms a solid foundation for interpreting clinical text data, while LSTM's sequential analysis capabilities enable the model to capture temporal dynamics efficiently.

This integration enables the model to navigate through the complexities of clinical datasets, extracting deeper insights than would be possible with either approach alone. Ultimately, this combined architecture empowers clinicians and researchers

to uncover hidden patterns, make informed decisions, and drive transformative discoveries in healthcare research and practice.

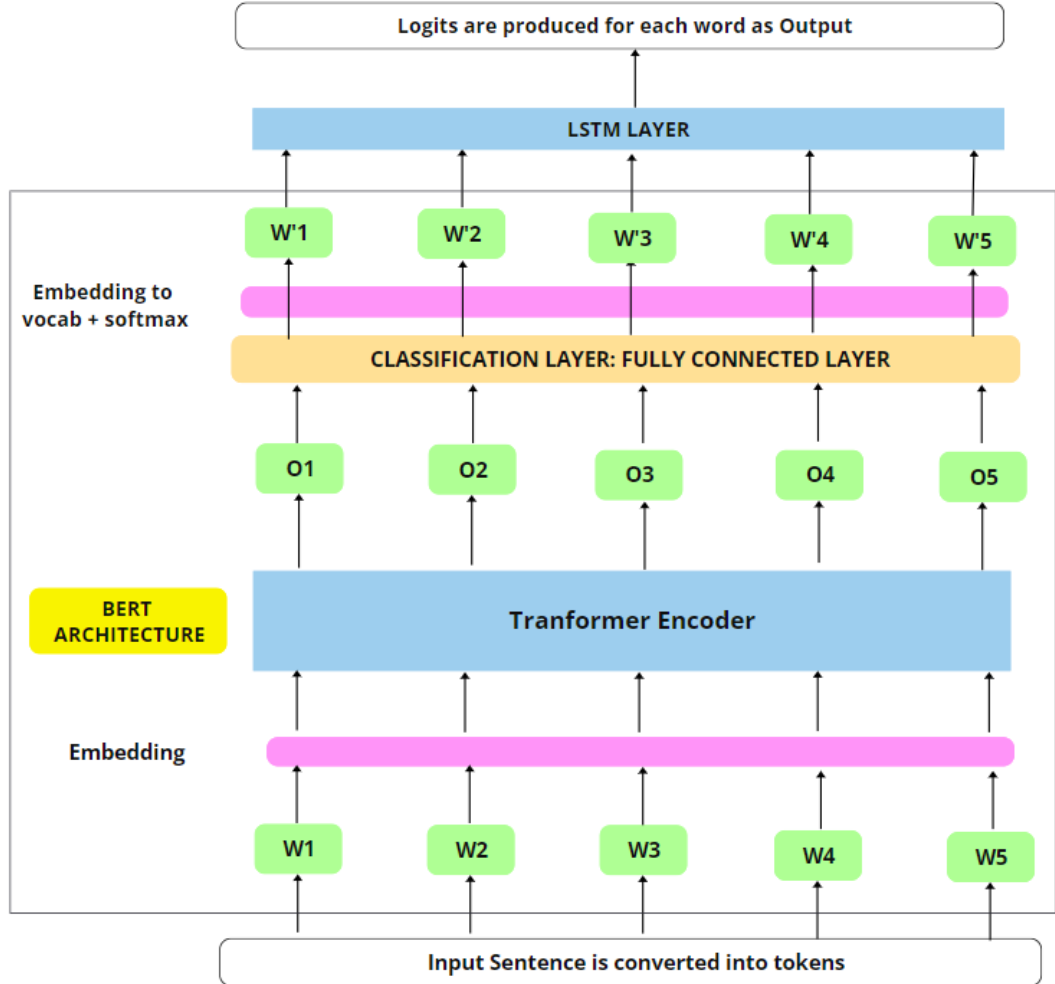


Figure 3.1: Architecture of the Existing System

### 3.1.2 Limitations of the Existing System

1. **Computational Complexity:** Integrating BERT and LSTM models increases computational demands, leading to longer training times and higher resource requirements.
2. **Lack of OCR functionality:** The current system cannot accurately extract text from handwritten doctor prescriptions, leading to errors in data interpretation.
3. **Manual Query Generation:** Query generation in the existing system relies heavily on manual intervention, resulting in time-consuming and error-prone processes. Manual query generation not only increases the workload for users but also

introduces inconsistencies and inaccuracies in the extracted information, leading to suboptimal results.

**4. Difficulty in Capturing Long-Term Dependencies:** While LSTM is designed to address the vanishing gradient problem in traditional RNNs, it may still struggle to capture long-term dependencies in sequences that extend over many time steps. This limitation can affect the model's ability to understand complex temporal relationships in data.

## **3.2 Proposed System**

### **3.2.1 Models in Proposed System**

In our proposed system, we leverage two key components: BERT and Bi-LSTM. BERT, a transformer-based language representation model, is employed to encode contextual information from clinical text data, such as medical records and reports. Its bidirectional approach enables it to capture rich contextual information by considering both preceding and succeeding words in a sentence, facilitating a deeper understanding of medical terminologies and relationships within patient data. Complementing this, Bi-LSTM, a variant of the Long Short-Term Memory (LSTM) architecture, is utilized for analyzing time-series clinical data.

By processing sequences bidirectionally, Bi-LSTM captures temporal dependencies effectively, enabling the system to discern patterns indicative of disease progression or treatment effectiveness. Together, BERT and Bi-LSTM form a powerful foundation for our proposed system, facilitating comprehensive analysis and interpretation of complex healthcare datasets with enhanced accuracy and efficiency.

### **3.2.2 Advantages**

The advantages of the proposed system are:

- 1. Enhanced Accuracy:** BERT's contextual encoding and Bi-LSTM's bidirectional processing contribute to improved accuracy in clinical data analysis tasks. The combination of these models allows for a deeper understanding of medical terminologies and relationships within the, leading to more precise interpretations

and predictions of entities.

2. **Better Contextual Understanding:** The contextual understanding provided by advanced NLP techniques enables the system to grasp the subtle nuances and relationships present in clinical data. By integrating NER with Bi-LSTM and BERT, the system gains the ability to accurately identify entities like diseases and drugs within handwritten prescriptions.

3. **Automatic Query Generation:** The model's ability to generate meaningful queries enhances its comprehension of clinical narratives. This refinement contributes to more nuanced understanding and precise extraction of essential information.

### 3.3 Hardware Requirements

1. **RAM:** 8 GB
2. **Hard Disk:** 100 GB
3. **Processor:** Intel Core i5 or i7 processor
4. **Speed:** 2 GHz or higher
5. **Processing Unit:** GPU (Graphics Processing Unit) Preferred, CPU (Central Processing Unit) Supported

### 3.4 Software Requirements

#### 3.4.1 Python

Python is a high-level programming language widely used in deep learning due to its simplicity, readability, and availability of numerous libraries and frameworks. Python provides a wide range of libraries for deep learning, including TensorFlow, Keras, PyTorch, and Transformers. Python's syntax allows developers to write concise and expressive code that can be easily maintained and extended. Additionally, Python has a large and active community that constantly contributes to the development of new libraries and tools, making it easier to work with deep learning. With its ease of use and versatility, Python has become the language of choice for many deep learning applications.

### **3.4.2 PyTorch**

PyTorch is a popular open-source machine learning framework that provides tensor computation with strong GPU acceleration and deep neural networks built on a tape-based autograd system. It is widely used for building and training neural network models, including those for natural language processing tasks such as NER and BERT. PyTorch's flexibility and ease of use make it suitable for prototyping, experimentation, and deploying deep learning models in production environments.

### **3.4.3 Pandas**

Pandas is a powerful Python library for data manipulation and analysis, particularly suited for working with structured data. It provides data structures like DataFrame and Series, along with functions to efficiently manipulate and analyze data, including data cleaning, transformation, and aggregation. Pandas is essential for preprocessing and organizing clinical data before feeding it into machine learning models, as well as for post-processing and analyzing model outputs.

### **3.4.4 Keras**

Keras is a high-level neural network API written in Python. It is designed to enable fast experimentation with deep neural networks and focuses on being user-friendly, modular, and extensible. Keras supports multiple backends, including TensorFlow, CNTK, and Theano, allowing users to choose the best backend for their specific hardware configuration. It provides a simple and intuitive interface for building and training various types of deep learning models, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and others. Keras also offers various pre-trained models, making it easy for users to leverage pre-trained models for various tasks such as image classification, object detection and NLP.

### **3.4.5 Transformers**

Transformers is an open-source library developed by Hugging Face, designed for natural language understanding tasks using state-of-the-art transformer-based models. It offers pre-trained transformer models like BERT, GPT, and RoBERTa, along with a unified API for using these models for various NLP tasks such as text classification,



NER, and question answering. Transformers library provides easy integration with PyTorch and TensorFlow, enabling seamless implementation of transformer-based models in NLP applications.

### **3.4.6 Scikit-learn**

Scikit-learn is a versatile machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It offers a wide range of algorithms for classification, regression, clustering, dimensionality reduction, and model evaluation. scikit-learn is used in this project to evaluate the performance metrics of machine learning models, such as accuracy, precision, recall, and F1-score, as well as for tasks like cross-validation and hyperparameter tuning.

### **3.4.7 Matplotlib**

Matplotlib is a comprehensive library for creating static, interactive, and animated visualizations in Python. It provides a MATLAB-like interface for generating plots, charts, histograms, and other types of graphical representations. Matplotlib is utilized in this project for visualizing various aspects of the data, model performance, and insights derived from clinical data analysis, aiding in data exploration, interpretation, and presentation.

### **3.4.8 PyPDF2**

PyPDF2 is a light-weight Python library designed for PDF manipulation. With PyPDF2, you can easily read, write, and modify PDF files. Its features include extracting text and meta data, merging and splitting PDFs, manipulating individual pages, encrypting and decrypting documents, adding annotations and filling out forms. Additionally, PyPDF2 allows integration with PDF extractor tools for more advanced tasks, such as extracting specific content or structured data from PDF documents. Compatible with both Python 2.x and 3.x, PyPDF2 is a versatile tool for various PDF-related tasks, from document management to automation scripts.

### **3.4.9 Streamlit**

Streamlit is an open-source Python framework for building interactive web applications for machine learning and data science projects. It allows developers to create intuitive and user-friendly interfaces for showcasing data visualizations, models, and insights, without the need for web development expertise. Streamlit is employed in this project to create a user interface for interacting with the clinical data analysis system, enabling users to input handwritten prescriptions, visualize model outputs, and explore analysis results in a seamless and interactive manner.

## CHAPTER 4

### Architecture of the Proposed System

#### 4.1 Architecture of the Proposed System

The architecture of the proposed system involving the process of predicting entities such as drugs and diseases is represented in Fig 4.1.

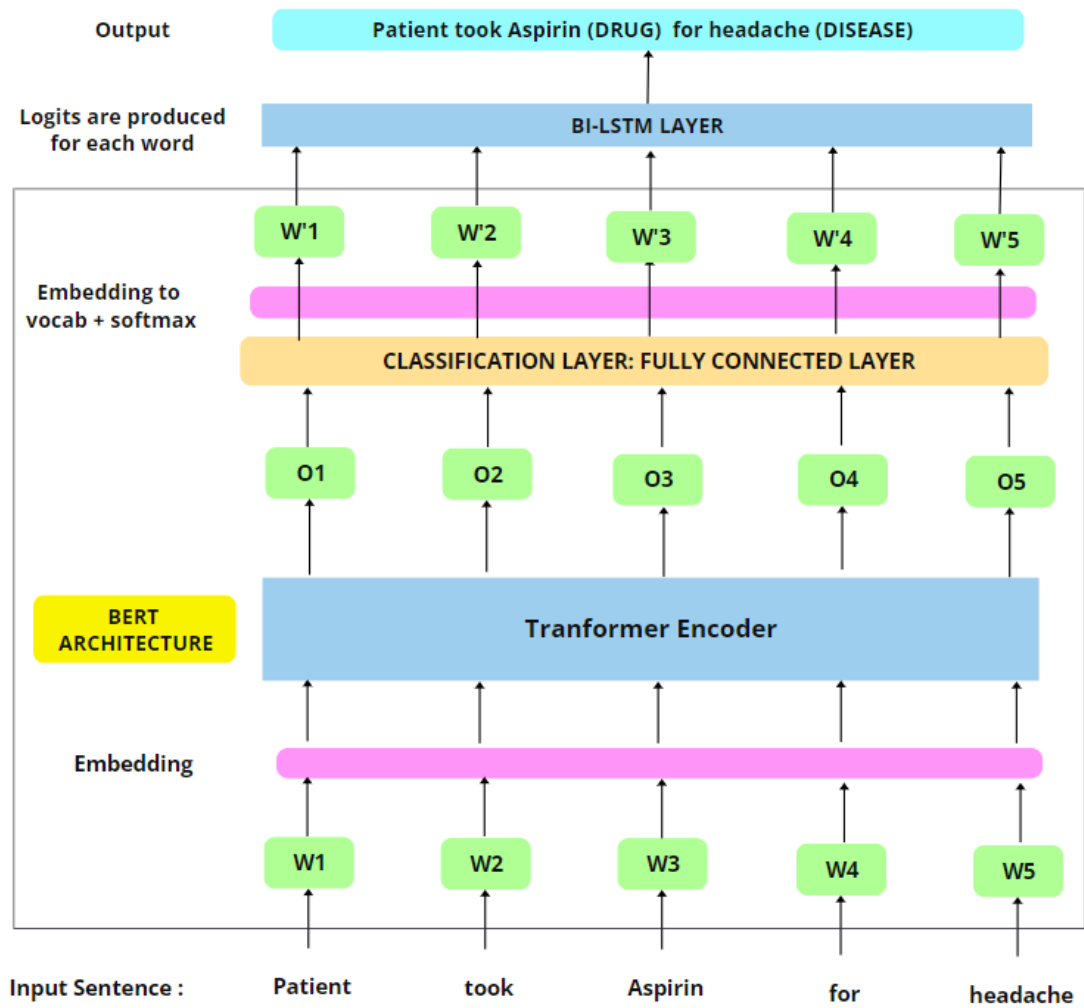


Figure 4.1: Architecture of the Proposed Model

## 4.2 BERT Model

### 4.2.1 BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a pre-trained NLP model introduced by Google in 2018. It's built upon the Transformer architecture, a neural network architecture that's particularly effective for sequence-to-sequence tasks like language translation and text generation.

BERT is used for NER because of its effectiveness in capturing contextual information from a given text.

Some of the reasons why BERT is well-suited for NER tasks:

1. **Contextual Understanding:** BERT captures bidirectional context information, which helps it understand the relationships between words in a sentence. This contextual understanding is crucial for accurately identifying named entities, especially in cases where the meaning of a word depends on its surrounding words.
2. **Pre-trained Representations:** BERT is pre-trained on large corpora of text, which means it has learned rich representations of words and phrases. These pre-trained representations can be fine-tuned on NER-specific datasets, allowing the model to adapt to the specific nuances of the task.
3. **State-of-the-art Performance:** BERT has achieved state-of-the-art performance on a wide range of NLP tasks, including NER. Its effectiveness in capturing complex linguistic patterns and contextual information contributes to its success in NER.

### 4.2.2 Architecture

The model architecture for clinical data analysis utilizes a combination of cutting-edge techniques to process textual data extracted from handwritten doctor prescriptions. To begin with, the input text undergoes tokenization using the BERT tokenizer, which divides the sentences into individual tokens. These tokens are then encoded into input IDs and attention masks, representing the indices of tokens in the BERT

vocabulary and indicating which tokens should be attended to during processing, respectively.

Once the input IDs and attention masks are generated, they are organized into a dataset using a DataLoader, with a batch size of 32 for efficient processing. This DataLoader facilitates the batching of input sequences, enabling smoother integration into the model pipeline. Subsequently, the word embeddings for each token are generated by passing the input IDs through the BERT model. These word embeddings capture contextual information about each token's meaning within the sentence, providing rich representations crucial for downstream processing.

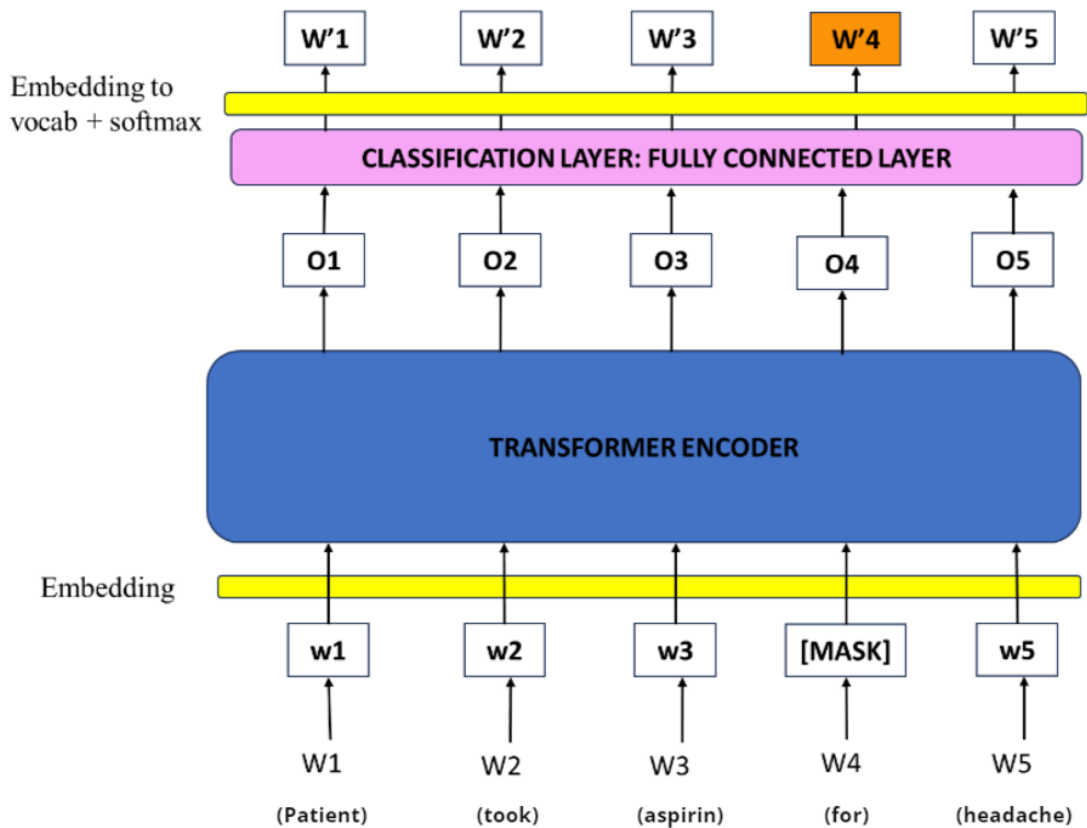


Figure 4.2: Architecture of BERT

### 4.3 Word Embeddings

Word embeddings refer to numerical representations of words or tokens generated by the BERT model. These embeddings capture rich semantic information about each word or token in the input text.

```

Shape of input_ids: torch.Size([3, 32])
Shape of attention_masks: torch.Size([3, 32])
Batch 0
Input IDs: tensor([[ 101, 2182, 2003, 1996, 2117, 6251, 1012, 102, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0],
                  [ 101, 1998, 2023, 2003, 1996, 2353, 6251, 1012, 102, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0]])
Attention Masks: tensor([[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])
Batch 1
Input IDs: tensor([[ 101, 2023, 2003, 1996, 2034, 6251, 1012, 102, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0],
                  [ 101, 2023, 2003, 1996, 2034, 6251, 1012, 102, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    0, 0, 0, 0, 0, 0, 0, 0]])
Attention Masks: tensor([[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
                        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])

```

Figure 4.3: Example for Representation of input IDs and attention masks

When the input sentences from handwritten doctor prescriptions are tokenized and encoded into input IDs and attention masks, they are passed through the BERT model, which has been pre-trained on large corpora of text. During this process, BERT generates dense vector representations, or embeddings, for each token in the input sequences.

These word embeddings encode contextual information about the meaning and usage of each token within the context of the entire sentence. They capture semantic relationships between words, allowing the model to understand the nuanced meanings of words based on their surrounding context. For example, the word "cold" may have different embeddings depending on whether it refers to a medical condition or a low temperature.

In the subsequent stages of the model pipeline, such as the Bidirectional Long Short-Term Memory (Bi-LSTM) layer, these word embeddings serve as input features, providing the model with rich contextual information to make predictions or perform further processing tasks, such as NER.

```
BaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=tensor([[[ 0.0131, -0.2477, -0.1943, ..., 0.0981, 0.0220, 0.5718],
[ -0.4217, -0.4706, -0.8185, ..., 0.0962, 0.1811, -0.3232],
[ 0.2471, -0.2295, -0.2051, ..., -0.6135, -0.0994, -0.0831],
...,
[ 0.6919, -0.0849, -0.2851, ..., 0.0241, -0.7946, -0.2957],
[ 0.5887, -0.1068, -0.3718, ..., 0.1842, -0.6782, -0.3788],
[ -0.3277, -0.0380, -0.4514, ..., 0.2994, -0.0083, 0.1644]],
[[ -0.4908, -0.5611, -0.7175, ..., 0.0371, 0.0876, 0.3455],
[ 0.8556, -0.2203, -0.1927, ..., 0.3428, 0.5939, 0.3270],
[ 0.6708, -0.1220, -0.6512, ..., -0.0923, 0.1940, -0.1461],
...,
[ -0.0740, -0.6620, 0.1258, ..., 0.0504, -0.1122, 0.4148],
[ -0.1952, -0.7806, 0.2138, ..., 0.1999, -0.2303, 0.4887],
[ -0.4231, -0.5478, 0.1246, ..., 0.1092, -0.4221, 0.3782]]],
grad_fn=<NativeLayerNormBackward0>), pooler_output=tensor([[-0.8261, -0.5324, -0.6394, ..., -0.3795, -0.5999, 0.5613],
[-0.8924, -0.5832, -0.7281, ..., -0.7445, -0.5627, 0.6637]],
grad_fn=<TanhBackward0>), hidden_states=None, past_key_values=None, attentions=None, cross_attentions=None)
```

Figure 4.4: Example for Word Embeddings produced by BERT

## 4.4 Bi-LSTM Model

### 4.4.1 Bi-LSTM

A Bidirectional Long Short-Term Memory (Bi-LSTM) is a type of recurrent neural network (RNN) architecture that is particularly effective for modeling sequential data, such as text. It extends the capabilities of traditional LSTM networks by processing input sequences in both forward and backward directions simultaneously.

Some of the reasons why Bi-LSTM is used for NER tasks:

1. **Sequential Modeling:** Bi-LSTM is well-suited for sequential modeling tasks because it can capture dependencies between words in a sentence. In NER, the context surrounding a word is crucial for accurately identifying named entities, and BiLSTM's ability to model both past and future context helps in this regard.
2. **Long-Term Dependencies:** Traditional RNNs suffer from the vanishing gradient problem, which makes it challenging to capture long-range dependencies in sequential data. Bi-LSTM partially mitigates this issue by processing sequences in both directions, allowing it to capture long-term dependencies more effectively.
3. **Contextual Understanding:** Named Entity Recognition often requires understanding the context in which a word appears to determine whether it represents a named entity or not. Bi-LSTM's bidirectional processing enables it to capture rich contextual information from both preceding and subsequent words, enhancing its

ability to recognize named entities accurately.

4. **Feature Extraction:** BiLSTM serves as a feature extractor, transforming the word embeddings generated by models like BERT into higher-level representations that capture relevant patterns for NER. These representations are then used as input features for subsequent layers or classifiers in the NER model.

#### 4.4.2 Architecture

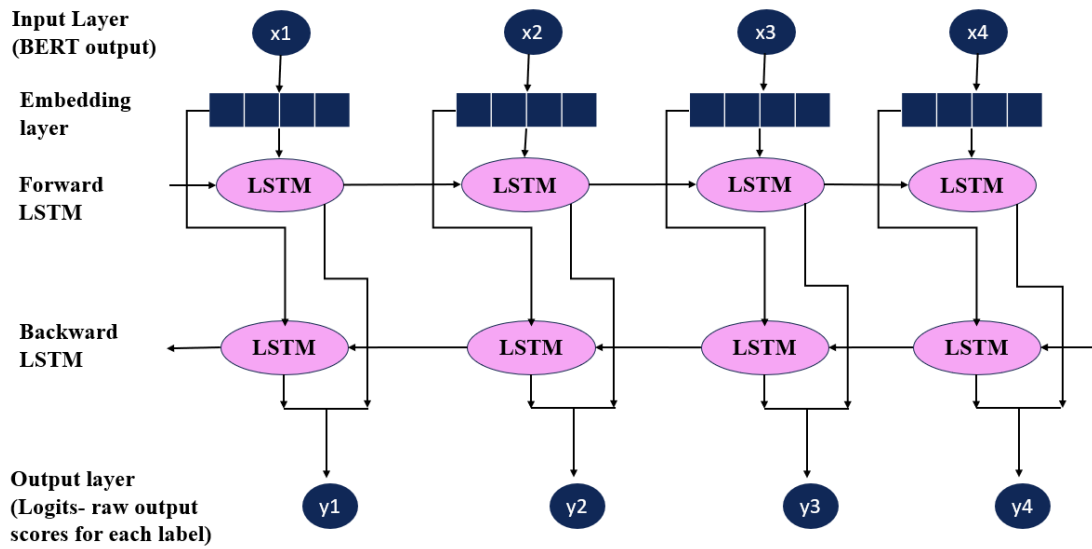


Figure 4.5: Architecture of Bi-LSTM

The architecture of Bi-LSTM consists of several key components, each serving a specific purpose in the NER task.

1. **Input Layer:** The input to the Bi-LSTM architecture consists of pre-trained word embeddings generated by a BERT model. These embeddings encode rich semantic information about each token in the input sequences.

2. **Embedding Layer:** The pre-trained word embeddings are passed through an embedding layer, which maps each token to a dense vector representation. This layer helps to transform the input tokens into a continuous vector space where similar tokens are closer together, facilitating meaningful computation.



3. **Bi-LSTM Layer:** The embedding vectors are then fed into the Bi-LSTM layer, which consists of two LSTM networks: one processing the input sequence in the forward direction, and the other processing it in the backward direction. Each LSTM unit within the Bi-LSTM layer maintains a cell state and hidden state, allowing it to capture sequential dependencies and context information within the input sequences. By processing the input sequences in both forward and backward directions, the Bi-LSTM layer can effectively capture both past and future context for each token, enhancing the model's ability to understand the input text.

4. **Fully Connected Linear Layer:** The output of the Bi-LSTM layer is passed through a fully connected linear layer, which performs a linear transformation of the Bi-LSTM output. This layer serves to further process the Bi-LSTM output and prepare it for the final stage of the NER task.

5. **Output Layer (Logits Generation):** The output of the fully connected linear layer is then used to generate logits for each word in the input sequences. Logits represent the raw predictions made by the model. The logit score with the highest value determines the entity of that particular word.

```
Logits: tensor([[[ 6.9532, -8.4944, -1.8565, -2.7000, -4.7531],
 [ 5.0676, -3.9644, -3.2633, -2.2685, -3.6761],
 [ 5.5390, -5.4102, -1.2592, -3.0459, -4.6066],
 [ 5.0569, -2.9148, -2.2061, -3.5175, -5.9050],
 [ 5.2172, -4.8358, -3.2428, -2.3144, -3.1565],
 [ 4.9883, -5.8931, -3.1013,  0.0353, -4.2075],
 [ 6.5507, -4.9236, -2.9546, -4.2632, -4.2127],
 [ 6.5946, -5.8935, -2.5516, -2.4528, -5.2934],
 [ 5.6999, -4.5744, -2.0240, -2.7141, -6.2183],
 [ 6.0035, -6.5878, -2.7456, -1.5250, -3.9137],
 [-0.3271, -5.4017,  4.7200,  3.6629, -8.2276],
 [-0.7952, -3.6084,  3.0797,  3.4612, -5.2694],
 [ 4.9717, -6.4447, -3.3717,  0.5801, -1.5680],
 [ 5.9748, -3.9487, -3.1764, -3.5273, -4.2246],
 [ 5.5661, -4.6454, -3.6248, -2.7533, -2.8720],
 [ 5.7180, -5.9437, -1.8243, -2.8120, -3.8152],
 [ 5.0168, -6.1257, -3.0381,  0.1786, -3.3599],
 [ 6.4004, -6.2624, -2.8488, -2.2080, -5.5667],
 [ 5.4284, -2.8018, -3.8883, -3.1464, -4.6421],
 [ 5.7130, -4.0513, -4.6547, -2.7967, -1.8988],
 [ 5.0311, -6.0421, -3.0885,  0.2076, -3.6580],
 [ 6.3407, -5.6343, -3.9912, -2.7718, -3.3498],
 [ 5.0352, -5.9558, -3.0041,  0.1133, -4.3507],
 [ 6.5725, -4.9759, -2.9519, -4.2423, -4.2210],
 [ 6.5227, -5.4527, -4.2987, -1.7030, -5.3136],
 ...])
```

Figure 4.6: Logits- Output produced by Bi-LSTM

## CHAPTER 5

### Workflow of the Proposed System

#### 5 Workflow of the Proposed System

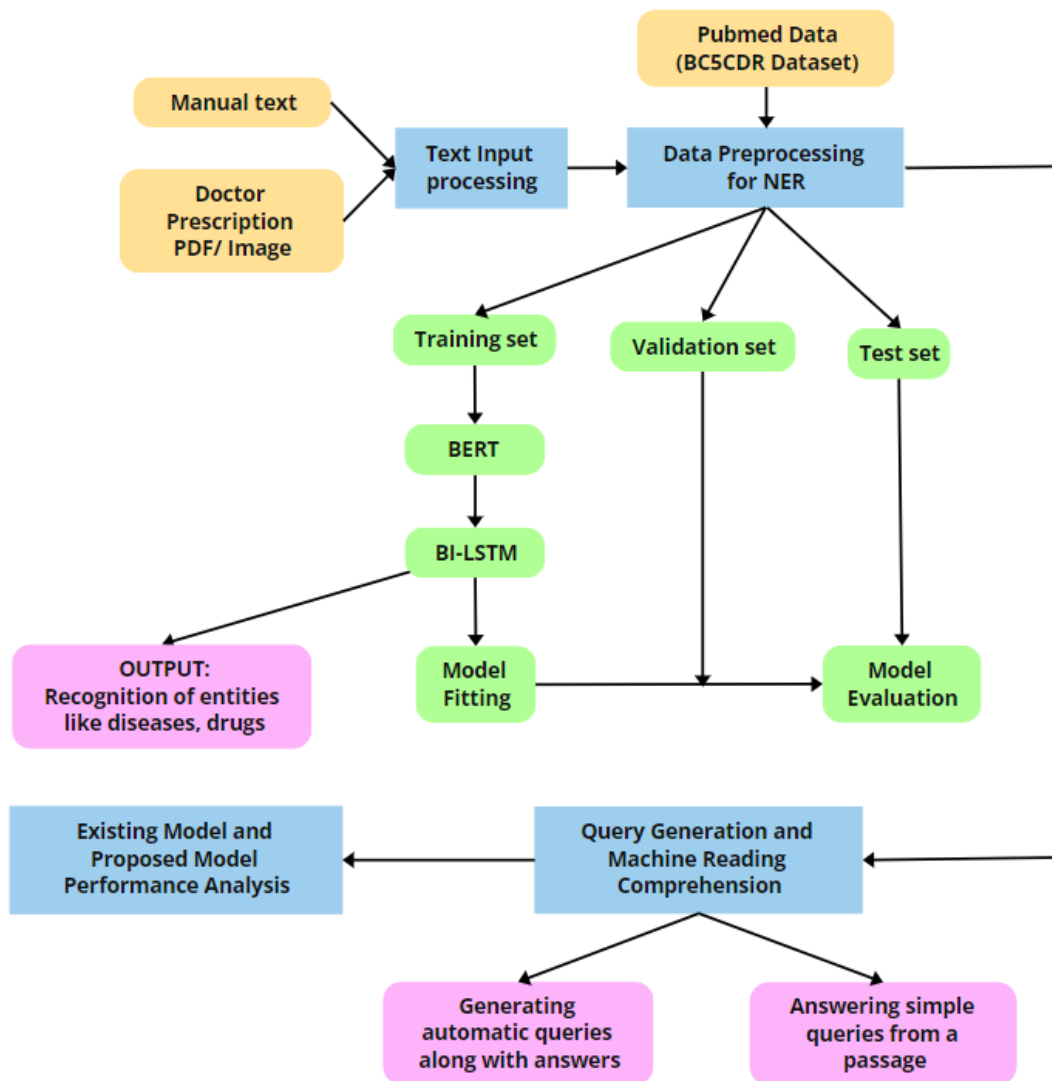


Figure 5.1: Workflow of the Proposed Model

## **5.1 Text Extraction from Prescriptions**

### **5.1.1 Text Extraction from Prescription Image**

The initial phase of the proposed system is the pivotal task of text extraction, laying the foundation for the intricate domain of OCR. This foundational step is where Keras-OCR comes into play, presenting itself as an open-source Python library, streamlining OCR tasks with its robust Convolutional Neural Network (CNN) architecture.

Leveraging the prowess of deep learning, particularly CNNs, Keras-OCR excels in deciphering text from images with suboptimal accuracy. Its architecture comprises a series of convolutional layers adept at discerning diverse features within input images. Following this, a fully connected layer synthesizes these insights into precise text predictions.

The training process of Keras-OCR hinges upon a meticulously curated dataset, where images are meticulously annotated with corresponding text. Through iterative optimization, the model minimizes the disparity between predicted and ground truth text, enhancing its predictive prowess. Once trained, Keras-OCR seamlessly integrates into workflows, effortlessly discerning text from novel images. Preprocessing steps, including normalization of brightness and contrast, removing noise, grey-scale conversion, precede text recognition, ensuring suboptimal performance across various image types.

## **5.2 Text Extraction from Prescription PDF**

Similarly, if the input file is of the type PDF, PDF-Extractor is used to extract text from the file. Leveraging its intuitive functionalities, users can effortlessly retrieve text content from uploaded PDF files. Within this streamlined process, the module employs a combination of temporary file handling and PdfReader functionalities to navigate through each page of the PDF document, aggregating text content iteratively. This ensures comprehensive extraction of text from multi-page PDFs, enhancing the versatility of the extraction process.

The text extracted through Keras-OCR and PyPDF2 facilitates entity recognition and machine reading comprehension, enabling the generation of answers to questions and the formulation of meaningful queries.

## 5.3 Data

### 5.3.1 Pubmed Abstracts



The screenshot shows the PubMed interface. At the top, the NIH National Library of Medicine logo is visible. A search bar contains the text "covid 19". Below the search bar, the search results for "COVID-19 diagnosis -A review of current methods" are displayed. The authors listed are Meral Yüce, Elif Filiztekin, and Korin Gasia Özkaya. The abstract text is visible, starting with "A fast and accurate self-testing tool for COVID-19 diagnosis has become a prerequisite to comprehend the exact number of cases worldwide...". On the right side, there are links for "FULL TEXT LINKS", "Cite", and "Collections".

Figure 5.2: Example of Pubmed Abstract

PubMed is a free search engine accessing primarily the MEDLINE database of references and abstracts on life sciences and biomedical topics. The United States National Library of Medicine (NLM) at the National Institutes of Health maintains the database as part of the Entrez system of information retrieval. PubMed contains over 32 million citations for biomedical literature from MEDLINE, life science journals, and online books. These citations may include links to full-text content

from PubMed Central and publisher websites.

PubMed data typically consists of bibliographic information such as author names, publication titles, journal names, publication dates, and abstracts. Researchers, clinicians, and other users can search PubMed to find articles relevant to their interests, ranging from basic research studies to clinical trials and systematic reviews. PubMed abstract data” refers to a subset or collection of abstracts extracted from the PubMed database.

### 5.3.2 BC5CDR Dataset

The BC5CDR dataset, also known as the BioCreative V Chemical Disease Relation dataset, is a widely used resource in the field of biomedical text mining and NLP. It focuses specifically on extracting relationships between chemicals and diseases from biomedical literature.

The BC5CDR dataset, a cornerstone in biomedical text mining, is meticulously constructed using PubMed abstracts, which serve as the primary data source. Through careful processing, each abstract is tokenized into individual words or entities, forming the basis for annotation. These tokens are then tagged with numerical labels to signify their entity type within the text. Specifically, the tagging schema employs the codes "1" and "2" to denote the beginning of chemical and disease entities, respectively, while "3" and "4" represent the continuation of a disease or chemical entity, respectively. Additionally, the tag "0" signifies entities classified as "other." This systematic approach ensures the precise identification and categorization of biomedical entities, facilitating the development of robust machine learning and NLP models tailored to extract complex relationships between chemicals and diseases. As a result, the BC5CDR dataset stands as a pivotal resource for advancing research in biomedical informatics and catalyzing innovations in medical science.

[illegible]

Figure 5.3: Screenshot of BC5CDR dataset

## 5.4 Data Preprocessing

In the initial phase of the deep learning workflow, the gathering and preparation of the dataset lay the foundation for subsequent model training. For instance, let's consider the BC5CDR dataset, which comprises 16,423 lines of tokenized text extracted from PubMed abstracts, each accompanied by corresponding tags. This dataset encompasses over 300,000 words, providing a substantial corpus for training purposes.

Tokenization, a crucial preprocessing step, converts the sequences into discrete tokens, enabling efficient processing by the model. To work with this data, tokenization is applied to convert the sequences into discrete tokens. Each token, along with its associated tag, is then processed. In the provided context, a dictionary of tokens and their tags is constructed. Tokens are mapped to unique indices, facilitating efficient processing by the model.

The dataset preparation process is crucial for Named Entity Recognition (NER) tasks, as it ensures that the model receives properly formatted input data. By converting tokens to indices and tags to numerical labels, the dataset becomes compatible with deep learning models like BERT.

Additionally, it's essential to consider preprocessing steps such as handling class imbalances, padding sequences to a uniform length, and performing exploratory data analysis to understand the dataset's characteristics and potential biases. These steps contribute to the overall effectiveness of the deep learning model in accurately identifying named entities within text.

## 5.5 Building the Model

Following the data preprocessing stage, the prepared dataset undergoes model training to facilitate accurate NER. In this training process, the dataset, previously tokenized and tagged, is loaded and preprocessed to ensure compatibility with the model architecture.

The dataset, divided into batches for efficient processing, is fed into the training loop iteratively over a specified number of epochs. Within each epoch, the model is trained to minimize the loss function, optimizing its ability to predict NER tags

accurately.

During training, the model is transferred to the appropriate device—either a GPU if available, else the CPU—to leverage hardware acceleration for faster computations. Hyperparameters such as vocabulary size, embedding dimension, hidden dimension, output size (representing the number of NER tags), number of epochs, and learning rate are set to guide the training process.

The BERT and Bi-LSTM Model combined, a neural network architecture, is instantiated with the provided hyperparameters. This model combines the power of BERT and Bi-LSTM layers to capture both contextual and sequential information in the input data.

### 5.5.1 Cross Entropy Loss

Cross-entropy loss, a fundamental concept in machine learning, serves as a crucial metric, particularly in classification tasks. Derived from information theory, it evaluates the disparity between two probability distributions: the actual distribution of labels and the predicted distribution generated by the model. At its core, cross-entropy loss measures the inefficiency of encoding the true distribution using the predicted probabilities. This loss function penalizes the model more heavily for confident incorrect predictions, providing a means to train models that aim for accurate and calibrated probability estimates.

In mathematical terms, if  $y_i$  represents the true probability distribution (usually a one-hot vector with 1 for the true class and 0 elsewhere) and  $\hat{y}_i$  represents the predicted probability distribution (the output of the softmax function), then the cross-entropy loss  $L$  is often defined as:

$$L(y, \hat{y}) = - \sum_i y_i \log(\hat{y}_i)$$

### 5.5.2 Adam Optimizer

The Adaptive Moment Estimation (Adam) optimizer is a widely utilized algorithm in deep learning for optimizing neural networks. It stands out due to its adaptive learning rate mechanism, which adjusts the learning rates of individual model parameters during training. By maintaining separate estimates for the first and second



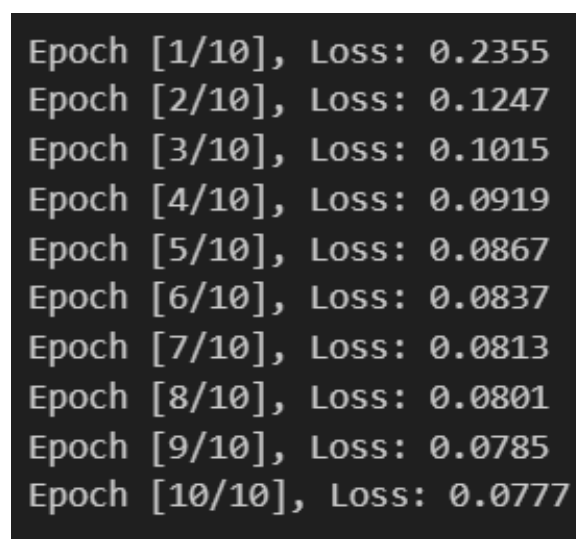
moments of the gradients, Adam can dynamically adapt the learning rates based on the characteristics of the gradients. This adaptability often leads to faster convergence and better performance compared to traditional optimization algorithms like stochastic gradient descent (SGD).

To optimize the model's performance, a loss function (CrossEntropyLoss) and an optimizer (Adam) are defined. The optimizer adjusts the model's parameters based on the gradients computed during backpropagation, aiming to minimize the loss.

Within each epoch, the training loop iterates through the batches of the dataset. For each batch, the model computes predictions (logits) based on the input tokens and their corresponding attention masks, generated by the BERT model. These predictions are then compared with the actual tags, and the resulting loss is calculated.

Through backpropagation, the gradients of the loss with respect to the model's parameters are computed, and the optimizer updates these parameters accordingly to improve the model's performance. This process iterates until all batches have been processed within the epoch.

At the end of each epoch, the average loss over all batches is computed and displayed, providing insight into the model's training progress. This iterative training process continues for the specified number of epochs, refining the model's ability to accurately identify named entities within the text data.



Epoch [1/10],	Loss: 0.2355
Epoch [2/10],	Loss: 0.1247
Epoch [3/10],	Loss: 0.1015
Epoch [4/10],	Loss: 0.0919
Epoch [5/10],	Loss: 0.0867
Epoch [6/10],	Loss: 0.0837
Epoch [7/10],	Loss: 0.0813
Epoch [8/10],	Loss: 0.0801
Epoch [9/10],	Loss: 0.0785
Epoch [10/10],	Loss: 0.0777

Figure 5.4: Loss in each Epoch: Tracking Model Training Progress

After training, the model's performance is evaluated on a validation dataset. This

evaluation assesses metrics like loss, accuracy, precision, recall, and F1-score. It helps understand how well the model generalizes to unseen data, guiding adjustments to improve performance and prevent overfitting.

## **5.6 Testing the Model**

After training the BERT and Bi-LSTM model using the training and validation sets, the performance of the model is evaluated on the test set. The purpose of testing is to assess the generalization ability of the model, i.e., how well it performs on unseen data.

To test the model, the test set is passed through the trained model, and the predicted outputs are compared with the ground truth labels. The performance metrics such as accuracy, precision, recall and F1-score are calculated to evaluate the model's performance.

The accuracy is calculated as the ratio of the correctly predicted samples to the total number of samples in the test set. Other metrics such as precision, recall, and F1-score provide more detailed information about the model's performance. Precision measures the proportion of true positives among the predicted positives, while recall measures the proportion of true positives among the actual positives. The F1-score is the harmonic mean of precision and recall and provides a balance between the two.

Overall, this helps to ensure that the model can perform well on unseen data and is not over fitting to the training data. It also provides insights into the model's strengths and weaknesses and helps to identify areas for improvement.

## **5.7 Machine Reading Comprehension Module**

For the machine reading comprehension task, the proposed model utilizes a pipeline from Transformers called "question-answering." This pipeline is trained on various comprehension-related datasets but fine-tuned using the Stanford Question Answering Dataset (SQuAD). SQuAD consists of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to each question is a segment of text from the corresponding passage.

The training process involves fine-tuning a pre-trained transformer model such as BERT on SQuAD data. The model is first initialized with pre-trained weights from large-scale language modeling tasks, such as masked language modeling or next sentence prediction. Then, it's fine-tuned on the SQuAD dataset using a task-specific objective function that encourages the model to accurately predict the answer span within the passage for each question.

## 5.8 Automatic Query Generation Module

In order to generate automatic queries from a given text, the proposed model utilises a pre-trained model from transformers called "castorini/doc2query-t5-base-msmarco". This model is a variant of the T5 (Text-To-Text Transfer Transformer) architecture specifically fine-tuned for the task of document-to-query generation. The dataset used to train this model is the MS MARCO (Microsoft MACHine Reading COMprehension) dataset. MS MARCO is a large-scale dataset containing real user queries collected from Bing search engine logs along with corresponding documents that were likely to contain the answer to the query.

The core architecture of T5 is based on the transformer model, which consists of multiple layers of self-attention mechanisms and feed-forward neural networks. These layers allow the model to capture intricate relationships and dependencies within input sequences. The transformer architecture facilitates bidirectional processing of input sequences, enabling the model to effectively understand and generate text.

In T5, the model consists of both an encoder and a decoder. The encoder processes the input text and generates a contextualized representation of it, while the decoder takes this representation and generates the output text. During training, T5 is trained to map input text sequences to target text sequences using a large-scale dataset and a pre-defined objective function, such as maximum likelihood estimation or masked language modeling.

In practical use, the T5-based model seamlessly integrates into document-to-query workflows. Leveraging the `T5Tokenizer` and `T5ForConditionalGeneration` classes from the Hugging Face Transformers library, the code efficiently preps the model for inference. Given document text, the model encodes it into tokens

and generates multiple query candidates. Incorporating sampling and top-k strategies, it explores diverse query possibilities to capture relevant information. These generated queries then undergo decoding and question answering, extracting informative insights from the document content. This streamlined approach highlights T5's effectiveness in natural language understanding, promising enhancements to information retrieval systems, particularly in generating questions from given text.

## 5.9 Using the Model

The following are the sequence of steps involved in using the model:

**Step 1: Input Text Extraction:** The user provides input in the form of manual text, images, or PDFs containing doctors' prescriptions.

**Step 2: Text Preprocessing:** The input text undergoes preprocessing, including extraction of relevant information and conversion into a format suitable for model input.

**Step 3: BERT and BiLSTM Processing:** The preprocessed text undergoes BERT model processing, generating word embeddings as output. These embeddings serve as input for the Bi-LSTM model. Finally, the Bi-LSTM model generates logits as the ultimate output.

**Step 4: Entity Prediction:** The model predicts entities such as diseases and drugs from the input text, providing insights into the medical information contained within.

**Step 5: Automated Queries Generation:** Based on the identified entities, automated queries are generated to extract additional information or clarify medical details. These queries might seek clarification on dosage, frequency, or drug-disease relationships.

**Step 6: Machine Reading Comprehension (MRC):** The input text undergoes machine reading comprehension (MRC). This involves understanding the context of the queries within the text and retrieving relevant information to provide accurate answers.

By following these steps, the model facilitates the extraction and analysis of medical information from doctor's prescriptions, generates queries to further understand the data, and employs MRC techniques to extract answers, enabling efficient medical information retrieval and analysis.

# CHAPTER 6

## SOURCE CODE

### 6.1 SOURCE CODE

#### 6.1.1 Importing the necessary Libraries

```
import torch
import torch.nn as nn
from torch.utils.data import Dataset, DataLoader
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score
```

#### 6.1.2 Data Preprocessing

```
class NERDataset(Dataset):
    def __init__(self, data_path, tokenizer, max_length):
        self.data = pd.read_csv(data_path)
        self.data['Token'] = self.data['Token'].astype(str)
        self.token_to_idx = {token: idx for idx, token in
            enumerate(sorted(set(self.data['Token'])))}
        self.tag_to_idx = {0: 0, 1: 1, 2: 2, 3: 3, 4: 4}
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        token = self.data.iloc[idx]['Token']
```

```

tag = self.data.iloc[idx]['Tag']

token_idx = self.token_to_idx[token]
tag_idx = self.tag_to_idx[tag]

return {
    'token_idx': token_idx,
    'tag_idx': tag_idx
}

```

### 6.1.3 Initializing the Bi-LSTM Module

```

import torch.nn as nn

class BiLSTMModel(nn.Module):
    def __init__(self, vocab_size, embedding_dim, hidden_dim,
output_size):
        super(BiLSTMModel, self).__init__()
        self.embedding = nn.Embedding(vocab_size,
embedding_dim)
        self.lstm = nn.LSTM(embedding_dim, hidden_dim,
bidirectional=True, batch_first=True)
        self.fc = nn.Linear(hidden_dim * 2, output_size)

    def forward(self, x):
        embedded = self.embedding(x)
        lstm_output, _ = self.lstm(embedded)
        tag_space = self.fc(lstm_output)
        return tag_space

```

### 6.1.4 Integrating BERT and Bi-LSTM Module

```
class BERT_BiLSTM_Model(nn.Module):
    def __init__(self, bert_model, hidden_dim,
        output_size, vocab_size):
        super(BERT_BiLSTM_Model, self).__init__()
        self.bert = bert_model
        self.hidden_dim = hidden_dim
        self.bilstm = BiLSTMModel(vocab_size,
            bert_model.config.hidden_size, hidden_dim,
            output_size)

    def forward(self, input_ids, attention_mask):
        bert_output = self.bert(input_ids=input_ids,
            attention_mask=attention_mask)[0]
        bilstm_output = self.bilstm(bert_output)
        return bilstm_output
```

### 6.1.5 Preparing BERT inputs and Processing BERT outputs

```
import torch
from torch.utils.data import TensorDataset, DataLoader,
RandomSampler
from transformers import BertTokenizer, BertModel

device = torch.device('cuda' if torch.cuda.is_available()
is_available() else 'cpu')
def prepare_bert_inputs(sentences):
    tokenizer = BertTokenizer.from_pretrained
        ('bert-base-uncased')
    bert_model = BertModel.from_pretrained
        ('bert-base-uncased')
    inputs = tokenizer(sentences, padding=True,
```



```

truncation=True, return_tensors="pt")
input_ids = inputs["input_ids"]
attention_masks = inputs["attention_mask"]
return input_ids, attention_masks, tokenizer, bert_model

def process_bert_outputs(input_ids,
attention_masks, bert_model):
    batch_size = 32
    train_data = TensorDataset(input_ids, attention_masks)
    train_sampler = RandomSampler(train_data)
    train_dataloader = DataLoader(train_data,
sampler=train_sampler, batch_size=batch_size)

    outputs_all = []
    for batch in train_dataloader:
        batch_input_ids, batch_attention_masks = batch
        outputs = bert_model(input_ids=batch_input_ids,
attention_mask=batch_attention_masks)
        outputs_all.append(outputs)
    return outputs_all

tokenizer = BertTokenizer.from_pretrained
('bert-base-uncased')
bert_model = BertModel.from_pretrained
('bert-base-uncased')
bert_model.to(device)

```

### **6.1.6 Training of the model**

```

device = torch.device('cuda' if torch.cuda.is_available()
else 'cpu')
max_length = 128

```

```

dataset = NERDataset('bc5cdr/train.csv',
tokenizer, max_length)
train_dataset = NERDataset('bc5cdr/train.csv', tokenizer,
max_length)
train_loader = DataLoader(train_dataset, batch_size=32,
shuffle=True)
token_to_idx = train_dataset.token_to_idx
vocab_size = len(dataset.token_to_idx)
train_loader = DataLoader(train_dataset, batch_size=32,
shuffle=True)

```

```

VOCAB_SIZE = vocab_size
EMBEDDING_DIM = 128
HIDDEN_DIM = 256
OUTPUT_SIZE = 5
NUM_EPOCHS = 10
LEARNING_RATE = 0.001

```

```

model = BERT_BiLSTM_Model(bert_model, HIDDEN_DIM,
OUTPUT_SIZE, VOCAB_SIZE)
model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(model.parameters(),
lr=LEARNING_RATE)

```

```

for epoch in range(NUM_EPOCHS):
    model.train()
    total_loss = 0
    num_batches = len(train_loader)
    for batch in train_loader:
        tokens = batch['token_idx'].to(device)
        tags = batch['tag_idx'].to(device)

```

```

input_ids, attention_masks, _, _ =
prepare_bert_inputs(tokens)
input_ids = input_ids.to(device)
attention_masks = attention_masks.to(device)
tags = tags.to(device)
optimizer.zero_grad()
bert_outputs = process_bert_outputs(input_ids,
attention_masks, bert_model)
logits = model(bert_outputs, attention_masks)
logits = logits.view(-1, OUTPUT_SIZE)
tags = tags.view(-1)
loss = criterion(logits, tags)
loss.backward()
optimizer.step()
total_loss += loss.item()
average_loss = total_loss / num_batches
print(f'Epoch [{epoch+1}/{NUM_EPOCHS}], Loss:
{average_loss:.4f}')

```

### 6.1.7 Evaluating the model

```

import torch
from torch.utils.data import Dataset, DataLoader
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

valid_dataset = NERDataset('bc5cdr/valid.csv',
tokenizer, max_length)
valid_loader = DataLoader(valid_dataset, batch_size=32,
shuffle=False)

```

```

def evaluate_model(model, dataloader, criterion):
    model.eval()
    total_loss = 0
    all_preds = []
    all_labels = []
    with torch.no_grad():
        for batch in dataloader:
            tokens = batch['token_idx'].to(device)
            tags = batch['tag_idx'].to(device)
            logits = model(tokens)
            loss = criterion(logits.view(-1, OUTPUT_SIZE),
                             tags.view(-1))
            total_loss += loss.item()
            preds = torch.argmax(logits, dim=-1)
            all_preds.extend(preds.cpu().numpy().flatten())
            all_labels.extend(tags.cpu().numpy().flatten())
    average_loss = total_loss / len(dataloader)
    accuracy = accuracy_score(all_labels, all_preds)
    precision = precision_score(all_labels, all_preds,
                                average='macro')
    recall = recall_score(all_labels, all_preds,
                           average='macro')
    f1 = f1_score(all_labels, all_preds, average='macro')
    return average_loss, accuracy, precision, recall, f1

val_loss, val_accuracy, val_precision, val_recall,
val_f1 = evaluate_model(model, valid_loader, criterion)

print(f'Validation Loss: {val_loss:.4f}')
print(f'Validation Accuracy: {val_accuracy:.4f}')
print(f'Validation Precision: {val_precision:.4f}')
print(f'Validation Recall: {val_recall:.4f}')

```

```
print(f'Validation F1-score: {val_f1:.4f}')
```

### 6.1.8 Saving the model

```
import torch
model_save_path = 'NER_Model.pth'
torch.save(model, model_save_path)
print(f"Model saved to '{model_save_path}'")
```

### 6.1.9 Testing the model

```
import pandas as pd
test_data = pd.read_csv('bc5cdr/test.csv')
true_labels = test_data['Tag'].tolist()
print(true_labels[:10])

import torch
from torch.utils.data import Dataset, DataLoader
import pandas as pd
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score

class TestNERDataset(Dataset):
    def __init__(self, data_path, token_to_idx):
        self.data = pd.read_csv(data_path)
        self.data['Token'] = self.data['Token'].astype(str)
        self.token_to_idx = token_to_idx
    def __len__(self):
        return len(self.data)
    def __getitem__(self, idx):
        token = self.data.iloc[idx]['Token']
        token_idx = self.token_to_idx.get(token, 0)
```

```

        return {'token_idx': token_idx}

model_save_path = 'NER_Model.pth'
model = torch.load(model_save_path)
test_dataset = TestNERDataset('bc5cdr/test.csv',
token_to_idx)
test_loader = DataLoader(test_dataset, batch_size=32,
shuffle=False)

def test_model(model, dataloader):
    model.eval()
    all_preds = []
    with torch.no_grad():
        for batch in dataloader:
            tokens = batch['token_idx'].to(device)
            logits = model(tokens)
            preds = torch.argmax(logits, dim=-1)
            all_preds.extend(preds.cpu().numpy().flatten())
    return all_preds

test1_predictions = test_model(model, test_loader)
test_accuracy = accuracy_score(true_labels,
test1_predictions)
test_precision = precision_score(true_labels,
test1_predictions,
average='macro')
test_recall = recall_score(true_labels, test1_predictions,
average='macro')
test_f1 = f1_score(true_labels, test1_predictions,
average='macro')

print(f'Test Accuracy: {test_accuracy:.4f}')

```

```

print(f'Test Precision: {test_precision:.4f}')
print(f'Test Recall: {test_recall:.4f}')
print(f'Test F1-score: {test_f1:.4f}')

import spacy
from spacy import displacy

def perform_ner(text):
    nlp_bc5cdr = spacy.load("en_ner_bc5cdr_md")
    doc_bc5cdr = nlp_bc5cdr(text)
    combined_entities = []
    for ent in doc_bc5cdr.ents:
        label = "DRUG"
        if ent.label_ == "CHEMICAL"
        else ent.label_
        combined_entities.append((ent.start_char,
ent.end_char, label))
    return doc_bc5cdr, combined_entities
def visualize_entities(doc, entities):
    colors = {
        "DRUG": "lightblue",
        "DOSAGE": "grey",
        "DISEASE": "lightgreen"
    }
    ents = []
    for start, end, label in entities:
        ents.append({"start": start, "end":
end, "label": label, "color":
colors.get(label, "white")})
    html = displacy.render([{"text":
doc.text, "ents": ents, "title": None}],
style="ent", manual=True, options={"colors": colors})
    return html

```

### 6.1.10 Question Answering Module in MRC

```
from transformers import pipeline as transformers_pipeline
def perform_qa(question, context):
    nlp_pipeline = transformers_pipeline
    ("question-answering")
    result = nlp_pipeline(question=question, context=context)
    return result["answer"]
```

### 6.1.11 Generating Automatic queries along with Answers

```
import torch
from transformers import T5Tokenizer,
T5ForConditionalGeneration
from transformers import pipeline as transformers_pipeline

def generate_queries(doc_text):
    device = torch.device('cuda' if torch.cuda.is_available()
    else 'cpu')
    tokenizer = T5Tokenizer.from_pretrained
    ('castorini/doc2query-t5-base-msmarco')
    model = T5ForConditionalGeneration.from_pretrained
    ('castorini/doc2query-t5-base-msmarco').to(device)

    input_ids = tokenizer.encode(doc_text,
    return_tensors='pt').to(device)
    outputs = model.generate(
        input_ids=input_ids,
        max_length=64,
        do_sample=True,
        top_k=10,
```



```

num_return_sequences=3)

generated_queries = [tokenizer.decode(output,
skip_special_tokens=True) for output in outputs]
answers = [perform_qa(query, doc_text) for query in
generated_queries]
return generated_queries, answers

```

### 6.1.12 Extracting text from PDF file

```

from PyPDF2 import PdfReader
import tempfile
import os
import streamlit as st

def extract_text_from_pdf(uploaded_file):
    text = ""
    with st.spinner("Extracting text..."):
        with tempfile.NamedTemporaryFile(delete=False)
as tmp_file:
            tmp_file.write(uploaded_file.read())
            tmp_file.seek(0)
            pdf_path = tmp_file.name
        with open(pdf_path, "rb") as file:
            reader = PdfReader(file)
            for page_num in range(len(reader.pages)):
                page = reader.pages[page_num]
                text += page.extract_text()
        os.unlink(pdf_path) # Remove temporary PDF file
    return text

```

### 6.1.13 Extracting handwritten text from Image

```
import streamlit as st
import keras_ocr

def get_image():
    uploaded_file = st.file_uploader("Upload Image File",
    type=['jpg', 'jpeg', 'png'])
    if uploaded_file is not None:
        return uploaded_file
    else:
        return None

def perform_ocr(image):
    pipeline = keras_ocr.pipeline.Pipeline()
    predictions = pipeline.recognize([image])[0]
    text = ' '.join([text for text, _ in predictions])
    return text
```

### 6.1.14 Integrating the code into Streamlit

```
import streamlit as st
from pdf_extractor import extract_text_from_pdf
from ocr import get_image, perform_ocr
from predict import perform_ner, visualize_entities
from qa import perform_qa
from query_generation import generate_queries;
import base64
import streamlit as st
import plotly.express as px

def main():
    df = px.data.iris()
    @st.cache_data
    def get_img_as_base64(file):
        with open(file, "rb") as f:
```

```

        data = f.read()

        return base64.b64encode(data).decode()

img = get_img_as_base64("bg.jpg")
page_bg_img = f"""
<style>

[data-testid="stAppViewContainer"] > .main {{
background-image:
url("https://wallpapercave.com/wp/wp12390197.jpg");
background-size: 200%;
background-position: top left;
background-repeat: no-repeat;
background-attachment: local;
}}

[data-testid="stSidebar"] > div:first-child {{
background-image: url("data:image/png;base64,{img}");
background-position: center;
background-repeat: no-repeat;
background-attachment: fixed;
}}

[data-testid="stHeader"] {{
background: rgba(0,0,0,0);
}}

[data-testid="stToolbar"] {{
right: 2rem;
}}

.st-cg, .st-eb, .st-fj, .st-bc, .st-es, .st-fk,
.st-bd, .st-fc {{
background-color: #40BAD7;
}}

.st-cg, .st-eb, .st-fj, .st-bc, .st-es, .st-fk,
.st-bd, .st-fc {{
border-color: #40BAD7;

```

```

}}

.sidebar-content .stButton>button {{
background-color: #40BAD7 !important;
color: #40BAD7 !important;
border-color: #40BAD7 !important;
}}
</style>
"""
st.markdown(page_bg_img, unsafe_allow_html=True)
st.title("Clinical Data Analysis using NER")
document_type = st.sidebar.radio("Select Document Type:",
("Manual Text", "PDF", "Image"))
text = "" # Initialize text variable
if document_type == "Manual Text":
    st.sidebar.write("You selected Manual Text")
    manual_text = st.text_area("Enter text here:", "")
    text = manual_text.strip()
elif document_type == "PDF":
    st.sidebar.write("You selected PDF")
    uploaded_file = st.file_uploader("Upload a PDF file",
type=["pdf"])
    if uploaded_file is not None:
        text = extract_text_from_pdf(uploaded_file)
        st.write("Extracted text:")
        st.write(text)
elif document_type == "Image":
    st.sidebar.write("You selected Image")
    input_image = get_image()
    if input_image:
        text = perform_ocr(input_image)
        st.header("OCR Text Extraction Results:")

```

```

        st.write(text)
if st.sidebar.button("Entity Recognition"):
    if text:
        st.sidebar.subheader("Automated Entity
Recognition:")
        doc_bc5cdr, combined_entities = perform_ner(text)
        html = visualize_entities(doc_bc5cdr,
combined_entities)
        st.write(html, unsafe_allow_html=True)
    else:
        st.warning("Please input some text before
performing entity recognition.")
st.sidebar.markdown("---")
st.header("Question Answering")
question = st.text_input("Enter your query:")
if st.button("Search"):
    if text:
        if question:
            answer = perform_qa(question, text)
            st.subheader("Answer:")
            st.write(answer)
        else:
            st.warning("Please enter a query before
searching.")
    else:
        st.warning("Please input some
text before
querying.")
if st.sidebar.button("Query Generation"):
    if text:
        st.sidebar.subheader("Automated Query
Generation with answers:")

```

```

generated_queries, answers = generate_queries
(text)
st.subheader("Generated Queries with Answers:")
for i, (query, answer) in
enumerate(zip(generated_queries, answers)) {
    st.write(f"Query {i + 1}: {query}")
    st.write(f"Answer: {answer}")
else:
    st.warning("Please input some text
before performing entity recognition.")
if __name__ == "__main__":
    main()

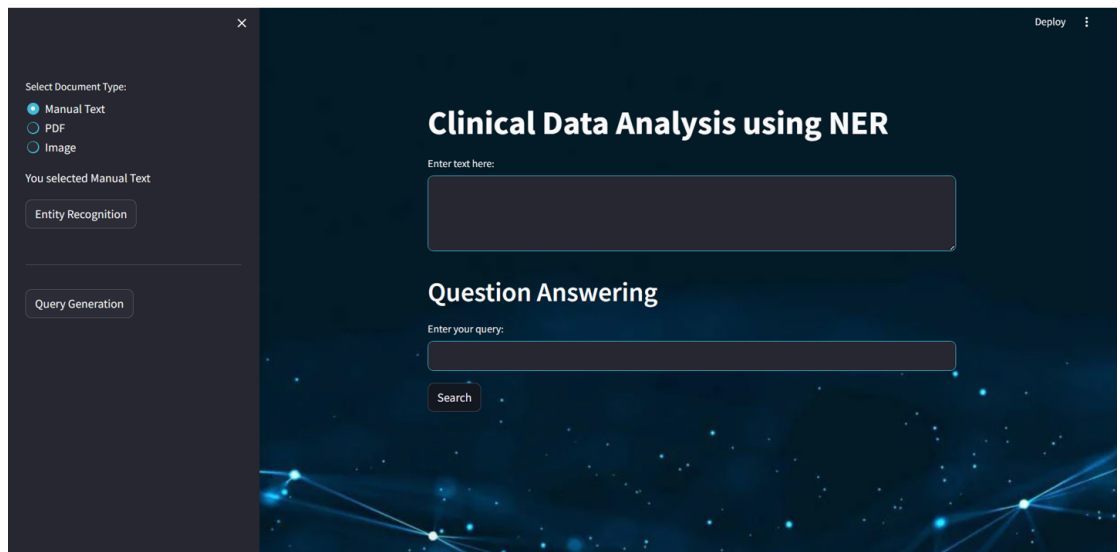
```

# CHAPTER 7

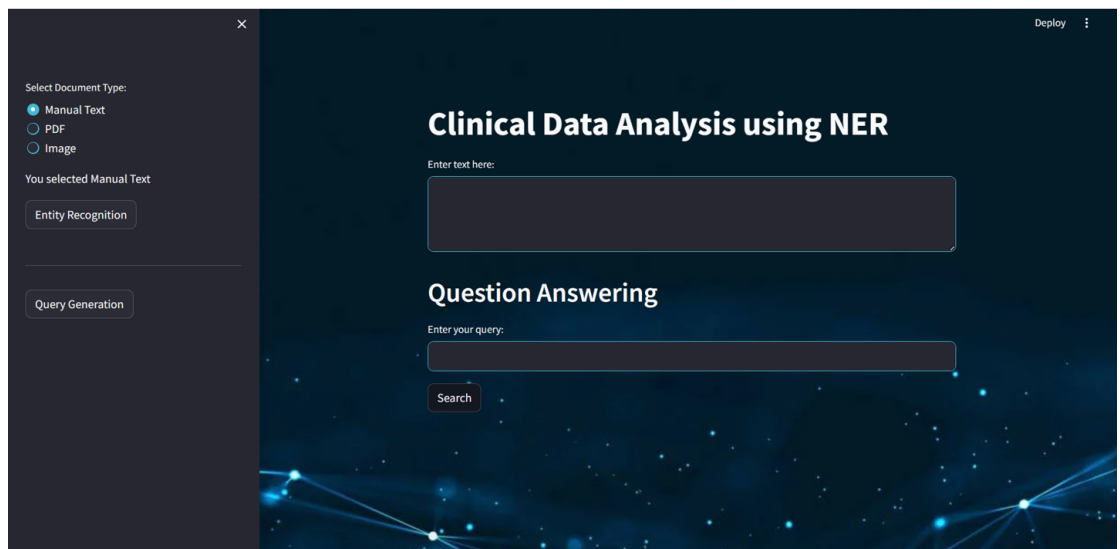
## OUTPUTS

### 7.1 Screenshots

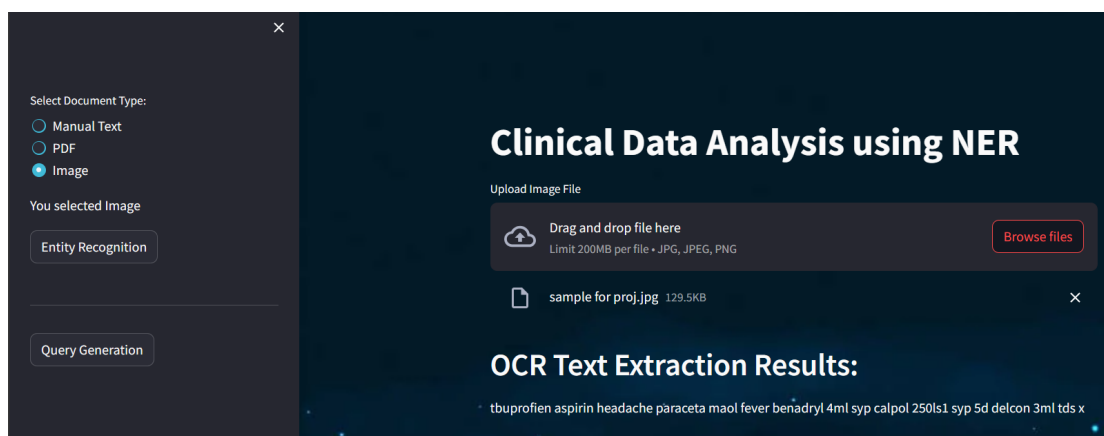
#### 7.1.1 User Interface



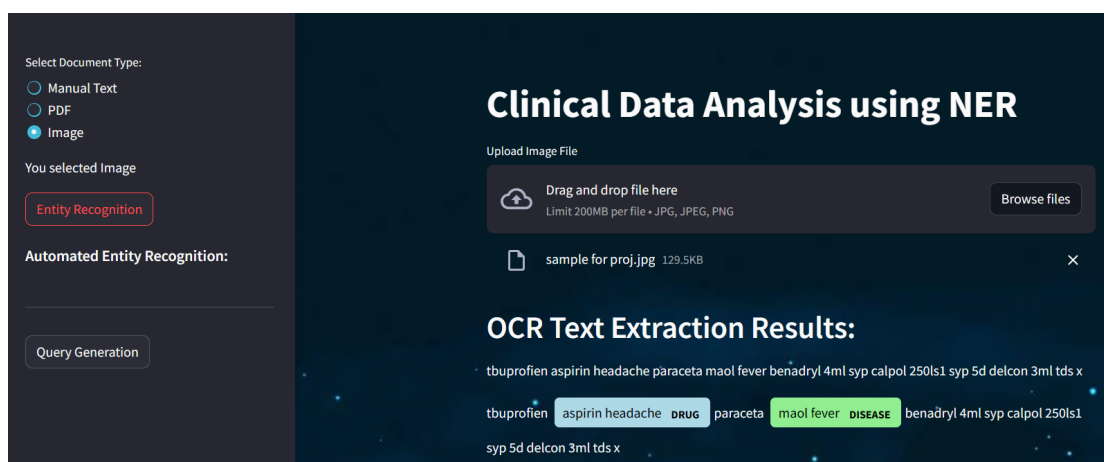
#### 7.1.2 NER from Manual Text



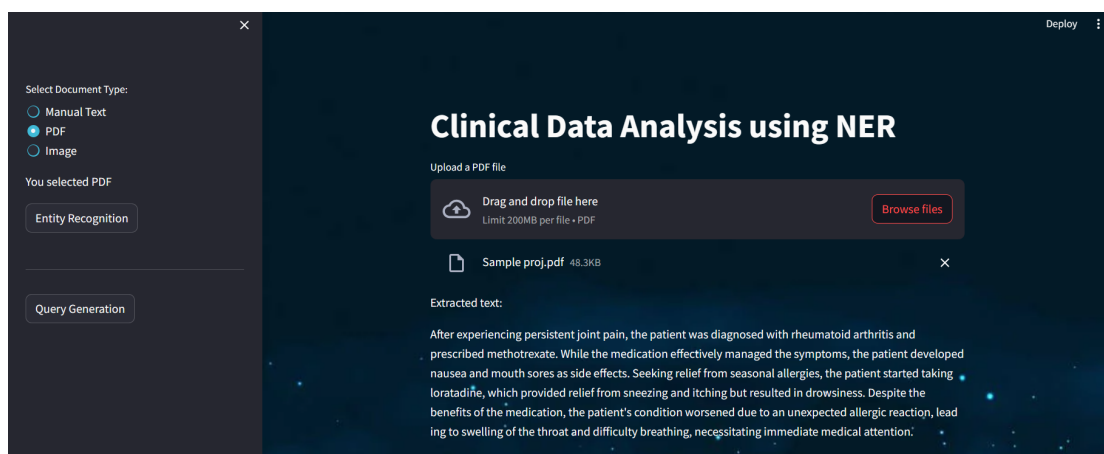
### 7.1.3 Extracting Handwritten text from an Image



### 7.1.4 NER from Extracted Text of Image



### 7.1.5 Extracting Text from a PDF file





### 7.1.6 NER from Extracted Text of PDF

The screenshot shows a web application interface for Named Entity Recognition (NER). On the left, there is a sidebar with the following elements:

- Select Document Type:** Three radio buttons: ☐ Manual Text, ☒ PDF, and ☐ Image.
- You selected PDF**
- Entity Recognition** (highlighted in red)
- Automated Entity Recognition:**
- Query Generation** (highlighted in red)

The main content area displays the extracted text from a PDF document, with Named Entities (NEs) highlighted in green boxes and labeled as **DISEASE** or **DRUG**. The text is as follows:

Extracted text:

After experiencing persistent joint pain, the patient was diagnosed with rheumatoid arthritis and prescribed methotrexate. While the medication effectively managed the symptoms, the patient developed nausea and mouth sores as side effects. Seeking relief from seasonal allergies, the patient started taking loratadine, which provided relief from sneezing and itching but resulted in drowsiness. Despite the benefits of the medication, the patient's condition worsened due to an unexpected allergic reaction, leading to swelling of the throat and difficulty breathing, necessitating immediate medical attention.

The entities identified are:

- pain** (DISEASE)
- rheumatoid arthritis** (DISEASE)
- methotrexate** (DRUG)
- nausea and mouth sores** (DISEASE)
- allergies** (DISEASE)
- sneezing** (DISEASE)
- itching** (DISEASE)
- loratadine** (DRUG)
- allergic reaction** (DISEASE)
- swelling** (DISEASE)
- throat** (DISEASE)

### 7.1.7 Automatic Query Generation along with Answers

The slide is titled **Generated Queries with Answers:** and displays three queries and their corresponding answers:

- Query 1: can loratadine cause itching**  
**Answer: drowsiness**
- Query 2: what is the side effect of methotrexate**  
**Answer: nausea and mouth sores**
- Query 3: side effects of methotrexate**  
**Answer: nausea and mouth sores**

### 7.1.8 Machine Reading Comprehension

## Clinical Data Analysis using NER

Enter text here:

developed nausea and mouth sores as side effects. Seeking relief from seasonal allergies, the patient started taking loratadine, which provided relief from sneezing and itching but resulted in drowsiness. Despite the benefits of the medication, the patient's condition worsened due to an unexpected allergic reaction leading to swelling.

## Question Answering

Enter your query:

what drug caused drowsiness

Search

**Answer:**

loratadine

# **CHAPTER 8**

## **RESULTS AND PERFORMANCE ANALYSIS**

### **8.1 Results and Performance Analysis**

Clinical data analysis, particularly for doctors' prescriptions, represents a crucial area where computational techniques can offer substantial support. In our project, we undertook the task of processing and analyzing doctors' prescriptions using NLP methods. We developed an integrated system capable of handling diverse document formats, including manual text input, PDF files, and images. Utilizing techniques such as OCR and entity recognition, we extracted pertinent information such as drug names and diseases mentioned in the prescriptions. Furthermore, we employed transformer-based models for question answering and query generation, facilitating efficient retrieval of relevant information from the prescriptions. Through thorough evaluation and analysis, we aim to enhance the utility of our system and provide valuable insights for healthcare professionals in their decision-making processes.

#### **8.1.1 Existing System**

The combination of LSTM and BERT presents a potent fusion of two powerful architectures in NLP. While LSTM excels in capturing sequential dependencies and long-term context in text data, BERT provides deep contextual representations by leveraging bidirectional information within a sentence. By integrating LSTM with BERT, the hybrid model can effectively capture both local features and long-range dependencies, enhancing its capability to tackle complex NLP tasks. This synergistic approach enables the model to adapt to specific tasks through fine-tuning and offers promising advancements in various applications such as text classification, and named entity recognition.

### 8.1.2 Proposed System

The combination of Bi-LSTM and BERT offers a robust solution for NLP tasks. Bi-LSTM, an extension of LSTM, captures both past and future context by processing sequences in both forward and backward directions, enabling it to grasp long-range dependencies in text. On the other hand, BERT provides deep contextual embeddings by leveraging bidirectional context within a sentence. By integrating Bi-LSTM with BERT, the hybrid model can effectively capture hierarchical features, local context, and long-range dependencies, thereby enhancing its performance on complex NLP tasks. This combination facilitates better understanding and representation of text data, making it suitable for applications such as text classification, named entity recognition, sentiment analysis, and more.

### 8.1.3 Accuracy

Accuracy is a metric used to evaluate the performance of a model in machine learning. It measures the percentage of correct predictions made by the model on the total number of predictions. In other words, accuracy indicates how well a model can classify data points into the correct categories. It is calculated by dividing the correct predictions by the total number of predictions and multiplying the result by 100 to get a percentage. Accuracy can be affected by various factors, such as the quality of the data, the complexity of the model, and the size of the dataset. A higher accuracy indicates that the model is performing better, while a lower accuracy suggests that the model needs improvement.

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

where the number of correct predictions refers to the number of instances that the model predicted the correct output and the total number of predictions is the total number of instances in the dataset that the model made predictions on. The accuracy is usually expressed as a percentage.

### 8.1.4 Precision

Precision is a metric used to evaluate the accuracy of a model's positive predictions. It measures the percentage of true positive predictions out of all positive predictions made by the model. In other words, precision indicates how well a model can identify the correct positive class. It is calculated by dividing the number of true positive predictions (TP) by the total number of false predictions (FP).

$$\text{Precision} = \frac{TP}{TP + FP}$$

Precision is important in situations where the cost of false positives is high, such as in medical diagnosis. A higher precision indicates that the model is making fewer false positive errors.

### 8.1.5 Recall

Recall is another metric used to evaluate the performance of a classification model. It measures the percentage of actual positive cases that are correctly identified by the model. In other words, recall tells us how many of the positive cases in the dataset were correctly identified as positive by the model. It is calculated by dividing the number of true positives (TP) by the sum of true positives and false negatives (FN), and multiplying the result by 100 to get a percentage.

$$\text{Recall} = \frac{TP}{TP + FN}$$

A high recall indicates that the model is good at identifying positive cases, while a low recall suggests that the model may be missing some positive cases.

### 8.1.6 F1-Score

The F1 score is a metric used to evaluate the overall performance of a classification model. It is calculated based on the precision and recall of the model's predictions. Precision is the proportion of true positive predictions out of all positive predictions, while recall is the proportion of true positive predictions out of all actual positives in the data. F1-score is the harmonic mean of precision and recall, which balances both metrics equally.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

It ranges from 0 to 1, with higher values indicating better performance.

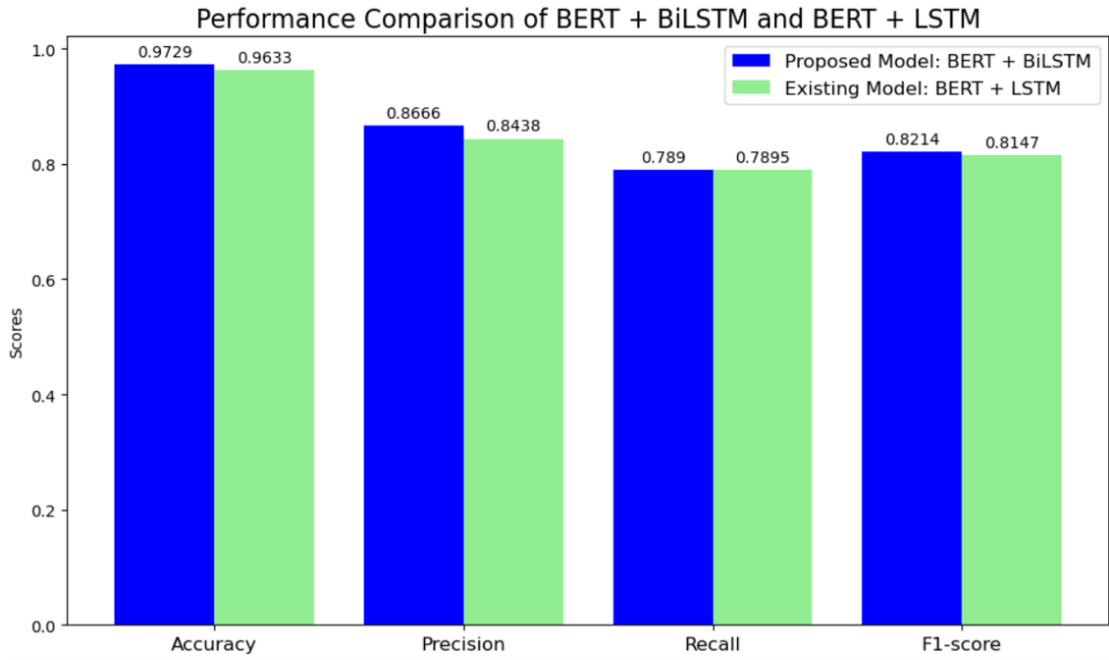


Figure 8.1: Performance Comparison of Existing and Proposed Model

	precision	recall	f1-score	support
0	0.98	0.99	0.99	103682
1	0.97	0.98	0.97	5385
2	0.84	0.73	0.78	4423
3	0.73	0.57	0.64	2424
4	0.88	0.60	0.71	404
accuracy			0.97	116318
macro avg	0.88	0.77	0.82	116318
weighted avg	0.97	0.97	0.97	116318

Figure 8.2: Classification Report of the Proposed Model

## CHAPTER 9

### CONCLUSION AND FUTURE SCOPE

#### 9.1 CONCLUSION AND FUTURE SCOPE

##### 9.1.1 Conclusion

The project showcases the application of natural language processing (NLP) techniques in processing and analyzing clinical text data, particularly doctors' handwritten prescriptions.

It mainly enables efficient extraction of relevant information from clinical documents for entity recognition and transformer-based models for question answering and query generation. The Streamlit-based interface provides a user-friendly platform for users to interact with the system, allowing manual text input, PDF file uploads, and image recognition.

Through entity recognition, named entities such as drugs and diseases are identified, offering valuable insights into the content of clinical documents. Additionally, the question-answering module generates answers to user queries based on the document context, enhancing the understanding of specific topics within the text.

##### 9.1.2 Future Scope

**Model Refinement:** Fine-tuning the transformer models on domain-specific clinical datasets can improve the accuracy of entity recognition, question answering, and query generation tasks.

**Integration with Electronic Health Records (EHR):** Integrating the system with EHR systems can enable real-time analysis of patient data, supporting healthcare providers in decision-making processes.

**Interactive Visualization:** Enhancing the Streamlit interface with interactive visualization features can improve user engagement and facilitate a better understanding of analyzed clinical data.

**Improving OCR Accuracy for handwritten text:** Handwritten text recognition can

be improved by training or fine-tuning models using large-scale handwriting image datasets, optimizing recognition for the intricate characteristics of unstructured medical text.

**Deployment and Scalability:** Deploying the application on cloud platforms and ensuring scalability can increase accessibility and support for large-scale data processing tasks.



## REFERENCES

- [1] Bo Guo and Huaming Liu. “Integration of natural and deep artificial cognitive models in medical images: BERT-based NER and relation extraction for electronic medical records”. In: *Frontiers in Neuroscience* 17 (2023), p. 1266771.
- [2] Cong Sun, Zhihao Yang, Lei Wang, Yin Zhang, Hongfei Lin, and Jian Wang. “Biomedical named entity recognition using BERT in the machine reading comprehension framework”. In: *Journal of Biomedical Informatics* 118 (2021), p. 103799.
- [3] Cong Sun, Zhihao Yang, Lei Wang, Yin Zhang, Hongfei Lin, and Jian Wang. “Mrc4bioer: joint extraction of biomedical entities and relations in the machine reading comprehension framework”. In: *Journal of Biomedical Informatics* 125 (2022), p. 103956.
- [4] Youngjin Jang, Hyeon-gu Lee, and Harksoo Kim. “Long multispan prediction model for machine reading comprehension in healthcare domain”. In: *Expert Systems with Applications* 215 (2023), p. 119300.
- [5] Yuefan Fei and Xiaolong Xu. “GFMRC: A machine reading comprehension model for named entity recognition”. In: *Pattern Recognition Letters* 172 (2023), pp. 97–105.
- [6] Xiangyang Li, Huan Zhang, and Xiao-Hua Zhou. “Chinese clinical named entity recognition with variant neural structures based on BERT methods”. In: *Journal of biomedical informatics* 107 (2020), p. 103422.
- [7] Qiang Zhang, Yong Sun, Linlin Zhang, Yanfei Jiao, and Yue Tian. “Named entity recognition method in health preserving field based on BERT”. In: *Procedia Computer Science* 183 (2021), pp. 212–220.
- [8] Kai Xu, Zhenguo Yang, Peipei Kang, Qi Wang, and Wenying Liu. “Document-level attention-based BiLSTM-CRF incorporating disease dictionary for disease named entity recognition”. In: *Computers in biology and medicine* 108 (2019), pp. 122–132.

- [9] Ilaria Bartolini, Vincenzo Moscato, Marco Postiglione, Giancarlo Sperlì, and Andrea Vignali. “Data augmentation via context similarity: An application to biomedical Named Entity Recognition”. In: *Information Systems* 119 (2023), p. 102291.
- [10] Carson Tao, Michele Filannino, and Özlem Uzuner. “Prescription extraction using CRFs and word embeddings”. In: *Journal of biomedical informatics* 72 (2017), pp. 60–66.
- [11] Namrata Nath, Sang-Heon Lee, and Ivan Lee. “NEAR: Named entity and attribute recognition of clinical concepts”. In: *Journal of biomedical informatics* 130 (2022), p. 104092.
- [12] Yoojoong Kim, Jong-Ho Kim, Young-Min Kim, Sanghoun Song, and Hyung Joon Joo. “Predicting medical specialty from text based on a domain-specific pre-trained BERT”. In: *International Journal of Medical Informatics* 170 (2023), p. 104956.
- [13] Rey-Long Liu and Yi-Chih Huang. “Medical query generation by term–category correlation”. In: *Information processing & management* 47.1 (2011), pp. 68–79.
- [14] Veysel Kocaman and David Talby. “Accurate clinical and biomedical named entity recognition at scale”. In: *Software Impacts* 13 (2022), p. 100373.
- [15] David Fraile Navarro, Kiran Ijaz, Dana Rezazadegan, Hania Rahimi-Ardabili, Mark Dras, Enrico Coiera, and Shlomo Berkovsky. “Clinical named entity recognition and relation extraction using natural language processing of medical free text: A systematic review”. In: *International Journal of Medical Informatics* (2023), p. 105122.
- [16] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. “Domain-specific language model pretraining for biomedical natural language processing”. In: *ACM Transactions on Computing for Healthcare (HEALTH)* 3.1 (2021), pp. 1–23.

- [17] Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wieggers, and Zhiyong Lu. “BioCreative V CDR task corpus: a resource for chemical disease relation extraction”. In: *Database* 2016 (2016).
- [18] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* 36.4 (2020), pp. 1234–1240.
- [19] Batuhan Balci, Dan Saadati, and Dan Shiferaw. “Handwritten text recognition using deep learning”. In: *CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University, Course Project Report, Spring* (2017), pp. 752–759.